DEMOKRITOS

HELLENIC REPUBLIC
National and Kapodistrian
University of Athens
EST. 1837

Music Information Retrieval using Machine Learning & Convolutional Neural Networks

MSc. Control and Computing

Master Thesis

by

Rodo Kasidiari
(2019505)

Department of Physics
Department of Informatics & Telecommunications

Athens 2022

Present thesis is performed in collaboration with the Multimedia Analysis Group of the Computational Intelligence Lab (MagCIL) of the Institute of Informatics and Telecommunications, of the National Center for Scientific Research "Demokritos".

Acknowledgments

I would like to express my special thanks and gratitude to my Professor Mr. Reisis who has guided me through my whole journey of getting to know Computer Science. As well as to Mr. Perantonis who taught me everything I need to know about Machine Learning and Neural Networks and aroused my interest for further research.

I also wish to especially thank the Multimedia Analysis Group of the Computational Intelligence Lab (MagCIL) of the the Institute of Informatics and Telecommunications, of the National Center for Scientific Research "Demokritos", and especially Mr. Giannakopoulos, who gave me the opportunity to explore an interesting and at the same time fun area of Machine Learning. His consistent guidelines and advice played a key role in preparing the current thesis.

Last but not least, I want to thank my parents who believed in me and supported me throughout my student years.

Athens 2022
Rodo Kasidiari

Contents

Abstract

In this thesis an attempt is made to analyze and retrieve musical information using Machine Learning algorithms and Convolutional Neural Networks. The goal is to recognize and classify musical tracks based on their emotional impact, their genre and their similarity between others in a song collection. To accomplish this goal, models of Convolutional Neural Networks have been built related to the valence, the energy, the danceability and the genre of a song. The genre classes are created using clustering methods. All models are trained on a large volume of musical data using a Convolutional Neural Network from the Deep Audio Features python library. Finally, the evaluation of the models is based on a set of data created by user interaction comparing triplets of songs and deciding which song according to their opinion is the least compatible with the other two. A music content-based application has been created that contributed to the evaluation procedure of the CNN models so that songs of a small collection could be visualized and compared to each other according to some of their characteristics.

Keywords— Machine Learning, Music, Information Retrieval, Neural Networks, Genre, Clustering, Classification, Similarity, Features, Audio Signals, Mel Spectrogram

Περίληψη

Στην παρούσα διπλωματική εργασία γίνεται προσπάθεια ανάλυσης και ανάκτησης χαρακτηριστικών μουσικού περιεχομένου με τη χρήση αλγορίθμων Μηχανικής Μάθησης και Συνελλικτικών Νευρωνικών Δικτύων. Στόχος είναι η αναγνώριση μουσικών κομματιών και η κατηγοριοποίηση τους με βάση τα συναισθήματα που προκαλούν, το είδος και την ομοιότητά τους με άλλα τραγούδια μιας συλλογής. Για την επίτευξη αυτού του στόχου εκπαιδεύτηκαν μοντέλα Συνελικτικών Νευρωνικών Δικτύων που αφορούν το συναισθηματικό πρόσημο - σθένος, την ενέργεια, την χορευτικότητα και το είδος ενός τραγουδιού. Οι κλάσεις που αφορούν το είδος, δημιουργήθηκαν με τη χρήση μεθόδων ομαδοποίησης σε κλάσεις (clustering). Όλα τα μοντέλα εκπαιδεύτηκαν σε μεγάλο όγκο μουσικών δεδομένων με τη χρήση Συνελικτικού Νευρωνικού Δικτύου της python βιβλιοθήκης Deep Audio Features. Τέλος, η αξιολόγηση των μοντέλων βασίστηκε σε ένα σύνολο δεδομένων που δημιουργήθηκε από την αλληλεπίδραση χρηστών συγκρίνοντας τριπλέτες τραγουδιών και αποφασίζοντας ποιο τραγούδι κατά τη γνώμη τους ήταν το λιγότερο ταιριαστό με τα υπόλοιπα δύο. Στην διαδικασία της αξιολόγησης των μοντέλων συνέβαλε επίσης η δημιουργία εφαρμογής βασισμένης στο μουσικό περιεχόμενο με στόχο την οπτικοποίηση και σύγκριση ως προς ορισμένα χαρακτηριστικά τους, τραγουδιών μιας μικρής συλλογής.

Λέξεις Κλειδιά— Μηχανική Μάθηση, Μουσική, Ανάκτηση Πληροφορίας, Νευρωνικά Δίκτυα, Είδος, Ομαδοποίηση, Κατηγοριοποίηση, Ομοιότητα, Χαρακτηριστικά, Σήματα Ήχου, Μελ Φασματογράφημα

Chapter 1

Introduction

1.1 Music Information Retrieval - The Problem

Music holds an exceptional place in most people's life as it usually accompanies our daily activities and mood states. Someone could point out that as limitless the spectrum of our emotions is, so broaden the music selection could be. Thus, apart from musicology, other fields like psychology, data science and signals processing are interested in researching and exploring Music and its potentials. There is though, a research field young yet popular, which is engaged with techniques for the analysis and the process of music data and it is called Music Information Retrieval (MIR)[1]. MIR is an interdisciplinary field interfering with all the aforementioned scientific fields. Section 2.3 provides a detailed introduction on MIR tasks that this thesis deals with as well as a broad overview of the challenges that may lurk.

With the rapid development of technology and digital information, the way we store, select and listen to music is changing rapidly, while streaming platforms are now playing an active role in our lives. The data that is daily generated by streaming platforms is much and varied. Each user selects and stores, in digital collections, the songs that reflect their preferences. These huge music collections should be properly grouped and labeled. Therefore, a need is raised to manage large volumes of music data and extract as much information as possible, in order to optimize the users' experience. Consequently, MIR aims to utilize the data that is available, and develop new tools and techniques regarding the recommendation, classification or recognition of a music track, thus facilitating the users' life easier and taking music streaming services to the next level.

Most recommendation systems use models that are based on the users' preferences and the collections they create, the artists they choose or the songs they usually listen to. These models are called user-centric and extract information about the similarity between the songs that they are going to be recommended by them. But what do we define as "similarity"? Choosing a song can be either personal or random. A deeper analysis of music data, as in the case of content-based models, can give us additional information to categorize music into genres, to define similarities, to query for a song by humming, by example or to match a music track with a specific emotion. Hence, we use the so-called audio features of a musical signal (see subsection 2.2.2) as the fingerprint of a song. Features are extracted from the audio signal as a spectral representation of the timbre, the pitch, the duration or the amplitude, the melody, the rhythm, or the harmony. Finally, these features are used to train machine learning models and perform tasks regarding the estimation of the similarity of one song with another or the labeling of a song by the feelings it can trigger or the genres it belongs to.

This thesis focuses on the content-based analysis of music audio. Chapter 1 presents the MIR problem and the motivation to perform further research in order to increase the accuracy of the feature extraction procedure using different ways and techniques. Chapter 2 gives an overview of the theoretical background regarding Machine Learning, Audio Signals and Music Information Retrieval. To continue, chapter 3 provides in-depth illustration of the proposed methodology and the tools used in order to create and train the machine learning models, to extract audio features from a given music dataset and finally, to compare different songs in terms of their genre, their mood or their similarity with others. Chapter 4 reports the experimental results and analyzes the procedure followed to perform evaluation and reach our conclusions. Lastly, Chapter 5 conveys the experience gained from the experiments performed, the assumptions underlying our results and what our future work should be.

Chapter 2

Theoretical Background

2.1 Machine Learning

Learning is the process of acquiring new understanding, knowledge, behaviors, skills, values, attitudes, and preferences. Besides humans, animals or even some plants, and machines are now able to learn different tasks and "react" according to the input data.

Machine Learning is a scientific field that, although it appears to have grown during the latest decade, has been introduced by **Arthur Samuel**, in the early 50's. Since then, many studies and much research, related to the increasing need to process, manipulate, and analyze the plethora of data are being produced, nowadays have been conducted.

Machine Learning is about developing algorithms and methods that enable computers, and machines in general, to "learn" from data[2].It is highly connected to computer science, statistics, information theory and automation. Those are combined in terms of algorithms and software, allowing scientists to cope with tasks which are too complicated to be solved by traditional programming. The knowledge obtained from data, offers the opportunity to find patterns or make predictions about future events in a more efficient way.

There are three types of Machine Learning: Supervised Learning, Unsupervised Learning and Reinforcement Learning, as Figure 2.1 suggests. This thesis focuses only on **Supervised** and **Unsupervised Learning**, which are being further described in the corresponding subsections of this chapter.

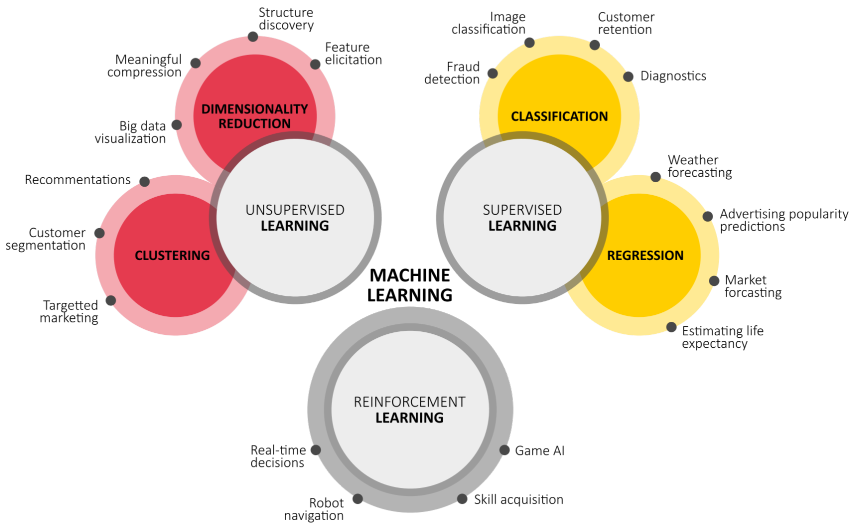


Figure 2.1: Different subfields of Machine Learning

2.1.1 Supervised Learning

Supervised Learning corresponds to training data that are class labeled. The data is represented in pairs of observation (x_n, y_n), n = 1, 2, ..., N where x_n is a vector, or a set of variables, called input variables or features, and y_n are the target variables that can also be a vector. The objective is to create a model, capable of learning from that training set and making predictions, **categorizing** (estimating a functional mapping) an unseen – testing set of the same type, to the respective output class. There are two main kinds of supervised machine learning, **Classification** and **Regression**. Within the context of this thesis, only classification is taken into consideration.

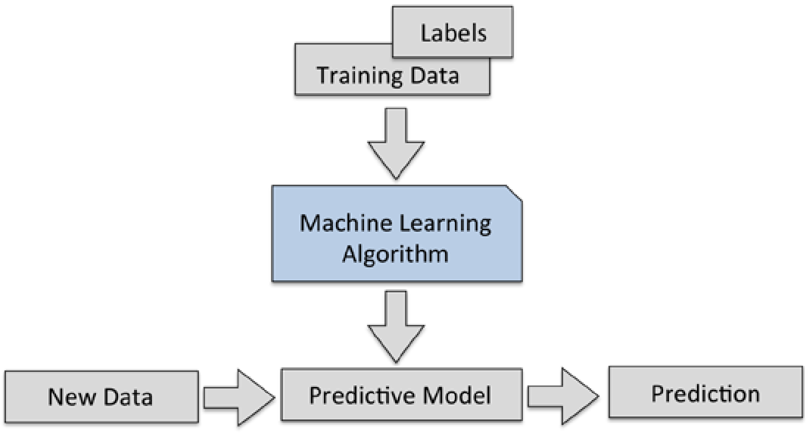


Figure 2.2: Supervised Learning

2.1.1.1 Classification

Classification in terms of machine learning is a supervised learning approach in which a trained model makes predictions over another dataset and categorizes each data, accordingly. A **classifier** is the model used for classification, and its output signal is the target class that the model predicts for its new unseen sample of testing data.

Classification can be described as a procedure with two stages, learning and classification/training. Some examples of such algorithms are Logistic Regression, Naïve Bayes Algorithm, Stochastic Gradient Descent, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machine and Neural Networks (see subsection 2.1.3).

To be more explanatory, here is a simple example. Suppose there are two folders with songs of two different genres, pop, and rock. Each folder represents a target class. The songs are the dataset fed to a classifier, which, afterwards, make predictions about an unknown test dataset. To start with, the dataset needs to be processed and each song needs to be represented as a features vector. When the training procedure takes place, the dataset is fed into the classifier, and the model learns to distinguish each song according to its features and its target class. Following this procedure, the unknown test set, which may be a folder with mixed genres songs, is now fed to the model. The classifiers' goal is to categorize each of the test data song according to what previously learnt, into pop or rock categories.

