POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria delle Telecomunicazioni
Dipartimento di Elettronica, Informazione e Bioingegneria

Analysis of Musical Structure: an Approach based on Deep Learning

Supervisor: Prof. Augusto Sarti
Assistant supervisors: Ing. Michele Buccoli, Dr. Massimiliano Zanoni

Master graduation thesis by:
Davide Andreoletti, ID 799420

Academic Year 2014-2015

POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria delle Telecomunicazioni
Dipartimento di Elettronica, Informazione e Bioingegneria

Analisi della Struttura Musicale: un Approccio basato sull'Apprendimento Approfondito

Relatore: Prof. Augusto Sarti
Correlatori: Ing. Michele Buccoli, Dr. Massimiliano Zanoni

Tesi di Laurea di:
Davide Andreoletti, ID 799420

Anno Accademico 2014-2015

To my dear friend Mary.

Acknowledgments

I would like to thank Prof. Augusto Sarti for giving me the opportunity to realize a such interesting project. I also want to thank all the members of the ISPG laboratory, an amazing place where it is possible to learn extremely beautiful things. I am particularly grateful to Massimiliano Zanoni and Michele Buccoli for their precious suggestions. Michele has been more than an assistant supervisor; he has always helped and supported me, especially when things were not going as expected.

I would like to thank all the amazing people that I have met in my years in Milano, since each one of them has been somehow important to me. I want to particularly thank my PoliMi friends Andrè, Domè, Erica, Alessandra, Fabio, Sandra, Giulia, Baio, Shari, Marco, Roberta, Milton, Marco, Manuel, Simone, Lorela, Lollo, Giulia and all the others that I can not list. Thank to Bomber for his craziness; i will never forget how he revitalized that night. Thank to my special coffee friends Chiara and Silvia, hoping that Karma will not punish us. I would like to mention all my roommates: Alessio, Luca, Etis and Fariborz.

A special thank goes to my Lebanese brother Omran. Our "pinguedine" and my Arabic proficiency are still the same of the famous September's plan, but we have a lifetime to fix the problem. Another very special thank goes to Luca, Sanviz, Pollins, Quoc and Andre, who have been my family and who have formed the best-performing team Poli has ever seen. I am very lucky to have spent these beautiful years with such brilliant people.

Thanks to my old friends Alex, Lillo, Pol, Zoki, Kuzz and Kille, who are the people I absolutely feel the most comfortable with. Thank to Valentina for her rare quality: sincerity.

I would really like to thank my dear friend Mary, even though she can not read these pages. I will never forget her crazy ideas and, above all, her hugs.

Finally, I am grateful to all my family. Thank to my cousin Elena for her complicity with me. Thanks to my sisters Angela and Chiara, whose continuous fights have the effect of keeping me always awake. And, most of all, thanks to my mother and father, who gave me the gift of life.

My sincere thanks go to each one of you.

Abstract

A distinctive trait of the digital era is the easy accessibility to an enormous quantity of music content. The Music Information Retrieval (MIR) is a broad research field with the principal aim of extracting salient information from the audio signals. In this way, the organization of contents in large libraries results incredibly facilitated. Music Structural Analysis is one of the topics in MIR.

The purpose of the Music Structure Analysis is to retrieve the structure of songs at the largest temporal scale, i.e., its division in structural parts like the Chorus and the Verse, in automatic fashion. The analysis of the structure of songs benefits in several areas: for example, the improvement of the auto-tagging systems, or the generation of audio thumbnails, i.e., representative summaries of the song.

Music Structural Analysis mainly focuses on the detection of the temporal variation of some characteristics along the music piece. Among the characteristics that are commonly used there are the harmony, the timbre and the rhythm. However, the selection of these features is often a problematic procedure, since it is necessary to know which are the most effective to perform the task, and consequently build procedures to compute them. The obtained descriptors are generally called *hand-crafted*, since they are specifically designed to represent the sought properties. An alternative approach is represented by the deep learning techniques, which are able to automatically obtain an abstract representation of data, without the explicit knowledge of the salient features to extract.

Since the deep learning techniques have proved to be effective in several areas, in this work we investigate on their use in Music Structural Analysis. More precisely, we use a Deep Belief Network to extract a sequence of descriptors that is successively given as input to several Music Structural Analysis algorithms presented in literature. We finally compare the performance of the obtained descriptors with the commonly used hand-crafted features.

Sommario

Un tratto distintivo dell'era digitale è la possiblità di accedere facilmente ad un'enorme quantità di contenuti musicali. Il Music Information Retrieval (MIR) è un vasto campo di ricerca con il principale obiettivo di estrarre informazioni salienti dai segnali audio. In questo modo, l'organizzazione di contenuti in vaste librerie risulta incredibilmente faciliatata. L'Analisi della Struttura Musicale è una delle applicazioni del MIR.

L'obiettivo dell'Analisi della Struttura Musicale è di estrarre la struttura delle canzoni alla più alta scala temporale, i.e., la sua suddivisione in parti strutturali come il Ritornello e la Strofa, in modo automatico. L'analisi della struttura delle canzoni porta benefici in alcune aree: per esempio, il miglioramento dei sistemi di auto-tagging, o la generazione di anteprime audio, i.e., riassunti rappresentativi di canzoni.

L'Analisi della Struttura Musicale si occupa principalmente di individuare la variazione temporale di qualche proprietà lungo il brano musicale. Tra le proprietà che sono comunemente usate ci sono armonia, timbro e ritmo. Tuttavia, la selezione di queste features è spesso una procedura problematica, in quanto è necessario conoscere quali siano le più efficaci per eseguire il compito, e di conseguenza sviluppare procedure per calcolarle. I descrittori ottenuti sono generalmente chiamati *artigianali*, poiché sono specificatamente progettati per rappresentare le proprietà cercate. Un approccio alternativo è rappresentato dalle tecniche di apprendimento approfondito, che sono in grado di ottenere automaticamente una rappresentazione astratta dei dati, senza la conoscenza esplicita delle features salienti da estrarre.

Poiché le tecniche di apprendimento approfondito hanno dimostrato la loro efficacia in diverse aree, in questo lavoro investighiamo il loro utilizzo nell'Analisi della Struttura Musicale. Più precisamente, utilizziamo una Deep Belief Network per estrarre i descrittori che sono successivamente dati in ingresso ad alcuni algoritmi di Analisi della Struttura Musicale presentati in letteratura. Infine, confrontiamo le prestazioni dei descrittori ottenuti con le features artigianali comunemente usate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The digitalization and the widespread diffusion of the Internet led to the possibility to access to an enormous quantity of information in a simple and quick way. An important aspect of such phenomenon is the increasing availability of music contents. The quantity of songs that are accessible from the Web is larger than a person can listen to in his life, and consequently both the music listening experience and the music distribution have radically changed. The traditional paradigms, in which music dealers and magazines had a role of mediation between the user and the musical contents, can not be applied to the current scenario. New approaches to the organization, recommendation and retrieving are needed. The Music Information Retrieval (MIR) is an interdisciplinary research field that aims to extract information from music content. As a consequence, it may be useful to design systems that can help to organize, understand and search songs into music libraries. MIR community includes researches in the fields of signal processing, machine learning, music theory, psychology and psychoacoustic.

The context-based approaches are classical attempts to deal with such a large collection of data. They consist in manually annotating the music contents with metadata, which are textual description associated with the songs, like title, author, genre and so on. However, this strategy lacks in objectivity (the metadata are generally collected from the users on the Web) and it does not scale well to large and heterogeneous databases. In fact, this paradigm requires that a large number of people collaborate to provide the information about the contents, and it is consequently neither complete nor flexible. An example of incompleteness is the *long-tail* phenomenon, i.e., few musical contents (generally the most famous ones) are annotated by many users, whereas no information is provided for the rest of data. As far as the flexibility is concerned, a so large number of people unlikely quickly react to the possible evolutions of the task, such as the addition of a new type of metadata in a tagging system. In order to address these issues, in the recent years a new kind of approach has been proposed, which aims at extracting information directly from the music content (content-based approaches), such as acoustic cues, harmony, mood and genre.

There are two main categories of descriptors of music: Low-level features (LLFs) and Mid-level features (MLFs). The LLFs are objective information that are extracted from the signal, like some characteristics of the distribution of its spectrum. The LLFs are deterministic, and consequently easily scalable to large amounts of data, but lack in semantics. From the LLFs it is possible to obtain, through the addition of musicological information, the MLFs, which are able to capture musical aspects like the melody and the rhythm.