In the case of a binary classification task, similar to the one described above, the number of categories is two. Otherwise, for more than two categories, the case is the multi-class classification task. For Music Information Retrieval (MIR) challenges, the classification tasks are usually multi-class. Figure 2.3 indicates both classification cases. As can be seen, the goal is to categorize the input data to the corresponding classes correctly.

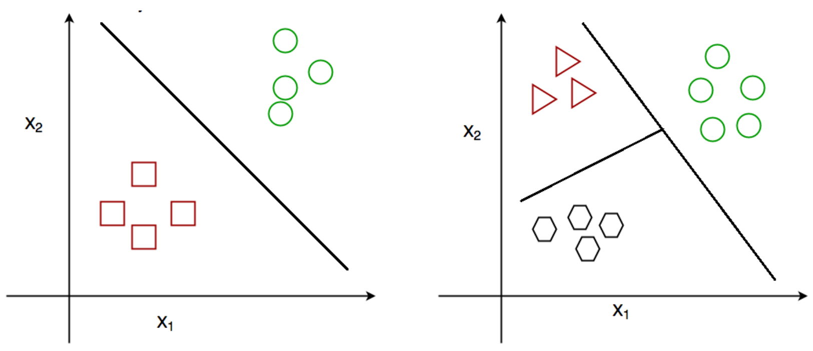


Figure 2.3: Binary Classification and Multi-class classification

2.1.2 Unsupervised Learning

In Unsupervised Learning, the data is *unstructured* and *unlabeled*. In other words, the output or label information are not available. Main purpose of this learning method is to resolve the hidden structure of the dataset, and, thus, retrieve meaningful information. The most common facets of Unsupervised Learning are *Clustering* and *Dimensionality Reduction*. Both learning methods are applicable for the progress of this thesis' experiments and are further analyzed in the following paragraphs.

2.1.2.1 Clustering

Clustering is a data processing model that organizes an unstructured data set into meaningful subgroups (clusters), without having any prior information about their cluster memberships. Each derived cluster groups data points that share a degree of **similarity** with each other but look quite dissimilar compared to the data points that belong to another cluster. The task of clustering is to assign a number of points, $x_1, ..., x_N$, into K groups or clusters, as one can see in Figure 2.4. Clustering is a specific allocation of the points to clusters. The same cluster for points indicates similarity and K needs to be provided as input to the system.

One of the most classic clustering schemes is the **k-means** algorithm. The goal of this algorithm is to represent each cluster by the vector of mean attribute values of all training instances for numeric attributes and by the vector of modal (most frequent) values for nominal attributes that are assigned to that cluster. Further explained, k-means algorithm performs an iterative process that for each training instance assigns to it the "closest" cluster relating to the specified distance function which is applied to the instance and cluster center. The process continues by recalculating the cluster centers as the mean attribute value vectors of the instances that are assigned to the particular clusters. The initialization of the cluster centers is done randomly by picking a specific number of training instances, **k**, for instance. If no change is applied to the cluster the process stops. In practice, we choose the number of iterations to be from 3 to 36.

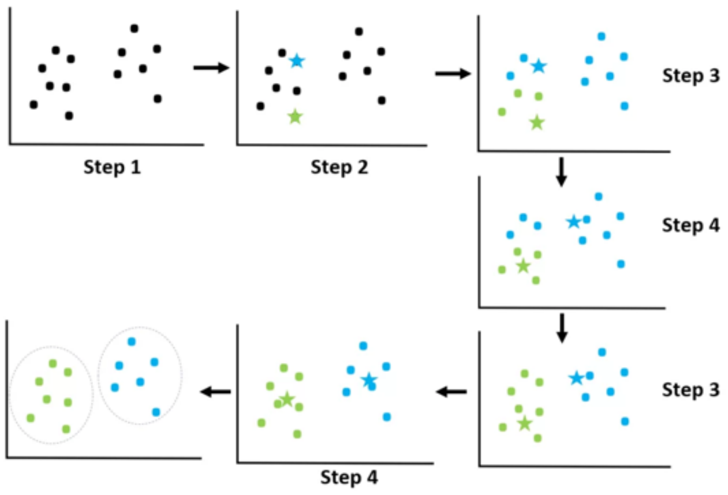


Figure 2.4: Unsupervised Machine Learning - Clustering

2.1.2.2 Dimensionality Reduction

Dimensionality Reduction is a form of unsupervised learning data preprocessing technique that reduces, as the name suggests, the dimensionality of the features vector (predictor variables). The aim is to reveal a particular structure of the data in case of lower memory or computational performance. Occasionally, a side-effect of reducing the dimensions is the reduction of the predictive performance of the models. Usual algorithms for dimensionality reduction are **Principal component analysis (PCA), Autoencoder (AE), T-distributed Stochastic Neighbor Embedding (t-SNE)**. In this thesis, only PCA and t-SNE are used for experimental purposes. AE is to be examined in our future work.

PCA is a technique used to reduce the number of dimensions in a dataset while retaining most information. It is an unsupervised method seeking linear projection that preserves the variance of the samples, without using class information. Mathematically speaking, the eigenvalues of the data's covariance matrix are computed, and the most valuable eigenvectors are used to project the feature space to a space of lower dimension. In general, it seeks to provide a minimum number of variables that keeps the maximum amount of variation or information about how the original data is distributed. [3]

t-SNE, on the other hand, is also an unsupervised technique and is commonly used for data visualization. Although the mathematical complexity is high, we can easily understand that it embeds the points from a higher dimension to a lower dimension trying to preserve the neighborhood of that point. This is achieved by preserving the Local structure of data while minimizing the Kullback–Leibler divergence (KL divergence) between the two distributions respecting the location of the points.[4]

2.1.3 Neural Networks

Artificial neural networks are computational models that structurally and conceptually mimic the functioning of the human brain. They resemble the biological nervous system in terms of acquiring information from the surroundings through learning algorithms. Neurons are the main processing units for the acquired knowledge. Neurons in the human brain are in charge of transmitting information between different areas of the brain and to the rest of the nervous system. The acquired knowledge stimulates the creation of a new synapse. Synapses are connecting links between neurons, and they are characterized by a synaptic weight. The modification of the synaptic weights of the network is the duty of learning algorithms. By the time a new type of information comes, a new synapse is created and other may die or be modified. Thus, the topology of the neural network changes, following a suchlike procedure as the human brain. The last characteristic is what we call plasticity and is in charge of our brain's development. [5]

2.1.3.1 Perceptron

The perceptron is invented by Frank Rosenblatt in 1958, as the simplest form of a feed-forward neural network linear classifier and the first supervised learning model. Its architecture contains of a single neuron, the input features and the corresponding synaptic weights. Then the external bias is added, and an activation function is applied on the result, forming the expected output. Figure 2.5 shows the fundamental representation of a perceptron.

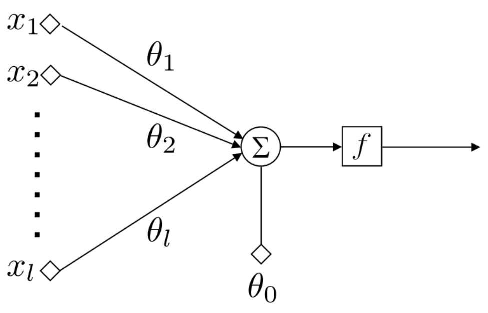


Figure 2.5: Simple neuron/perceptron architecture

Due to its simplicity, perceptron can only be effective for linearly separable classes. The hypothesis is built on to the binary classification but if the output(computation) layer of the perceptron is expanded in a way to include more than one neuron, we could perform classification to more than two classes.

The inputs of a neuron can either be features from a dataset or outputs from a previous layer's neurons. Neural networks have the advantage of the iterative learning process. Small batches of training data are presented to the network while the weights associated with the input values are adjusting until all are presented. This is described as the learning phase. The most popular algorithm to train neural networks is back-propagation [6] the weights of a neural network are fine-tuned based on the error rate obtained in the previous training epoch, by calculating the gradient of a loss function with respect to all the network's weights. The benefit of this method is that by tuning the weights properly we can reduce error rates and make the model reliable by increasing its generalization.

Solving more complex patterns and capturing non-linearity can be tricky. Thus, additional layers, known as the hidden layers, are introduced to the neural network architecture (multilayer perceptron). Figure 2.6 shows the architecture of a feed forward neural network consisting of only one hidden layer.

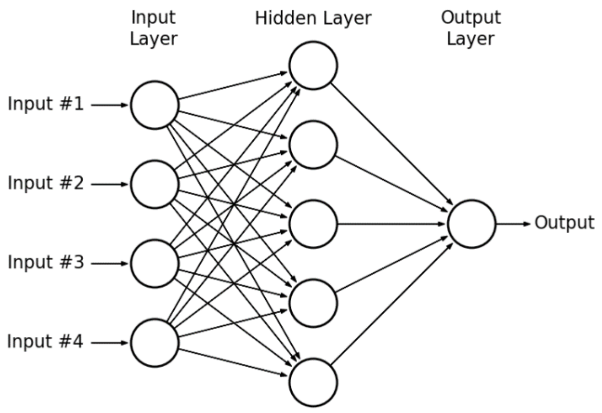


Figure 2.6: Feed-forward neural network with one hidden layer

Architectures that consist of multiple hidden layers are known as Deep Neural Networks(DNN) [2]. Suppose the network is successfully trained and the dataset is sufficiently large, DNN are in position to learn all the possible mappings and make quite accurate predictions.

2.1.3.2 Convolutional Neural Networks

One of the most famous types of neural networks that are widely used for image, speech, or audio signal analysis, are Convolutional Neural Networks (CNN). Their main characteristic, as their name indicates, is the use of convolution in at least one of their layers, instead of inner matrix multiplication. [2] Key to this kind of NN is the combined learning of dataset's features and neural network's parameters. [6]

The design of CNN neural networks is inspired specifically by the cat's visual cortex which has a complex cell arrangement. This arrangement requires the connection of certain neuronal cells that are sensitive to specific sub-regions of the visible field, which act as local 'filters' in these sub-regions. In this way, the cat takes advantage of the large spatial correlation of the various sub-areas, resulting in a better space awareness. In essence, CNN networks are variants of multi-layer perceptron networks as well as the latest neuron levels of a CNN usually is a multi-layer perceptron network. The differentiation lies in the levels of neurons that preceded by multi-layer perceptron, which as in the above example of the optical cortex of the cat, we can perceive them as filters in specific subsets of a dataset. After all, in their most basic form, CNN networks can be described as networks that use multiple copies of the same neurons. This enables computer processing of large models and data, while keeping the number of parameters to identification relatively small. In a standard convolutional neural network architecture, two main categories are noticed:

- The Convolutional Layers. Convolutional Layers consist of a neurons' network. Each convolutional level is the input from a network of its previous level whose weights remain the same for all subsequent network's neurons. In this way it is understood that each level works in a convolutional way compared to the previous level, with the size of the applied network being different in each one of them. In the case of two-dimensional data, networks must be square, for 3D data cubic and so on.
- The Pooling Layers. Pooling Layers are usually placed after convolutional layers and accept as input, the output of the previous convolutional layer. Their output is a unique value that can be the maximum value of the network, the average value, a linear regression etc. Layers that return the maximum are called Max-Pooling Layers.

Clearly, after any architecture and sequence of convolutional or pooling layers, the set of neurons of the last level becomes input to a multi-layer perceptron network or any other feed-forward network. In particular, if the problem we have concerns a 2D dataset or images(pattern recognition, video analysis, classification, etc.), then convolutional layers and the other parts of the network, such as the convilutional filters, kernels, are also 2D.

Furthermore, we try to describe a basic CNN architecture and the mathematical operations that take place as Figure 2.7 indicates.

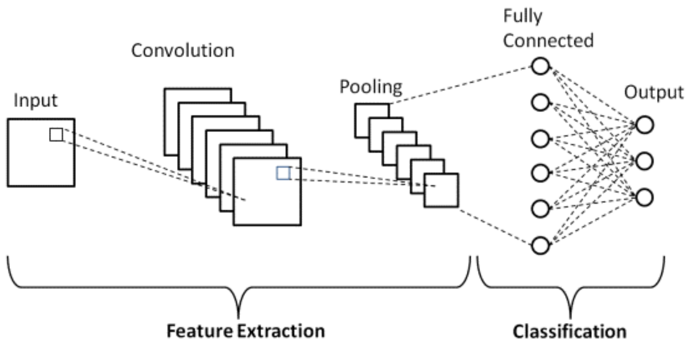


Figure 2.7: A basic Convolutional Neural Network used for classification

Beginning with, the convolution layer detects local features at distinct regions of the input by performing a dot product (convolution) between the set of learnable parameters known as a kernel, and the feature map. A sub-window of the input that produces the feature, is called the Receptive Field (RF). RF is used to associate an output feature (of any layer) to the input region (patch).