Such features aim to describe the musical events and their ordered relationship, which is what distinguishes music from an arbitrary sequence of sounds. In fact, music presents a, generally hierarchical, structure at various levels, such as the harmonic and the rhythmic ones. The aforementioned features change in accordance with the variations within such structures, and can consequently be used to perform music segmentation, i.e., the operation that searches the segments that are homogeneous according to some criterion.

A typical example of music segmentation task is the Music Structural Analysis, which aims at finding the structure of songs at the largest temporal scale, i.e., at the semantic level. The structure sought is defined by *sections*, which are meaningful intervals within the song. Examples of sections are the Chorus or the Verse in Pop Music. For the sake of simplicity, in the rest of this work we will use Music Structural Analysis and Music Segmentation as synonymous. The purpose of music segmentation is to find the boundaries of the aforementioned sections, as well as to group them into sets of similar sections. Such groups are not assigned a label corresponding to their role in the structure, since music segmentation is not a classification task.

An efficient music segmentation algorithm can be beneficial in several situations. For example, it may improve music players devices offering the possibility to browse through the structural parts, or producing interactive visualization of musical pieces, or even providing representative summary of songs (*audio thumbnail*). The analysis of music structures may facilitate the automatic creation of remixed versions of songs, and may also improve the actual content recommendation systems, allowing suggestions made at the level of the structural parts.

As said, the music segmentation procedure consists in searching segments that are homogeneous respect to some characteristics, and a measure of similarity is consequently needed. A common approach is to provide the representation of songs as sequence of LLFs/MLFs, which are then pairwise compared. The quality of the comparison strongly depends on the ability of the features to describe the audio signal in term of its salient properties. Traditionally, specifically-designed signal processing procedures are exploited to extract such features, which are for this reason called hand-crafted.

However, two main problems characterize this approach. The first one is that it is necessary to precisely know which properties of the audio signal are the most appropriate to address a given task. The second one is that, even in the case of perfect knowledge of the task, the descriptors may not be able to completely represent the sought properties.

Thus, the problem of feature selection, which is common also in other machine learning tasks, has been addressed in the latest years by means of the deep learning techniques, which allow to avoid the hand-crafted feature extraction. Deep learning techniques are in fact able to extract an abstract and multilayer representation of data directly from the signal. The capability to obtain this type of representation is similar to the human brain problem solving mechanism, which consists in dividing a problem into subproblems and represent information in hierarchical fashion, with increasing abstraction. With a deep learning architecture it is possible to extract salient features from the signal in an unsupervised fashion to obtain a rather general representation of data, i.e., not task-dependent. A supervised phase can then be added in order to tune the low-level representation toward a specific target.

In this work we exploit the deep learning techniques to provide an abstract representation of music, in order to use it for the music segmentation task. The purpose is to overcome the aforementioned issues related to the hand-crafted features selection, by extracting a low-level representation directly from music data. The obtained features can then be compared with the traditional ones, using several state of the art music segmentation algorithms.

In particular, we train different topologies of a deep learning architecture, the Deep Belief Network (DBN), in unsupervised fashion, obtaining a general and not target-oriented features from the data. We then include a supervised phase in order to make the features specific for the music segmentation task. Finally, we use four music segmentation algorithms presented in literature to do a comparison between the traditional and the DBN-based features.

In the Chapter 2 we review the literature in both Music Segmentation and Deep Learning. The Chapter 3 is devoted to the presentation of the theoretical background and the tools that are used in this work. In Chapter 4 we present our system, which includes the features extraction phase and several segmentation algorithms. In Chapter 5 we show and discuss the results obtained with our approach, comparing them with the results of the state of the art. Finally, in Chapter 6 we present the conclusions and the possible future development of this work.

Chapter 2

State of the Art

In this chapter we provide an overview of the literature in the two main topics of the thesis: Music Segmentation and Deep Learning.

In Section 2.1 we describe the task of music segmentation. In particular, we explain how the music segmentation process requires a feature extraction phase, which is described in Section 2.1.1. Next, we present the two main categories of segmentation algorithms, respectively in Section 2.1.2 and Section 2.1.3. Finally, in Section 2.2 we introduce the application of Deep Learning techniques in the Music Information Retrieval field, and how they can be beneficial in extracting salient features for the music segmentation.