In essence, each local RF must be learned and analyzed by each hidden neuron. The first connection is now created and the RF "slides" along the rest input matrix by a fixed value called stride. The same procedure continues until all hidden neurons correlate with a connection of a local receptive field of the input layer. Below, Figure 2.8 represents the operation of convolution.

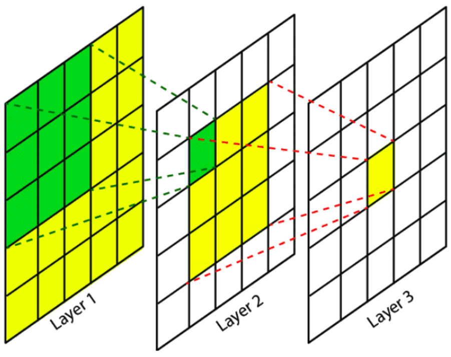


Figure 2.8: Representation of the convolution operation

Afterwards, the pooling layer reduces the dimensions or as we say, downsamples, the feature map. This results in a scaled down the spatial size of the representation and thus decreasing the amount of computation and weights.

Finally, convolution and pooling are used only for feature extraction and reduce the number of parameters of the original inputs. For the classification part, we need a more specific output. Hence, a fully-connected layer is applied producing an 1-dimensional output array that corresponds to the number of classes we have in our problem each time.

2.2 Audio Signals

Section 2.2 guides us through a physical and mathematical approach of the basic theory of audio signals.

Audio Signals are the electrical equivalent of sound. Sound is an energy form that is created by the pressure variations (sound waves) in a medium, for example the air. More specifically, it reflects the displacement of air molecules away from their equilibrium positions (vibrations). The region of compression (high pressure) and the region of rarefaction (low pressure) that are caused due to this movement, create what we call **sound** and what we, as humans, and other living creatures can **hear**. It is important to note that sound waves need a medium to travel (air, water or solid). Their main characteristics are frequency, wavelength, amplitude, sound pressure, speed and direction. We currently want to focus on the frequency and the amplitude of a sound wave. Amplitude is the extreme position a molecule can be found at, during its oscillation, as it can be seen in 2.9. Frequency is the number of waves that pass a molecule's equilibrium in a given amount of time, completing a whole oscillation and it is measured in Hertz (Hz).[7]

On the other hand, **audio signals** are the electrical variations of voltage. Their frequencies vary in the audio frequency range between 20 and 20,000Hz which are the limits of *human hearing*. Following subsections analyze the representation of audio signals and introduce their main features.

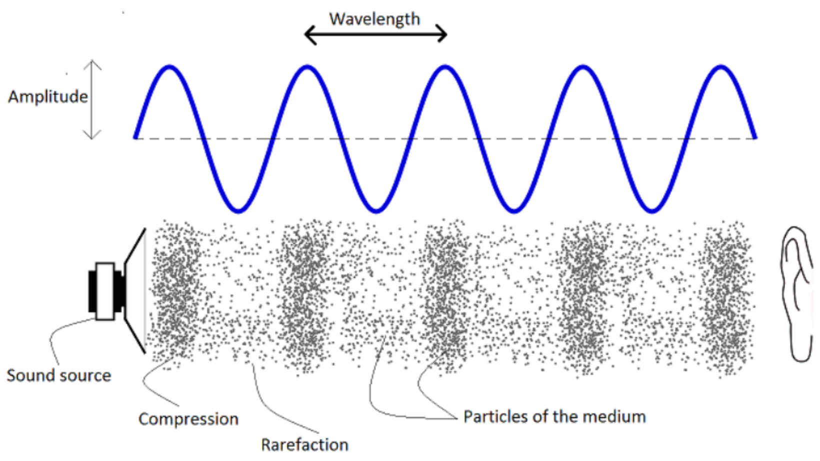


Figure 2.9: Physical interpretation of the propagation of a sound wave

2.2.1 Audio Signal Representation

It is common, when referring to audio signals, to refer to digital audio signals because with the help of technology we can easily imitate, transmit and encode sound waves in digital formats. Digital audio signals are **discrete-time** signals. Their exact value is no longer taken in a random instance of time as it is for the continuous signals(analog signals). Time instances are now specific and not necessarily equidistant. Thus, discrete points represent the amplitude of the waveform rather than a continuous sinusoidal wave. From a mathematical point of view, a discrete signal is described from a function as x[n], where n is an independent variable that represents the discrete time and takes only integer values in $(-\infty, +\infty)$. The procedure we follow, in order to create a discrete signal from its continuous equivalent, is called sampling. [8] Figure 2.10 shows the way an analog audio signal is sampled, in order to create the digital one.

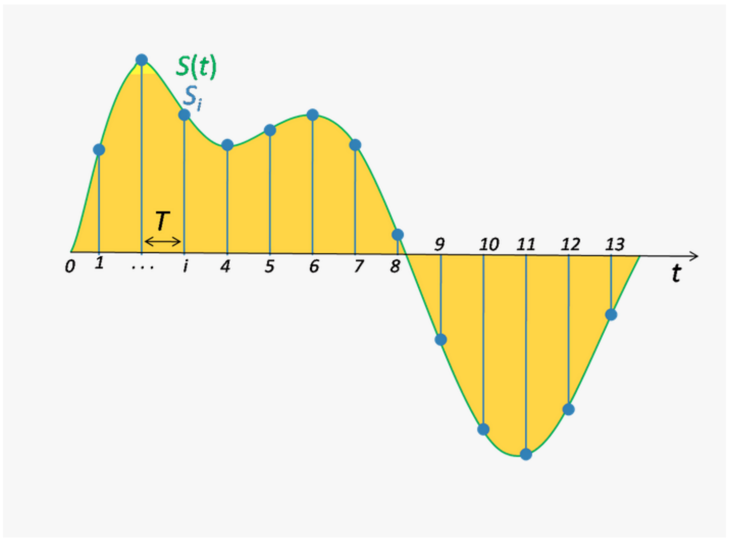


Figure 2.10: Sampling

As already mentioned, the discrete points are samples that represent the signal and correspond to the values of the amplitude that has been recorded in specific time glances. This is the only information we now have for this signal, and all the other between those time glances are in a way gone. Thus, the more the samples the better the quality that the representation has. For this reason, sampling frequency is of primordial role in the sampling procedure. Harry Nyquist, a great physicist and electronic engineer, proposed the *Nyquist rate* that ensures the success of sampling an analog signal. Nyquist rate is the lowest frequency we could have for the waveform to be accurately reconstructed. This lowest bound is equal to twice the maximum frequency of the signal and prevent distortion(aliasing) to be introduced in the reconstructed signal. [9]

To analyze and process digital signals it is important to represent the signal in the frequency domain (spectral representation). For this reason, Discrete Fourier Transform (DFT), helps us understand the signal through its frequency information by converting it to individual spectral components. To be more specific, the importance of DFT lies in the fact that we can easily calculate a signal's frequency spectrum. Doing so, we have direct access to encoded information in the frequency, phase, and amplitude of the component sinusoids. This type of encoding can be found, for example, in signals of human speech and hearing. Furthermore, the DFT can find a system's frequency response from the system's impulse response, and vice versa. This allows systems to be analyzed in the frequency domain, just as convolution allows systems to be analyzed in the time domain. The best implementation of DFT is Fast Fourier Transform (FFT), a computationally efficient algorithm that plays a leading role in digital signal processing. FFT samples a signal over a period of time and divides it into its frequency components. These components are single sinusoidal oscillations at distinct frequencies each with their own amplitude and phase. Over the time period measured, the signal contains 3 distinct dominant frequencies. This is shown in the following Figure 2.11

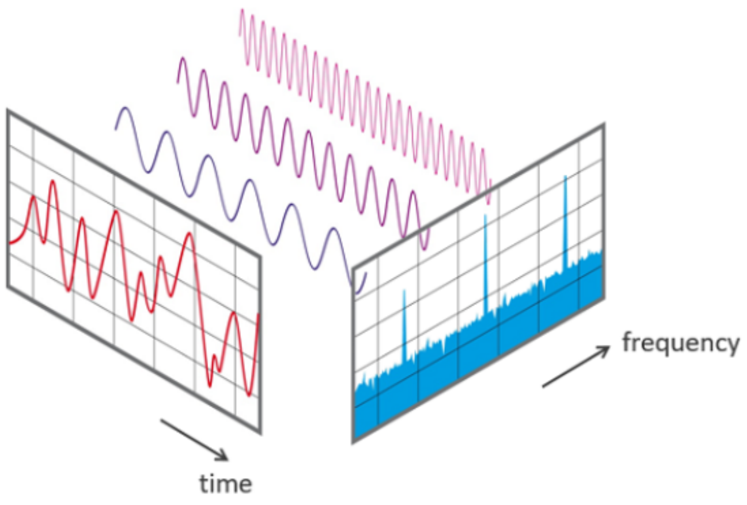


Figure 2.11: Time and frequency signal representation

Following, we try to give the mathematical approach of the DFT[9].

$$X(k) = \sum_{n=0}^{N-1} x[n]exp(-j\frac{2\pi}{N}kn), \quad k = 0, ..., N-1 \tag{2.1}$$

where $x[n]$ indicates a N-samples-long, discrete-time signal and $j \equiv \sqrt{-1}$ the imaginary number. So, Fourier Transform of a discrete-time signal $x[n]$ describes the analysis of the signal into its frequency components and allows the formulation of a linear combination of complex exponential functions whose frequencies are infinitely distant from each other on the axis of frequencies. In most feature extraction cases, the magnitude of the DFT coefficients, play the main role. [8]

2.2.2 Audio Features

As already explained in the previous sections, audio signals can be represented by time or frequency. In each domain we can distinguish features that are useful to us to analyze, process and produce any audio signal. These features are shown in Figure 2.12. Especially in machine learning tasks, it is important to perform **feature extraction** in order to collect the appropriate information of the under-investigation dataset. In audio analysis, the goal is to extract suitable features that can represent the original signal with such details, while reducing the volume of data. Throughout subsection 2.2.2, we highlight and cite the most valuable features for the purpose of this thesis.

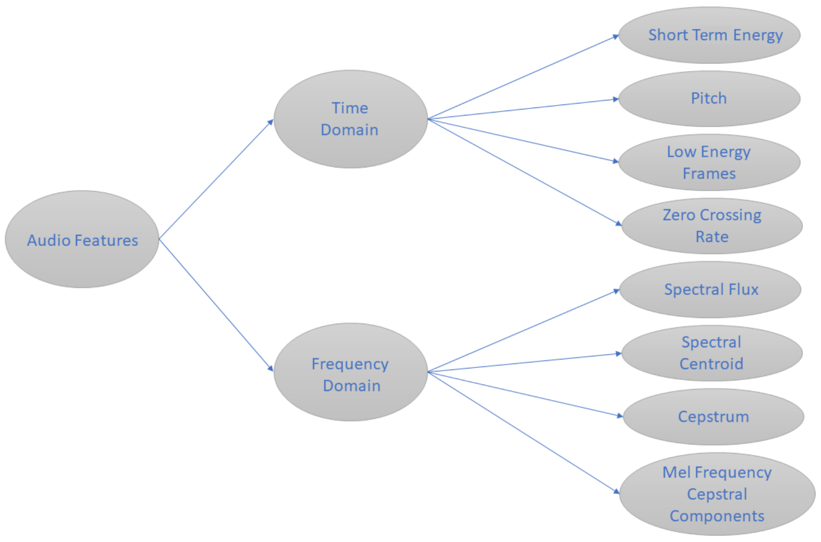


Figure 2.12: Audio Features - Time & Frequency domain

To begin with, framing is commonly used in most audio analysis and processing methods. For short-term processing, when extracting features, the audio signal is separated into short-time windows/frames than may or may not overlap, and several features' computations are performed for each frame. Continuing, we end up with a sequence, F, of feature vectors for each audio signal we process, and the vector's dimensions depend on the nature of the adopted features. Usually, we combine several features that have been extracted. The increased dimensionality is preferable to having 1D feature vectors. By the same logic, we can now divide the audio signal into mid-term windows/segments and then preform short-term processing for each segment. This way, after the F sequence has been computed, we use statistics to represent each mid-term segment for each short-term feature sequence, respectively. For the aforementioned procedure we have assumed that every mid-term segment has the same behavior according to the audio type. Thus, statistical computations can be performed and extract meaningful results. In section 3.3 we re-introduce short-term and mid-term features analysis through pyAudioAnalysis approach, a library which is dedicated to audio signal related issues.