2.1 Music Segmentation

In Section 2.1.1 we show the extraction of the low-level features that have been proven to be effective for the music segmentation task. Then, we present the two main categories of segmentation algorithms, which perform the structure analysis by considering different principles of segmentation. In [1], indeed, it is stated that the main principles are the homogeneity, the contrast, the repetitions, the temporal order and the variations. In Section 2.1.2 we present the state-based methods, which search the music structure in terms of homogeneity and contrast, whereas in Section 2.1.3 we describe the sequence-based methods, which search the music structure in terms of repetitions, temporal order and variations.

2.1.1 Feature Extraction

In [2], a conducted experiment shows that people, when performing a music segmentation task, tend to give more relevance to timbre, harmony and rhythm. Consequently, signal processing strategies are exploited to extract from the audio signal the low-level descriptors that better capture the aforementioned properties. The feature extraction is performed for each frame of the song.

As far as harmony is concerned, the most often used descriptor is the *Harmonic Pitch Class Profile* (HPCP) [3, 4, 5], which accounts for the distribution of energy over the musical notes, regardless to the octave. However, HPCP is negatively affected by key-transposition, which commonly occurs in songs. In [4] Nieto et al. compute the 2D-Fourier Magnitude Coefficients on a HPCP representation, obtaining a feature robust with respect to key-transposition.

As far as timbre is concerned, the *Mel-Frequency Cepstral Coefficients* (MFCCs), which describe the spectral envelope of the signal, are generally computed. MFCCs have been initially applied in the field of speech recognition [6], but have proven to work well in several music information retrieval tasks [7, 8], including music segmentation [9, 10, 11].

As far as rhythm is concerned, a common descriptor is the *rhythmogram*. Rhythmic patterns are connected to note onsets, which are detected by a function called *perceptual spectral flux* (PSF). From the autocorrelation of the PSF it is possible to derive the rhythmogram. However, rhythm has not received as much attention as the harmony and the timbre for the music segmentation task. To the best of our knowledge, the rhythmogram has been adopted only once [5].

Once the descriptors have been extracted for all the frames, the song can be described as a sequence of low-level features, which are dependent on tempo variations. Analogously to the key-transposition, tempo variations commonly occur in songs, and two segments may be considered similar even if they are played with different tempo. In order to make the segments robust to tempo variations, some solutions have been proposed. A method to make the features tempo-invariant is proposed in [12], where the issue is addressed making the features beat-synchronous, i.e., averaging the descriptors on intervals whose length is variable and synchronized with the tempo.

The obtained feature sequence is generally used to compute the Self Similarity Matrix (SSM), a matrix collecting the similarity between all the pairs of frames, by means of a distance measure computed over the corresponding features. SSM is introduced in [13], and it is exploited in several segmentation algorithms [3, 13, 14]. In [15] the SSM is computed over some tempo-scaled versions of the same song, obtaining robustness to tempo variations.

There are some issues in regard to the extracted low-level features. First of all, they are just loosely related to the properties that they aim to capture

[16]. Moreover, their design has to face the conflicting goals of sensitivity to segment changing from one side, and robustness to variations among similar segments from the other side [15].

We see in Section 2.2 how deep learning techniques may address the two issues, by extracting an abstract representation of data.

2.1.2 State-based

The state-based approaches search groups of frames that are homogeneous with respect to some acoustical properties. Hence, they are particularly effective with songs whose acoustic properties do not significantly varies within sections [14].

In the state based-approaches, two main phases can be identified: the detection of the instants that define sections (boundaries) and the clustering of similar sections. Generally the two sub-tasks are sequentially accomplished.

A common method to estimate boundary positions is based on the novelty curve. Such approach consists in computing a measure of distance between all the adjacent frames. The operation can be performed either on the main diagonal of the SSM [17] or directly on the feature sequence [18]. The result is the novelty curve, from which boundaries are estimated, e.g., by thresholding [19, 17]. However, this technique may lead to an issue known as oversegmentation, i.e., too many boundaries are found. In [14] such problem is addressed by means of the Non-Negative Matrix Factorization applied to the SSM; the capability of this matrix factorization to yield part-based representation of data is exploited to cluster consecutive segments previously obtained from the novelty curve, which are merged as one single segment. The NMF can be modified with the addition of the convexity constraint, which makes the structural parts more clear in the decomposition matrices. In [20] the Convex Non-Negative Matrix Factorization (C-NMF) is used to detect boundaries.

A common strategy to perform the clustering task consists in using statistical and algebraic operations. In [21], for example, it is defined an algorithm that realizes the boundaries extraction phase dividing the feature sequence into fixed-length contiguous segments. Each one of the resulting segment is then statistically modeled by means of a Gaussian distribution, and compared to all the others. Finally, similar clusters are merged in hierarchical fashion. In [22] an analogous statistical model is applied to the segments obtained with the novelty approach. After that, segments are pairwise compared, yielding to a segment-indexed SSM, which is much smaller than its frame-indexed counterpart; the reduced size of the new matrix makes easier the application of another type of matrix factorization, the Single Value Decomposition (SVD). SVD has many applications [23], including the image segmentation, and it is consequently appealing when dealing with the SSM, which can be treated as an image representing structural information. Such factorization produces a certain number of elements, each one accounting for a specific cluster, and in addition corrects the oversegmentation errors produced in the previous phase.

Boundaries extraction and clustering are also jointly carried out in [24], where a MFCC sequence is the input of an Hidden Markov Model (HMM). However, the sequence of states obtained applying the Viterbi algorithm to the HMM exhibits oversegmentation. To address this problem, in [25] the author finds an initial set of clusters from the SSM, from which a new sequence of features is derived; the HMM is then trained over this new sequence, obtaining a substantially reduced number of segments.

In [26] the Dynamic Texture Model (DTM), which is an HMM-like tool in which the hidden states assume continuous instead of discrete values, obtains better results with respect to the HMM, in the music segmentation task. The DTM is able to represent smoother state transitions, and it consequently captures the higher-level dynamics of the audio, like its rhythmic qualities and its temporal structure.

2.1.3 Sequence-based

The sequence-based methods search repeating patterns within the feature sequence, and are for this reason also called repetition-based. Differently from the state based methods, the temporal order of the features is a crucial point in sequence-based algorithms, since two subsequences cannot be considered similar if the order in which their elements appear is not the same.

Also the repetition-based methods work with the SSM, in which repetitions appear as stripes parallel to the main diagonal. Unfortunately, it is not always easy to detect them, due to distortions introduced by several factors, like variations in dynamics and timbre, or execution of note groups, modulation, articulation, and tempo progression [15, 27]. For this reason, the common strategy consists in enhancing the quality of the SSM by means of two typical image processing techniques: morphological operators and filtering. Morphological operators are used in order to fill small breaks and to remove too short line segments [28, 29], while filtering is used for noise reduction [12, 30, 31]. The tempo-invariant features described in Section 2.1.1 prevent the stripes to be not perfectly parallel to the main diagonal of the

SSM.

After the enhancement of the SSM, it is possible to apply some algorithms to find the stripes. In [32] the authors interpret the value corresponding to each entry of the SSM as a probability value, and define a transition cost. The Viterbi algorithm is then used to find the optimal paths, which likely correspond to the searched stripes. Another method involves the computation of the Hough transform, a tool used to detect straight lines in the image processing field. In particular, in [33] the Hough transform is computed on the SSM derived from an HMM-model of the audio. An alternative is presented in [34], in which pattern recognition techniques are used to find repetitions directly on the feature sequence.

2.2 Deep Learning in Music Information Retrieval

Deep learning architectures are machine learning techniques inspired by the structure of the mammal brain, and intend to reproduce its problem solving approach, which consists in organizing the external stimuli in hierarchical fashion, with progressively higher abstraction [35]. The purpose is to extract descriptors that are close to the way humans organize information, so that to overcome some issues related to the hand-crafted feature selection. In particular, in the traditional approach, it is necessary to select which musical aspects have to be described, and then to find the descriptors that better accomplish such task. However, even the most appropriate descriptors may not perfectly capture the sought musical properties, making the expressive power of the representation rather inadequate, resulting in the need for complex semantic interpretation method [36].

On the other hand, the abstract and multilayer representation found with deep learning techniques is appropriate in several artificial intelligence tasks [37], and as far as music is concerned, a multilayer representation is consistent with the way human brain organizes it in hierarchical fashion. For this reason, such representation can have a good impact on the MIR systems performance, which are highly dependent on the quality of the extracted features.