It is important to say that in the *time-domain* feature extraction is based on the samples of the audio signal without further processing. Two basic time-domain features are the short-term energy and short-term zero-crossing rate. Combined together with specific frequency-domain features, we can simply analyze audio signals.

Nevertheless, our focus remains on *frequency-domain* features. As mentioned before, to represent frequency distribution of an audio signal we use DFT. Based on the DFT we can easily extract spectral(as they are called) features.

Some of them are the Spectral Centroid, the Spectral Flux and the Mel-Frequency Cepstrum Coefficients (MFCCs). [9]

The **spectral centroid** is a metric used to characterize the frequency spectrum in digital signal processing. It indicates where the center of "gravity" of the frequency spectrum is located. It is widely used in digital audio and music signal processing, as a measure of the timbre of music. Mathematically is defined by equation 2.2:

$$C_i = \frac{\sum_{k=1}^{Wf_L} kX_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)} \tag{2.2}$$

where $X_i(k)$ represents the magnitude of the Fourier transform of the i-th frame at the frequency group.

To continue with, the **spectral flux** measures the rate of change of the signal spectrum. To do this, we compare the spectrum of the current frame with the spectrum of the previous short-term window. Assuming the spectrum is normalized, the spectrum flux does not depend on the phase. Thus, only the amplitudes can be compared. Calculating the spectral flux of an audio signal means to determine its timbre. The mathematical definition is shown in equation 2.3.

$EN_i(k)$ represents the k-th normalized DFT coefficient at the i-th frame and is calculated by equation 2.4.

$$F_{(i,i-1)} = \sum_{k=1}^{Wf_L} (EN_i(k) - EN_{i-1}(k))^2 \tag{2.3}$$

$$EN_i(k) = \frac{X_i(k)}{\sum_{l=1}^{Wf_L} X_i(l)} \tag{2.4}$$

Last but not least, the **Mel-Frequency Cepstrum Coefficients** (MFCCs), features of audio signals that have application in audio analysis due to their capability in classification tasks. Understanding MFCCs means, we must first get acquainted with spectrograms. As we mentioned previously, the Fast Fourier transform analyzes a signal into its frequency content, the spectrum. But in music and speech, for example, the signal's frequency content varies over time. These signals are called **non periodic signals**. The key for representing these signals is to compute several spectrums by performing FFT on several windowed segments of the signal. This procedure is known as short-time Fourier transform. The FFT is computed on overlapping windowed segments of the signal. The spectrogram is the result of this computation and the visualization of the amplitude of the signal, as it varies over time at different frequencies. It is proved that humans can perceive a very small and concentrated range of frequencies and amplitudes and can detect differences in lower frequencies more efficiently than in higher ones. Therefore, the mel-scale is proposed in order to help the listener to understand how the equidistant pitch sounds. When a spectrogram has the frequencies converted to the mel-scale, it is called the **Mel spectrogram**. Figure 2.13 and figure show the plot of mel versus the Hz scale and the visualization of a mel-spectrogram, respectively.

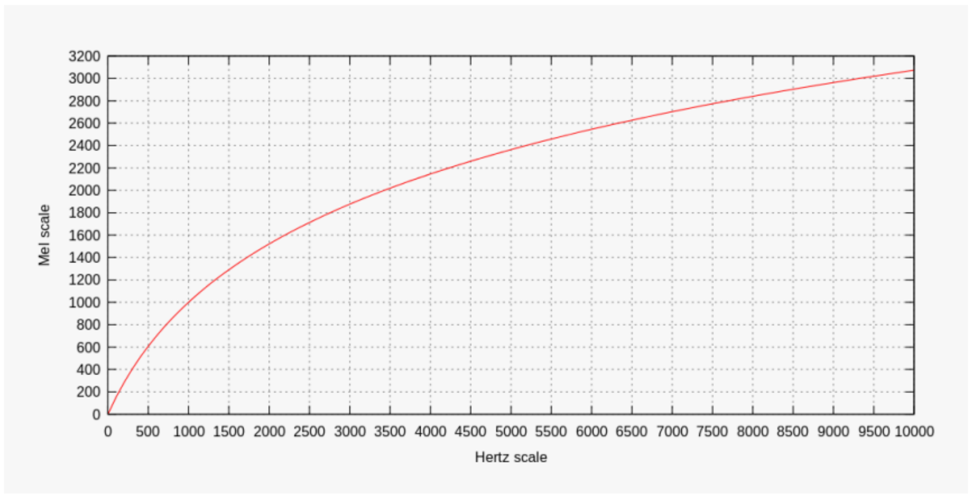


Figure 2.13: Mel scale vs. Hz scale

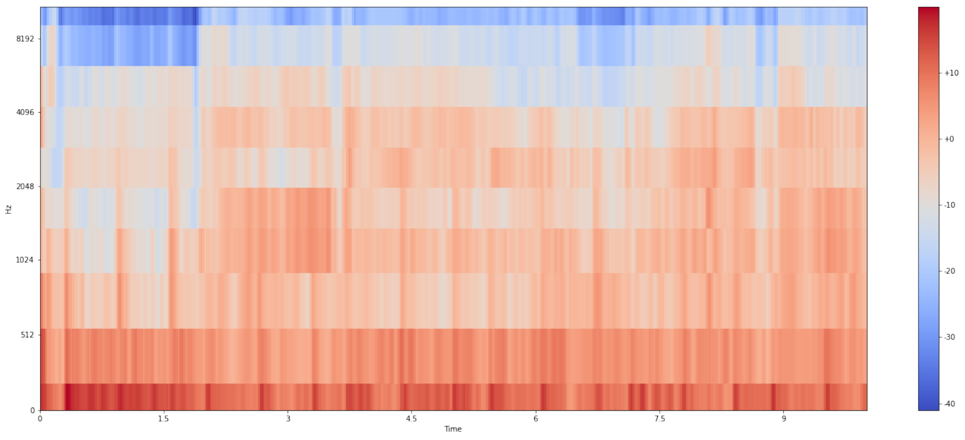


Figure 2.14: Mel Spectrogram of a Greek song, visualization over time using python - librosa library

To conclude, MFCCs could be computed after performing cepstral analysis on a mel-spectrogram. The procedure we follow is:

1. Computation of DFT.
2. A mel-scale filter bank receives the resulting spectrum as input. The filter bank consists of L filters with overlapping triangular frequency responses.
3. The equation 2.5 computes the MFCCs as the discrete cosine transform coefficients of the mel-scaled log-power spectrum [9]

$$c_m = \sum_{k=1}^{L} (log(\tilde{O_k}))cos[m(k-\frac{1}{2})\frac{\pi}{L}], m = 1, ..., L \qquad (2.5)$$

where $\tilde{O_k}$, k= 1,...,L corresponds to the power at the output of the k-th filter.

Following this procedure in a music genre classification task, for example, a song can be represented as a sequence of cepstral vectors and afterwards these cepstral vectors are fed to the classifiers to distinguish the genre of a song.

2.3 Music Information Retrieval

It is quite common, when we mumble a song and do not really know its lyrics, its title or its artist making us wonder how we could actually find this song. Another frequent desire is to discover songs that resemble the ones we already like listening to. Both are typical examples/questions that Music Information Retrieval (MIR) has answers to. To be more specific, MIR as a scientific field takes advantage the audio metadata and audio content information concerning various fields of science such as musicology, data science, signal analysis. It also covers a wide range of tasks[10] [11]. Some of them are:

- *Acoustic fingerprint.* A song is represented as small audio signal summary of its main attributes in digital form. It is usually used to recognize a sample of a song or identify similarities in a music database.[12] [13]
- *Audio thumbnail.* A music recording is being segmented by its most representative pieces while getting rid of the most redundant ones. [14][15]
- *Tempo Estimation.* Tempo is usually measured in beats per minutes. The estimation of it is either a regression or a classification task. [16] [17] [18]
- *Artist recognition.* The recognition of artists, when listening to a song, can be done by analyzing common characteristics and thus, audio features, that they use when composing music. [19]
- *Cover song detection.* Refers to identifying an alternative version of a music track, regardless the difference in each musical characteristic. [20]
- *Beat tracking.* There are instances in a song's duration when one taps their foot. Recovering a sequence of these time instances from a musical input is the main task of beat tracking. [21] It can be considered as a subtask of Query By Rhythm.
- *Query by Humming* A QbH system allows the user to find a song by humming or singing part of the tune.[22][23]

Additionally, there are MIR applications such as query by Example(QBE), query by Rhythm(QBR), music similarity and music genre classification that we choose to focus on and analyze them further in the following subsections.

2.3.1 Music Genre Classification

Music Classification is considered as a significant portion of Music Information Retrieval. It is quite useful to perform the task of classifying an unknown music track to specific classes and group all tracks, accordingly. As regards, the genre classification, we can analyze the audio content and classify any track to a set of predefined musical genres (classes). Despite the inherent difficulties in providing a rigorous definition of the term 'music genre', the computational tools we have proven that it is possible to derive solutions that exhibit acceptable performance in the context of genres of music. [9]

Labeling music into genres is a truly useful procedure when it comes to classifying and categorizing songs, albums or even artists into groups. Music genre labels are based on similar musical characteristics such as musical techniques, musical instruments, the rhythm, the cultural background or the year a song is created. The broadly known ones are Pop, Rock, Classical music, Hip-Hop, Latin, Punk. The list goes on as it is complicated enough to tell if a song or an artist belongs to one or another category.[24] A human can hear a song and easily understand in which category a music item could be classified. However, it is unclear what the features are in the music that allow us such a quick recognition, and, for this reason, it is difficult to build a phenomenological model that imitates the human ability to distinguish between music genres. In the past, the categorization was done manually but now, automatic genre classification is developing rapidly through the use of Neural Networks. Most works related to genres classification use handcrafted audio features and Convolutional NN as classifiers.

In section 3.1.2 we further explain the procedure followed in this thesis in order to perform genre classification using clustering and CNNs. All in all, each genre category is constructed by multiple others using Spotify labeling, in our try to represent the complexity of each song's genre.

2.3.2 Music Similarity

In MIR **similarity** is an interesting topic of research as it is essential in many applications regarding music retrieval and music recommendation. Detecting whether two music pieces are somehow similar can be tricky. As music has many dimensions such as rhythm, harmony, instrumentation, timbre, lyrics, genre, style, mood, music similarity is a subjective, context dependent and ill-defined research problem. Thus, the nature of the problem defines the way similarity is measured.

A simple approach and the one followed in this thesis, places feature values into a vector and then calculates the Euclidean distance between points. Euclidean distance can be defined as the distance between two n-dimensional vectors x and y or the magnitude of x - y.

$$|x - y| = \sqrt{\sum_{i=1}^{n}(x_i + y_i)^2} \tag{2.6}$$

Formula 2.6 implies the independence and the scale of the features. Most importantly, a Euclidean metric assumes that the features are (nearly) orthogonal so the distance along different axis can be summed. A Euclidean metric also assumes that each feature is equally important. The greater the distance, the lower the similarity between the two objects.

2.3.3 Query by Example

Retrieving a musical track from a large music collection can be tricky. A lot of research is done to enable us to query for a song or a relevant one using another song, or part of it as an example. This can be considered as a music similarity task and helps users to keep track on the music they like. As listening to music can be a quite subjective process and usually mood-based, using known songs as an example to find similar ones, makes this process a lot easier. Apart from this, QbE task is used to analyze a composer's evolution, resolve copyright issues and help the Query By Humming task we mentioned above. To perform a query by example, one must first extract the appropriate audio features from a large music dataset. These features help us to find a way of comparing music tracks and calculate similarities.[25]

2.3.4 Query by Rhythm

Another technique called Query by Rhythm, supports queries regarding similarities on the rhythm of songs[26]. Songs are being modeled as a string representation of rhythm patterns and notes. When one queries a database, the expecting result contains similar music patterns to the queried song and also shows where those patterns appear. The task is basically string-based and agrees with tempo estimation and beats tracking. QbR task can be used in music mixing and music composition.

Chapter 3

Proposed Methodology

This chapter addresses the different methods, libraries and modules used throughout the experiments with our main goal being to analyze audio signals, extract features, train CNN models and finally find similarities between songs. The programming language we used is Python 3.9.4, a high-level programming language with a great appliance in data science due to the plethora of libraries and tools it provides.

The procedure followed to label and create the dataset to train the CNN models has been already done by members of the MagCILab and it is not part of this thesis. The dataset is labeled according to the Spotify API, a RESTful API which provides JSON metadata about songs regarding music features such as danceability, valence, energy and more.[27]

Although music is usually classified into genres, it is in our interest to take into consideration the musical mood, the emotional responses that music evokes to listeners and understand why one could listen to a song and not to another. Emotion is quite subjective, while, on the other hand, genre is almost "a general truth" even though it is complex enough, as we discussed in subsection 2.3.1. Thereafter, we explain more about musical moods, the models we trained and the methodology we followed in order to extract musical features for our experiments.

3.1 Convolutional Neural Networks Audio Representation

The first step of a supervised representation of music is to create the appropriate CNN models to extract features about the danceability, valence, energy and genre of a song. For all the aforementioned models, **deep audio features** library is used. Deep audio features is a Python library using **Pytorch** wrappers in performing audio classification tasks or extracting features with the help of Convolutional Neural Networks training.[28]

3.1.1 Musical Mood Models

Our models regarding human emotions have been selected after studying thoroughly the 2-D models of emotion suggested by James A. Russell[29] and by Robert E. Thayer[30], afterwards. As Figures 3.1 and 3.2 indicate, emotions are distributed in a two-dimensional space where x-axis follows a person's valence and y-axis the energy. These models are widely used in order to classify songs emotionally. These two parameters are also used by Spotify API that we used. Thus, we now focus on how those moods and emotions can be described in terms of songs and music in general.

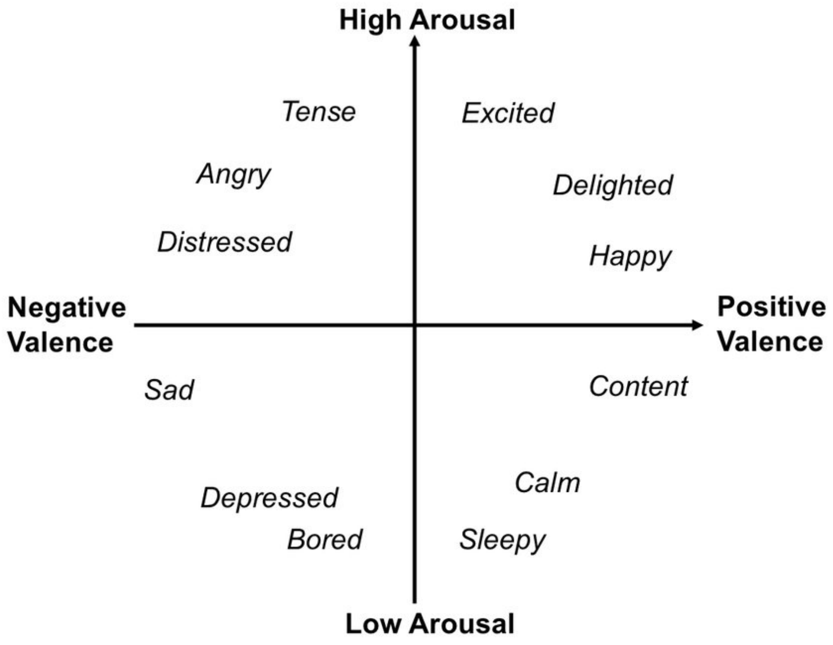


Figure 3.1: Human emotional categorization model by Russel(1980)

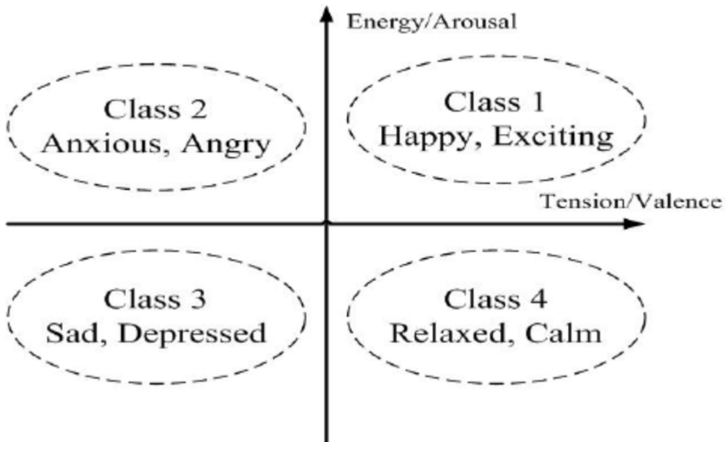


Figure 3.2: Human emotional categorization model by Thayer(1990)

Beginning with **arousal** or else **energy**, it depicts the physiological and psychological state of being reactive to stimuli. In other words, it is a perceptible measure of intensity and activity. A typical energetic song feels noisy, loud, and possibly fast. Next, **valence**, is the intrinsic attractiveness or adverseness of any emotion and depicts the musical positivity conveyed by a musical piece. All in all, valence and energy are strong indicators for the acoustic mood and the overall emotional quality of a song. [31] Last but not least, we chose to study the **danceability** of a song. Danceability is a key characteristic for choosing music to listen to. Danceability describes how suitable a track is for dancing, based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.

The CNN models regarding danceability, valence and energy, have been already created by training and validating the Convolutional Neural Network that the deep audio features library provides. The train-data are put into three folders named 'low', 'medium' and 'high'. The names indicated the intensity of the audio characteristic we studied each time and referred to the corresponding annotated labels. The model is saved as a 'pt' file. The validation statistics of these three models can been seen in the following table 3.1

CNN model	accuracy	f1 score	loss
danceability	72%	69%	4%
energy	71%	71%	4%
valence	62%	63%	5%

Table 3.1: Validation Results regarding danceability, energy and valence CNN models

3.1.2 Musical Genres Model

3.1.2.1 Data processing - KMeans Clustering

As part of this thesis goal, we have built a fourth model regarding the genre of a song, using the same tools and the same logic as the other ones. The difference lies in the dataset's processing procedure that we used for each feature of the music signal that we wanted to study. The goal is to create a model, based on a robust dataset that could achieve generalization for other music datasets as well, having, at the same time, the lowest variance possible. Thus, the dataset is preprocessed, so it can fit the dependencies of the experiment.

Initially, the dataset is comprised of 15,947 songs. The metadata of each song is extracted through the Spotify API and saved in 'csv' form, accordingly. For the genre of a song, in particular, a multi-label list is extracted and consisted of all the known -as far as the Spotify is concerned- genres that the song may belong to. For some songs the list is empty and so is the corresponding column in the 'csv' file. By discarding those songs, 10,673 songs remain. We also decided to discard all Greek songs because we want to perform a small MIR experiment on a Greek database that it is further described in the 2.1.3.2 subsection, and we want to avoid any bias interference. Eventually, the dataset ends up to 10,633 songs.

Furthermore, k-means **clustering** (see 2.1.2.1) is performed, for the genres to be decided and for all songs to be grouped, accordingly. The number of different lists is counted to be 2,557. To continue, we extract different **unique** genres from each genre list and count their occurrence frequency inside those genre lists. Finally, 1,410 genres are resulted and from those we choose the most 'popular', namely, those with occurrence frequency greater than 10 songs. This procedure ends up with 500 different unique genres. Figure 3.3 indicates a wordcloud generated by a small python script using the wordcloud library. Wordclouds visualize the frequency a word is introduced to a text. In this case, we provide a python dictionary we have made of genres and its occurrence frequency.

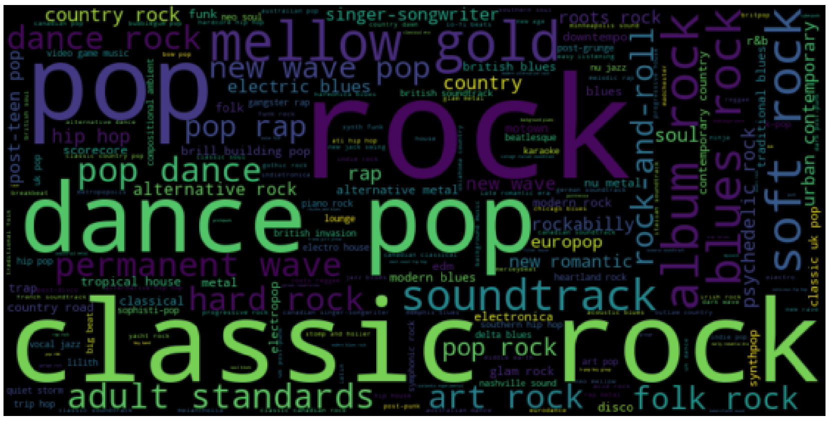
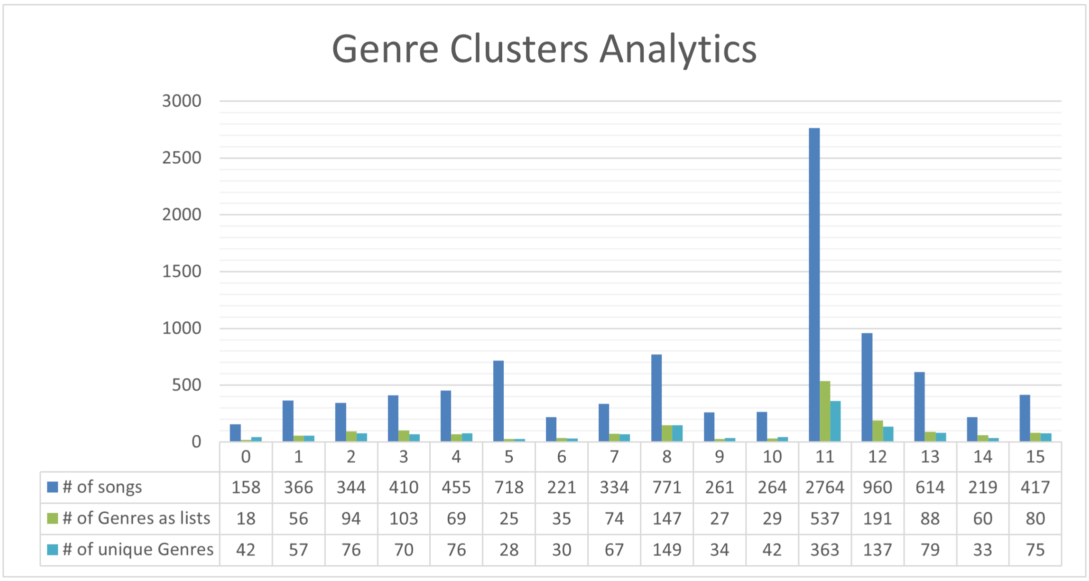


Figure 3.3: Wordcloud showing the most "popular" genres among the provided data.

To conclude, the Boolean table that is used for clustering is **10,663 x 500** having 1 to the corresponding genres positions that belong to the genre list of each song and 0 on those that do not belong to the respective list. After clustering is performed, we end up with 15 classes which are further processed. More specifically, one of the classes contains 673 lists of genres and 343 unique genres, regarding 3,953 songs. Due to the fact that this class seems to gather uncommon and rare genres-lists, we decide to perform again K-Means clustering producing three more classes. The associated classes are merged with others, in order to have a more reasonable result.

In consequence, the final result consists of **16 genres-classes** and **9,276 songs**. Each genre-class is a combination of multi-label lists. As mentioned before, these multi-label lists are a combination of unique, known genres. All the analytics about the classes are presented in Figure 3.4.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
🟦 # of songs	158	366	344	410	455	718	221	334	771	261	264	2764	960	614	219	417
🟩 # of Genres as lists	18	56	94	103	69	25	35	74	147	27	29	537	191	88	60	80
🟦 # of unique Genres	42	57	76	70	76	28	30	67	149	34	42	363	137	79	33	75

Figure 3.4: Analytics for each Genre Cluster.

The following table 3.2 indicates the most common genres that combined with each other -or with others- create the corresponding class.

Class ID	Top Unique Genres
0	electronica, trip hop, big beat, nu jazz
1	mellow gold, soft rock, adult standards, folk rock
2	dance rock, new wave, new romantic, new wave pop, synthpop
3	rap, hip hop, pop rap, trap, southern hip hop
4	rock, hard rock, album rock, classic rock, metal
5	scorecore, soundtrack, video game music british soundtrack
6	classical, late romantic era, post-romantic era, baroque
7	r&b, urban contemporary, dance pop, hip pop,neo soul, new jack swing, pop, quiet storm, pop rap
8	rock, alternative rock, modern rock, permanent wave, pop rock
9	contemporary country, road country, country, modern country rock
10	blues, electric blues, blues rock, traditional blues, classic rock, modern blues
11	soul, adult standards, motown, brill building pop, funk
12	dance pop, pop, pop dance, post-teen pop, tropical house
13	mellow gold, classic rock, rock, soft rock, album rock
14	dance pop, pop, pop dance, post-teen pop, tropical house
15	adult standards, rock-and-roll, brill building pop, rockabilly, lounge, merseybeat ,vocal jazz

Table 3.2: Most frequent unique Genres in each Class

3.1.2.2 Training Genres model

The last step to take, is to train the new model. Deep audio features library provides a CNN model that consists of four 2D convolutional layers using as activation function *Leaky ReLu* and *Max Pooling.* The input of the CNN is the Mel Spectrograms (2.2.2) extracted from the associated 'wav' files of our song dataset. The final three layers are Linear for the output to be predicted to the correct class. Figure 3.5 is an example of the CNN architecture used for each model. Beginning with a sequential pipeline of Convolutional Layer - LeakyReLu - Max Pooling, we notice the duplication of the output features. Afterwards, we apply Dropout - Linear Layer and LeakyReLu before the last Linear Layer with an output of 16 x 256, where the decision is made.

We should point out the use of BatchNorm2D and Dropout. Both are classes of torch.nn package of PyTorch library, and they serve different but important purposes for the CNN. Batch Normalization (BatchNorm2D) tends to minimize the internal covariate shift, a phenomenon that makes training nonlinear models extremely hard as the changing of the distribution throughout the network's layers decreases the learning rate and makes it difficult for us to initialize the parameters carefully. With BatchNorm2d, normalization is now enclosed in model's architecture and is performed for each training mini-batch.[32][33]

On the other hand, Dropout is a great method to reduceg overfitting. While the model is being trained, some of the input elements, given the Bernoulli's distribution probability, are randomly zeroed. Hence, co-adaptations are avoided and neurons learn to produce the correct answer due to the multiple combinations and the randomness of this procedure.[32] [34]

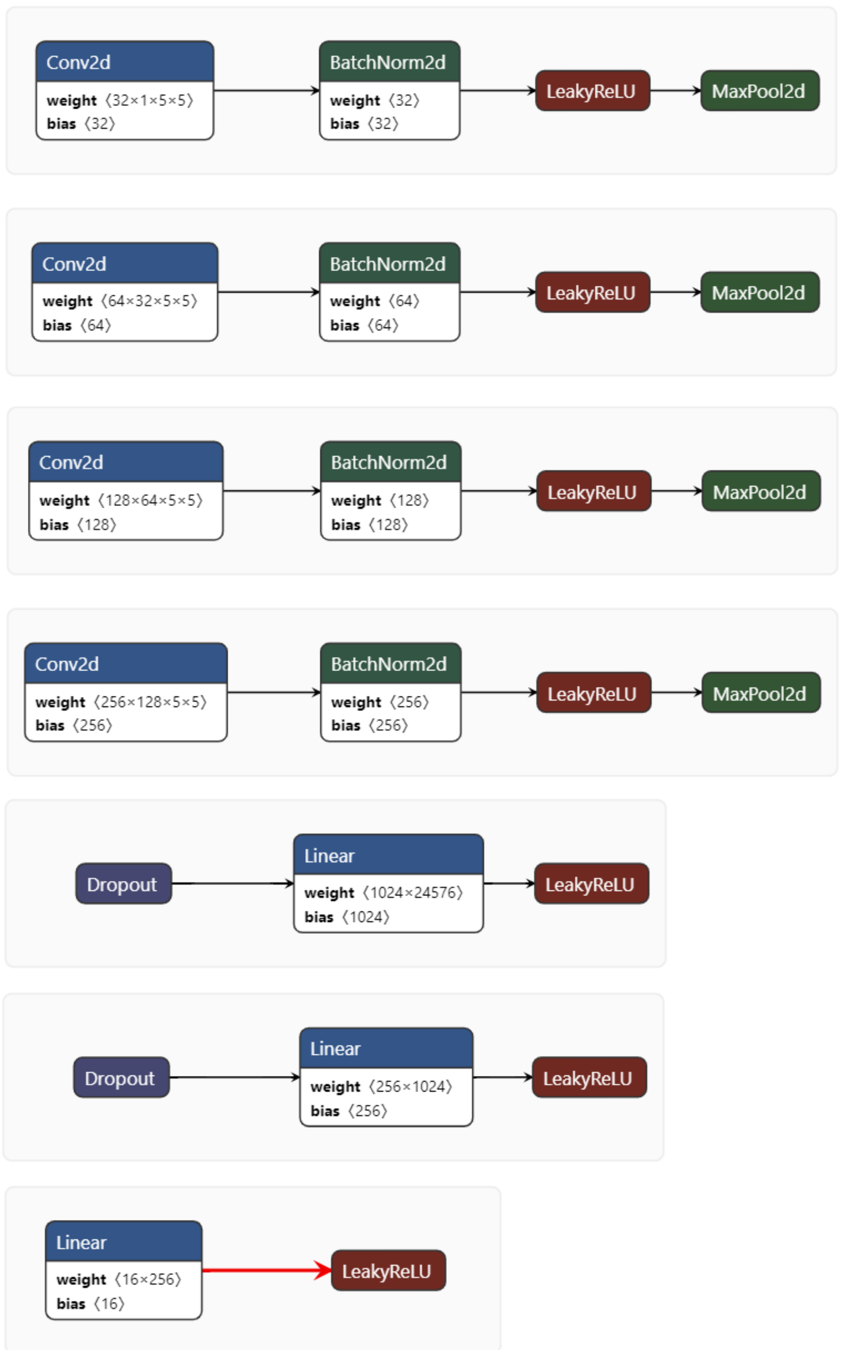


Figure 3.5: Deep Audio Features - Convolutional Neural Network Architecture

The validation statistical results are shown in the table 3.3 below. The percentages are indeed low, but this is an acceptable fact. Genre classification is a multi-label task in contrast with the previous CNN models which are three-class classification tasks. This thesis purpose is not the optimization of the CNN models, but to use the posteriors and draw conclusions about music similarity.

CNN model	accuracy	f1 score	loss
genres	40%	20%	10%

Table 3.3: Validation Results regarding genres CNN model

3.2 Data Visualization

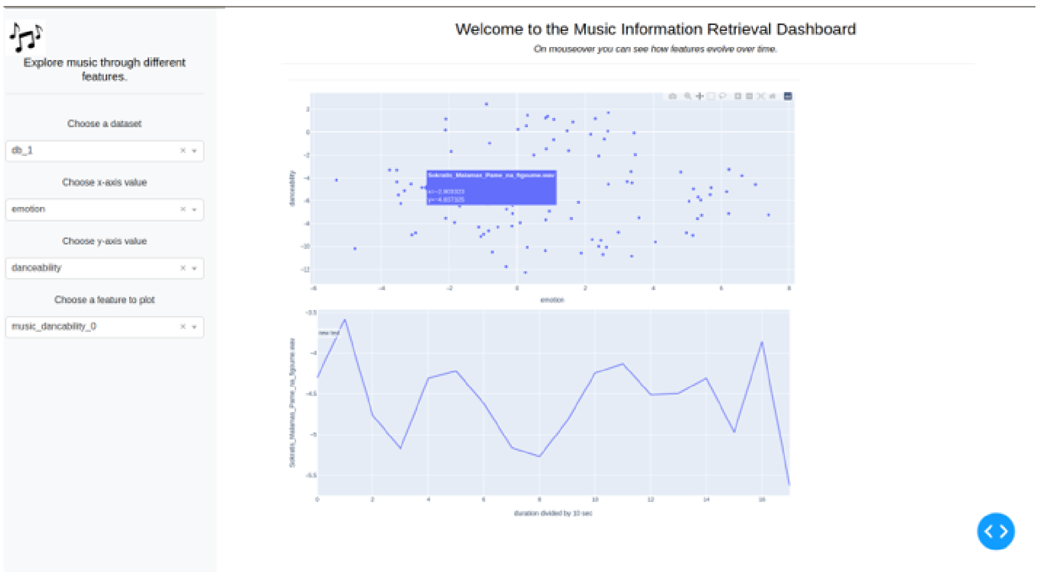
Nowadays, the numerous data available insert complexity to its processing and understanding. In addition, human brain can easier process visual data than dense written text. Therefore, data visualization plays a critical role to Data Analysis. The charts, the shapes, the colors, everything matters when it comes to display complex data and make it more approachable and easier to view. Visualizing data gives insights into it while making analysis more interesting. The conclusions we once drew from our processed data can now be proven or declined. We may be given the opportunity to draw new inferences and correct any errors that have occurred during our analysis.

Following our models' training, we aimd to visualize how songs of an unlabeled dataset could relate to each other. This way, we can qualitatively evaluate the performance of the CNN models(see 4.3.1), alongside with displaying and inferring the similarity between the songs of the under-study dataset.

That being the case, we choose 100 (one hundred) Greek songs of different styles and genres. Again, deep audio features library helps us, this time with the representation of the dataset. We build the dataset, using the appropriate method from the library and the set of the predefined CNN models. Finally, each song is expressed by each model as a combination of meta-features (a list of predictions) stored in a pickle file.

For the visualization part we create a simple application using Dash framework and Python 3.9.4. Plotly's Dash is an open-source Python library with a plethora of toolesets for the creation of reactive web applications that run on top of Flask and communicate JSON packets over HTTP requests. In accordance with the front-end, React.js is used, a Facebook's user-interface written in Javascript. All in all, Dash apps are quite handy when it comes to data analysis, modeling or visualization and is being used widely by the data science community. [35]

To begin with, we load the "pickled" dataset and create a data frame using pandas DataFrame method, with all the features' temporal and the songs' names. It is important to scale the input data and use a method to reduce their dimensions. In this regard, we experiment with PCA, TruncatedSVD and t-SNE, by sklearn library (see Chapter 2.1.2.2). We achieve the best results when using the t-SNE algorithm and the full length songs. We also use another dataset consisted of 196(one hundred ninety-six) songs of pop and classical music only. The overview of our app is presented in the Figures below.

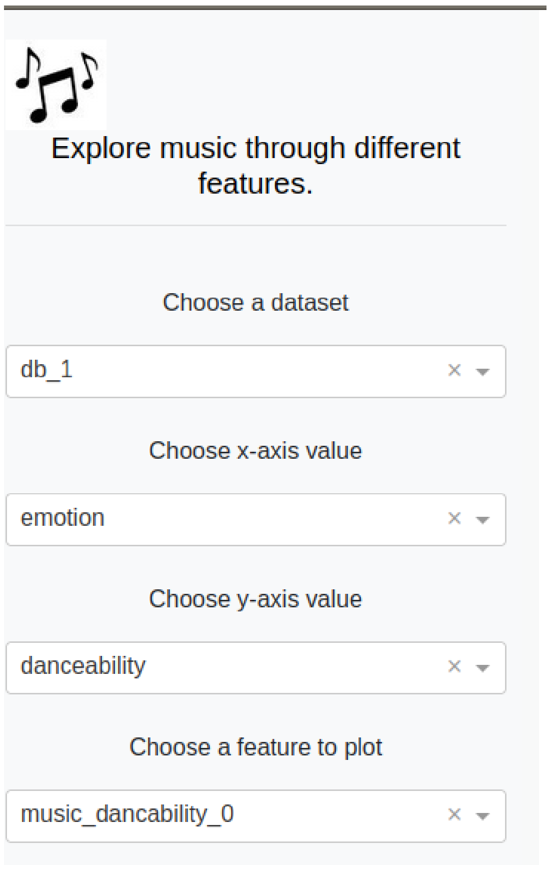


Explore music through different features.
Choose a dataset
db_1
Choose x-axis value
emotion
Choose y-axis value
danceability
Choose a feature to plot
music_dancability_0
Welcome to the Music Information Retrieval Dashboard
On mouseover you can see how features evolve over time.
Sokratis_Malamas_Pame_na_figoume.wav
danceability
emotion
duration divided by 10 sec

Figure 3.6: Overview of our dash application

To be more explanatory, the dropdown lists shown in figure 3.7 have the following options:

- Dataset choice:
  - First database: A folder with equally genre-distributed Greek songs(see Figures 4.3 and 4.4).
  - Second database: A folder with classical and pop songs only(see Figures 4.5 and 4.6).
- Axes choice:
  - Principal component 1: One of the components t-SNE created.
  - Principal component 2: One of the components t-SNE created.
  - Emotion: Energy and Valence combined.
  - Genre: All sixteen genres combined(see 3.1.2.1).
  - Danceability: Danceability model.
- Features choice:
  - Music danceability: Choices between high(index 0), medium(index 1) and low(index 2).
  - Music valence: Choices between high(index 0), medium(index 1) and low(index 2).
  - Music energy: Choices between high(index 0), medium(index 1) and low(index 2).
  - Genres: Choices between the sixteen genres.


Explore music through different features.

Choose a dataset

db_1

Choose x-axis value

emotion

Choose y-axis value

danceability

Choose a feature to plot

music_dancability_0

Figure 3.7: Left part of the app with dropdown lists to choose between features to display.

3.3 Handcrafted Audio Representation

Besides CNN, the audio representation using handcrafted features plays an important role to our experiments. On that account, we utilize the pyAudioAnalysis library.

PyAudioAnalysis is an open-source Python library with a plethora of audio analysis procedures, as it is able to extract features, visualize data, perform classification on audio signals and supervised and unsupervised segmentation. This library has also several research applications regarding audio events detection, speech recognition and more. [36].

Table 3.4 shows the complete list of the features that pyAudioAnalysis can extract and further analyze.

Audio Feature	Description
Zero Crossing Rate	Sign-changes' rate of the signal among a particular frame's duration.
Energy	Signal values' sum of squares, normalized by the respective frame length.
Entropy of Energy	Measure of abrupt changes. Entropy of sub-frames' normalized energies.
Spectral Centroid	Spectrum's gravity center.
Spectral Spread	Spectrum's second central moment.
Spectral Entropy	Normalized spectral energies' entropy for a set of sub-frames.
Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
Spectral Rollof	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
Chroma Deviation	The standard deviation of the 12 chroma coefficients.

Table 3.4: Implemented audio features

This thesis focuses on short and mid-term analysis of the features are shown above. We set the values equal to 0.1s for the sort-term and 10s for the mid-term features.

As regards the short-term features, all thirty-four (34) are calculated in each short frame the audio signal has been divided into. The usual values for the short-term window sizes are between 20ms and 100ms. The library provides the ability to use an overlapping procedure; thus the window step is smaller than the window length or a non-overlapping in which the window step is equal its length.

To continue, there is another approach in audio analysis that concerns the mid-term features of an audio signal as we have already mentioned in subsection 2.2.2, in which we divide the audio signal into segments(mid-term windows), which also, as regards the pyAudioAnalysis library, can be either overlapping or non-overlapping. The segment is processed regarding the short-terms and its features are being statistically handled forming a set. The mid-term segment can have a size of 1 to 10 seconds. It is important to note that if the duration of a recording is relatively long, it is proposed to average the mid-term features so that we can have a representation of mid-term statistics as an average vector.

Chapter 4

Experimental Results

This chapter elaborates the core stages of evaluating our experimental procedure. In MIR problems, it is important to have physical (real) users' interaction. This way, one can evaluate the results a computer provides in comparison with those of a person's hearing experience. For this reason, the MagCIL team has built a dataset based on users' decisions. Based on this dataset, we evaluate the results we get from the decision layer and the fully connected layers of our CNN models, taking into consideration the euclidean distance between feature vectors. The dataset, the experimental setup and the final results, are presented as follows.

4.1 Dataset

The dataset used in the evaluation procedure is a subset of the Music Content Similarity Dataset, which has been generated by the MagCI-Lab of the Institute of Informatics and Telecommunications, of the National Center for Scientific Research "Demokritos". The initial dataset consists of 2,540 triplets which are randomly created by 4,260 unique songs. The main idea is to let the users decide which song of a specific triplet is, according to them, the least "relevant". The relevance between songs relies on the users' essence. The script that implements the aforementioned concept, simply provides a user with random 10sec of each song of a triplet to listen to. After all three songs are played, one can listen again or just decide which of them is dissimilar to the other two. Each triplet and the users' decision ID is saved into an excel file with four columns. The first three are about the songs the user listened to (the songs the triplet consisted of), and the fourth one indicates the song the user has annotated as dissimilar/irrelevant.

Finally, after the annotations takes place, the results are processed, and we end up with what we call **ground truth** csv file and our similarity dataset. The agreement between the users is approximately 80%.

4.2 Experimental Setup

Now we try to explain further how we utilize users' experience to infer the accuracy of our CNN models. We use a subset of 2,252 songs grouped in 874 triplets. All songs are up to 10 seconds long, in a mp3 format, and we transform them to mono-canal wav files. The polestar for the evaluation is the "ground truth" csv file.

Following, the CNN models(valence, energy, danceability, genre) and the pyAudioAnalysis library are used for features extraction. For each triplet of the CSV file, we perform ten different features comparison. The four CNN models are concatenated into one vector. We choose features extracted by the last - **decision layer**, by the penultimate fully connected layer and by the third from the last fully connected layer. The goal is to examine the behavior of the model before the decision is made. For each song of a triplet is extracted:

- Layer’s 0 features - Decision Layer
- Layer’s -1 features - Penultimate Fully Connected Layer
- Layer’s -2 features - Antepenultimate Fully Connected Layer
- Features from pyAudioAnalysis
- Layer’s 0 combined with layer’s -1 features
- Layer’s 0 combined with layer’s -2 features
- Layer’s 0 combined with pyAudioAnalysis features
- Layer’s -1 combined with layer’s -2 features
- Layer’s -1 combined with pyAudioAnalysis features
- Layer’s -2 combined with pyAudioAnalysis features

Noticing that for layers -1 and -2, because of the size of the extracted features array, we perform PCA and normalization on the resulting array. The procedure is based on the *euclidean distance* between the feature vectors of a pair of songs in each triplet. When the distance is the greatest, the song under examination is the least relevant. If the song is also marked as irrelevant in the CSV file, then we had a success.

4.3 Final Results

Following the experimental setup, we present the final results of our evaluation procedure, and we discuss them regarding known facts and our expectations.

Table 4.1 and then Figure 4.1 indicate the accuracy we achieved when we extracted features using four different approaches.The result from the features that are extracted by the antepenultimate fully connected layer achieves the best accuracy, approximately 47%. We also mention the small differences when it comes to using the PCA method to reduce the vectors’ dimensionality.

Features Extraction	PCA & Standard Scaling	Without PCA & Standard Scaling
Layer 0	36.18%	-
Layer -1	35.43%	35.93%
Layer -2	45.98%	46.48%
pyAudioAnalysis	44.47%	-

Table 4.1: CNN models & pyAudioAnalysis success when comparing the euclidean distance between feature vectors of a pair of songs in each triplet.

To continue with, the Table 4.2 and Figure 4.2 summarizes the accuracy results when we combined different features extraction procedures. As we described previously, for each triplet we performed pair comparison using the euclidean distance between the feature vectors.

	pyAudioAnalysis	Layer 0	Layer -1	Layer -2
Layer 0	43.47%	-	37.19%	46.48%
Layer -1	37.44%	37.19%	-	45.73%
Layer -2	46.73%	46.73%	45.73%	-
pyAudioAnalysis	-	43.47%	37.44%	46.73%

Table 4.2: Combined features of CNN models and pyAudioAnalysis. PCA and Standard Scaling are applied.

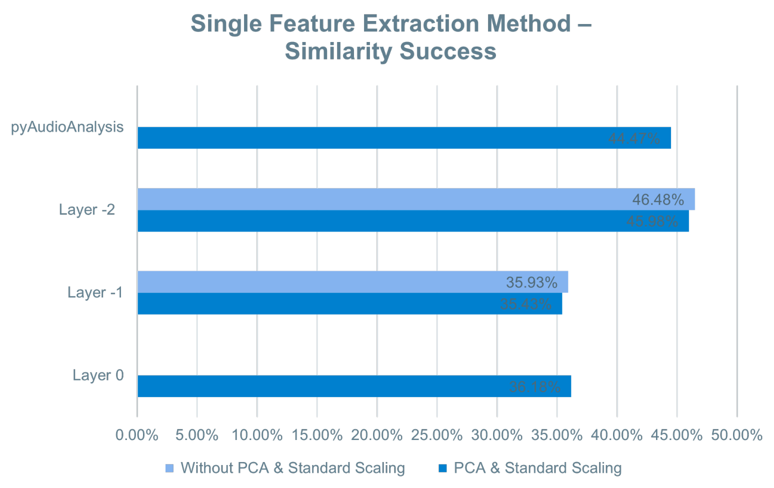


Figure 4.1: Bar chart indicating similarity success when using one of the methods for feature extraction.

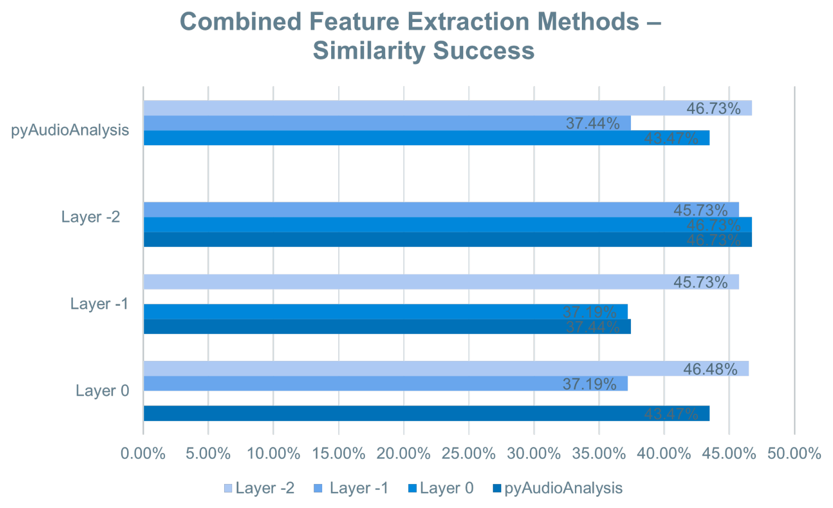


Figure 4.2: Bar chart indicating similarity success when using combined methods for feature extraction.

The best results are attained when combining pyAudioAnalysis features with features from the antepenultimate fully connected layer. This is something we expect as they are both content-based features that characterize the most the song under examination. What is odd, though, is that with PCA and normalization, success percentages are quite lower. Nevertheless, we are confident that our models are very close to the users' decisions and that equips us with experience for future content-based music similarity problems. Henceforth, in the last chapter we will discuss our future work, the steps to improve our experiments and the conclusions made through the implementation of this thesis.

4.3.1 Qualitative evaluation

When trying to evaluate our experimental results, it is important to have something more tangible than numbers. Visualizing data helps us understand, in a simplified manner, whether we achieve our goals or not. Following the DASH application(3.2) that we created for the visualization of our CNN models, we can evaluate the conclusions that are drawn from it.

Therefore, we examine the details of what is displayed on this app. The plots shown below give great assistance to understand the provided database's features and whether we can extract valuable information from them or not.

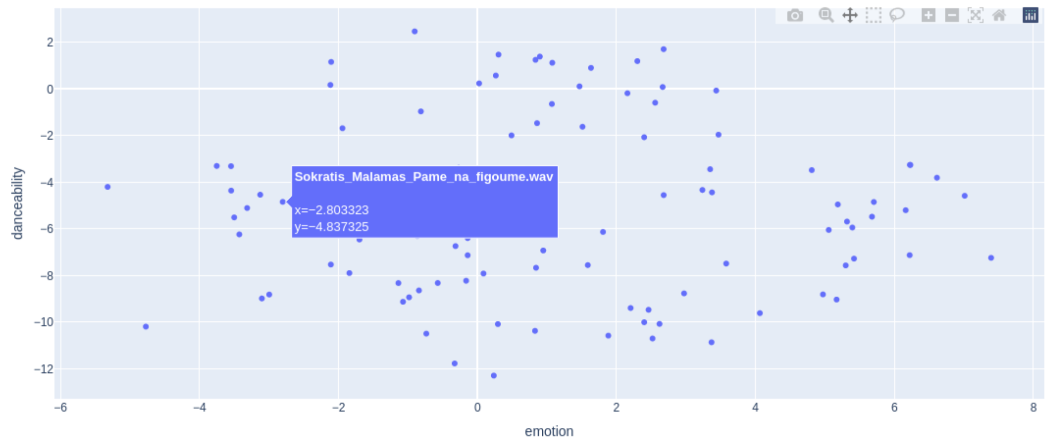


Figure 4.3: Greek songs DB: Plot of emotion - danceability for the first dataset. On hover one can see the exact coordinates (values) of each song and its name.

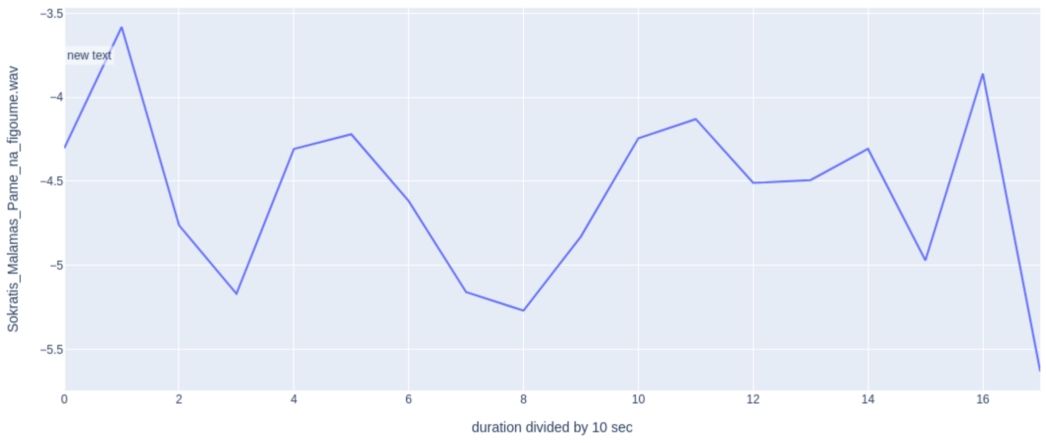


Figure 4.4: Greek songs DB: Plot of the chosen features that indicates high danceability (index 0) - duration of the song divided by 10 sec.

If we examine figure 4.5, for instance, all pop songs have high danceability and emotion(valence, energy) values, while, on the other hand classical music is low in terms of both danceability and emotion. This is, clearly, something broadly proven but it confirms that our approaches regarding feature extraction work as expected. Additionally, in figure 4.6 we can see that for a specific pop track danceability increases quite fast at its early seconds.

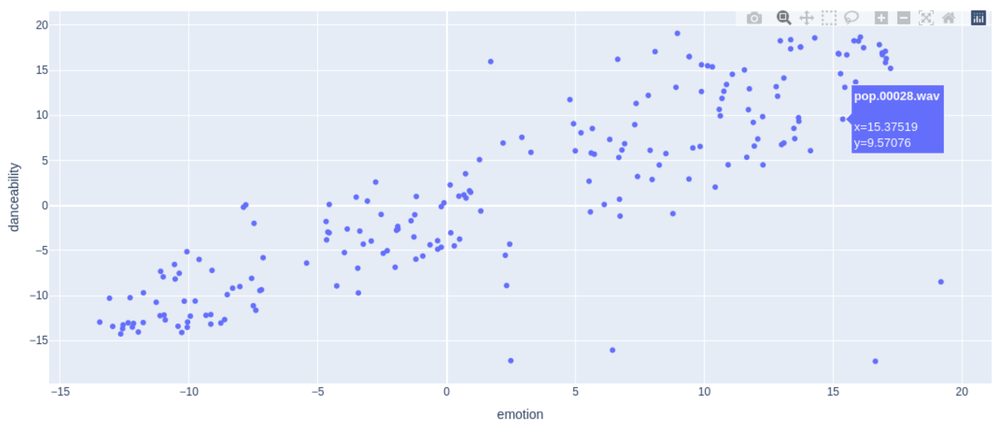


Figure 4.5: Classical and Pop songs DB: Plot of emotion - danceability for the first dataset. On hover one can see the exact coordinates (values) of each song and its name.

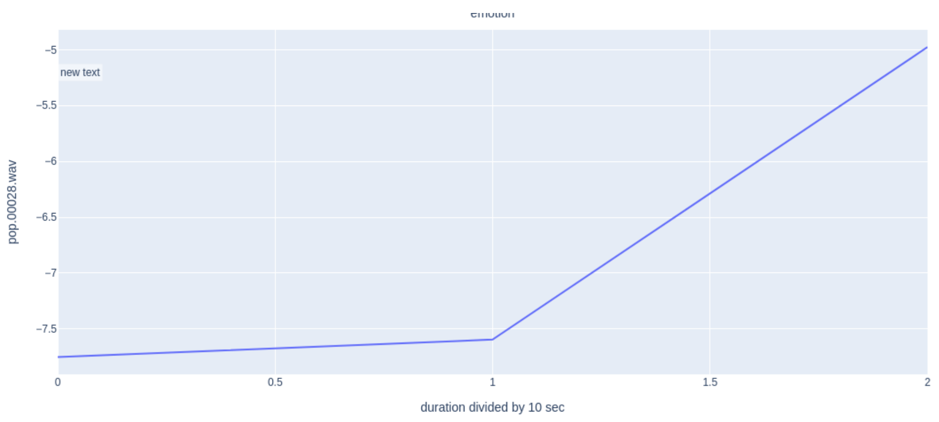


Figure 4.6: Classical and Pop songs DB: Plot of the chosen features that indicates high danceability (index 0) - duration of the song divided by 10 sec.

The way data are distributed in plots can easily guide users and help them choose a song according to their preferences or their mood. We can also make assumptions for the similarity between songs. It is expected to have songs near each other, as per the coordinates when they are similar in danceablity, emotion or any other feature under investigation. Visualizing data improves users' experience and saves us time when it comes to songs recommendation. It is also most helpful while trying to see the behavior of a feature during time.

Chapter 5

Conclusion

Concluding, Machine Learning algorithms and Convolutional Neural Networks are powerful tools in order to perform Music Information Retrieval tasks. With the rapid growth of the information technologies and the extended use of the Internet, listening to music and storing songs into a collection becomes quite different compared to the past. Music streaming platforms are now used widely by the most of us. As a result, large amount of music data is produced every day, creating a need to find ways to manage it, analyze it and extract essential information that will improve the users' experience. The plethora of techniques and algorithms machine learning and data science provide us, has proven to be the best way to achieve this.

Music has a wide spectrum of styles, genres and artists. Deciding what to listen is usually emotionally driven and can be quite frustrating when options are limited. Therefore, streaming platforms companies developed recommendation systems that provide users with songs that are similar with those they like listening to. Sorting songs by styles, mood or genre can be a lot tricky but most recommendation systems are user-centric, which means they create a personal profile that suits the users' preferences, and all the suggestions are based on the users' activity. To fulfill the needs of a user, it is important to understand the way songs resemble to each other. Thus, features that are extracted from a music track's signal can be meaningful inputs to neural networks that are trained to classify a song as per the genre or the mood. Our experiments prove that low level features are indeed a great way to represent a musical track and make assumptions about its similarity with others in a music collection, its genre or the emotion it arouses. However, there is always more to try, more to experiment and things to change.

First things first, in a machine learning task we should be confident about our dataset, the processing it goes through and the stability it will provide to our model after training. To that effect, our future work will prioritize building a dataset that will be appropriate and descriptive regarding the Genres. The dataset should be a representative sample of the music available in streaming platforms, so that we could be more accurate about the genre of a song and at the same time succeed the generalization of our results.

As far as the evaluation procedure is concerned, there are few points are to be discussed. Firstly, we could say that a more thorough evaluation procedure could involve more users. At the same time, their music profiles as per their taste in music should be created. This way, we could assume the percentage of subjectivity that lurks every time one listens to a song and decides the similarity. Another thing to consider is the use of another distance function, for example Mahalanobis or Manhattan, and see whether we could achieve a better accuracy on our models.

Furthermore, we could examine a whole new procedure regarding the identification of a song among others. A quite popular deep learning method that finds application in Information Retrieval tasks is Metric Learning [37][38] [38]. Understanding the way metric learning works is the first step into transferring this knowledge to our goal which analyzing songs in its low-level features and finding similarities. Metric learning could be a great supervised method to perform Music Information Retrieval tasks [39] as it can calculate the proximity between the representations of songs, introducing the idea of similarity in a machine learning manner.

Last but not least, another way to extract features could be proven helpful for our models to be trained. The idea is using a Convolutional Autoencoder which can learn a compressed representation of raw data. The part that concerns us is the encoder. Encoders could be used as a technique to prepare raw data for audio feature extraction[40] [41]. Hopefully, these features will hold important information regarding the songs identity and play a supporting role to our future experiments.

Bibliography

[1] J. S. Downie, "Music information retrieval," *Annual review of information science and technology*, vol. 37, no. 1, pp. 295–340, 2003.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[3] I. Jolliffe, "Principal component analysis," *Encyclopedia of statistics in behavioral science*, 2005.

[4] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[5] S. Haykin, *Neural networks and learning machines, 3/E.* Pearson Education India, 2009.

[6] S. Theodoridis, *Machine Learning - A Bayesian and Optimization Perspective*, ch. 18, pp. 902–1029. Academic Press, 2 ed., 2020.

[7] H. D. Young and R. A. Freedman, *University Physics with Modern Physics, 11th edition.* Pearson, 2004.

[8] A. I. Margaris, *Signals & Systems of Continuum & Discrete Time.* Tziola Publications, 2017.

[9] T. Giannakopoulos and A. Pikrakis, *Introduction to audio analysis: a MATLAB® approach.* Academic Press, 2014.

[10] M. Schedl, E. Gómez Gutiérrez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261.*, 2014.

[11] T. Langer, "Music information retrieval & visualization," *Trends in Information Visualization*, pp. 15–22, 2010.

[12] E. K. T. H. J. Cano, Pedro Batlle, "A review of audio fingerprinting," *The Journal of VLSI Signal Processing vol. 41 iss. 3*, vol. 41, nov 2005.

[13] P. Cano, E. Batlle, E. Gómez, L. de C. T. Gomes, and M. Bonnet, "Audio fingerprinting: Concepts and applications," in *Computational Intelligence for Modelling and Prediction*, 2005.

[14] M. Bartsch and G. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, 2005.

[15] N. Jiang and M. Müller, "Towards efficient audio thumbnailing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5192–5196, 2014.

[16] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *ISMIR*, 2018.

[17] H. Ishizaki, K. Hoashi, and Y. Takishima, "Full-automatic dj mixing system with optimal tempo adjustment based on measurement function of user discomfort," in *ISMIR*, 2009.

[18] A. Sternin, S. Stober, J. A. Grahn, and A. M. Owen, "Supplementary material for the paper "tempo estimation from the eeg signal during perception and imagination of music"," 2015.

[19] G. Widmer and P. Zanon, "Automatic recognition of famous artists by machine," in *ECAI*, 2004.

[20] J. Lee, S. Chang, S. K. Choe, and K. Lee, "Cover song identification using song-to-song cross-similarity matrix with convolutional neural network," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 396–400, 2018.

[21] A. Gkiokas and V. Katsouros, "Convolutional neural networks for real-time beat tracking: A dancing robot application.," in *ISMIR*, pp. 286–293, 2017.

[22] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: musical information retrieval in an audio database," in *MULTIMEDIA '95*, 1995.

[23] W.-H. Lai and C.-Y. Lee, "Query by singing / humming system using segment-based melody matching for music retrieval," 2016.

[24] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[25] K. Itoyama, M. Goto, K. Komatani, T. Ogata, and H. Okuno, "Query-by-example music information retrieval by score-informed source separation and remixing technologies," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 1–14, 2011.

[26] J. C. Chen and A. L. Chen, "Query by rhythm: An approach for song retrieval in music databases," in *Proceedings Eighth International Workshop on Research Issues in Data Engineering. Continuous-Media Databases and Applications*, pp. 139–146, IEEE, 1998.

[27] S. AB, "Spotify for developers," 2021.

[28] T. Giannakopoulos, "Deep audio features," 2020.

[29] J. A. Russell, "A circumplex model of affect.," *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[30] R. E. Thayer, *The biopsychology of mood and arousal.* Oxford University Press, 1990.

[31] A. C. North, A. E. Krause, L. P. Sheridan, and D. Ritchie, "Energy, typicality, and music sales: A computerized analysis of 143,353 pieces," *Empirical Studies of the Arts*, vol. 35, no. 2, pp. 214–229, 2017.

[32] T. Contributors, "Pytorch documentation," 2019.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[34] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

[35] Plotly, "Introducing dash," 2017.

[36] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, p. e0144610, 2015.

[37] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 209–216, 2007.

[38] A. Bellet, A. Habrard, and M. Sebban, "Metric learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 9, no. 1, pp. 1–151, 2015.

[39] J. N. Brian McFee, Jongpil Lee, "Metric learning for music information retrieval," 2020.

[40] G. Elhami and R. M. Weber, "Audio feature extraction with convolutional neural autoencoders with application to voice conversion," tech. rep., 2019.

[41] M. Defferrard, "Structured auto-encoder with application to music genre recognition," tech. rep., 2015.