It is possible to identify two different types of training strategies for the deep architectures: the unsupervised approach and the supervised approach. The former is done over a set of unlabeled data, and it aims at extracting a rather general representation of the data. The latter requires a labeled set, and makes the network able to extract a target-oriented representation of data. In [38] it is shown that a type of deep architecture, the Deep

Belief Network (DBN), can be trained in both unsupervised and supervised fashion, as detailed in Chapter 3.

The DBN is adopted in the context of MIR, for several tasks. In [39] the DBN is trained in unsupervised way in order to extract significant low-level features, which are successively elaborated by a supervised machine learning system in order to detect whether or not a song is a bootleg, i.e. an unofficially-recorded and redistributed content. The proposed method outperforms the state of the art, showing that features directly learned from music might better represent audio than the heuristically designed. A DBN sequentially trained in unsupervised and supervised way is successfully used also in [40], with the purpose to learn emotion-related features. In [41] the same approach is adopted, in order to perform the task of genre recognition and auto-tagging, obtaining better results with respect to the MFCC features.

In [42] deep learning is employed for music segmentation. In particular, a Convolutional Neural Network (CNN) is trained in supervised fashion and employed for the task of boundary detection. The results show a substantial improvement with respect to all the others boundary detection techniques. However, deep learning techniques have not yet been employed for both the tasks of boundary detection and clustering. In this work we use a DBN, trained in both unsupervised and supervised way, to find a representation of music pieces. We then use such representation as input for some segmentation algorithms present in literature.

Chapter 3

Theoretical background

In this chapter we present the theoretical background and the tools we need to develop our project. Initially, we focus on how signal processing tools are used to analyze the audio signal and to obtain descriptors of relevant acoustic properties. In particular, we present two widely adopted descriptors, the Harmonic Pitch Class Profile and the Mel-frequency Cepstral Coefficients, respectively described in Section 3.1.2 and Section 3.1.3.

Then, we show some basic music segmentation tools. We describe the Self Similarity Matrix, which collects the structural information of a song, and the Novelty Curve, which is used to detect boundaries.

Finally, we introduce some machine learning techniques, showing how they can be used to extract a representation of the input data. In particular, we describe the neural networks and the deep neural networks.

3.1 Signal Processing Tools

In the MIR field, signal processing strategies are exploited to analyze the audio signal in order to extract low-level descriptors, which are able to describe specific acoustic cues. This section gives an overview of the basic signal processing tools used in the music segmentation task, together with the description of two commonly extracted descriptors. Initially, we focus on the *Short-Time Fourier Transform* (STFT), which provides a frequency versus time representation of the signal, from which the Harmonic Pitch Class Profile and the Mel-frequency Cepstral Coefficients can be derived. The former, often referred to as chromagram, captures information on the harmonic content of the song, while the latter describes its timbre.

3.1.1 Short-Time Fourier Transform

The Discrete Fourier Transform (DFT) is a widely adopted transform used to compute the frequency-domain representation of a discrete-time signal:

$$X(\omega_k) = \sum_{n=0}^{N-1} x(n) e^{-j\omega_k n}, \qquad k = -\frac{N}{2}, \dots, \frac{N}{2}, \tag{3.1}$$

where:

- $\omega_k = 2\pi \cdot F_s \frac{k}{N}$ is the k-th frequency sample, which is also called *bin*;
- N is the number of samples of the time sequence;
- f_s is the sampling frequency;
- $X(\omega_k)$ is the frequency content of the signal at k-th bin;
- $x(n)$ is the n-th time sample of the signal.

For wide signals, it is useful to track the frequency content in its evolution over time. However, the DFT does not provide information on the temporal occurrence of the frequency content, hence the Short Time Fourier Transform (STFT) is employed.

The STFT is computed by means of a windowing process. The window is a simple function, with support N_{stft} (i.e., the interval where the function is not zero-valued), which slides over the signal with an hop size N_{hop}. The result is the division of the signal into a set of frames of length N_{stft}, which can be made overlapping if $N_{hop} < N_{stft}$. When the frames overlap, the obtained representation provides more information on the temporal dynamic of the signal.

$$X_r(\omega_k) = \sum_{n=0}^{N_{stft}-1} x(n - rN_{hop}) w(n) e^{-j\omega_k n}, \qquad k = -\frac{N_{stft}}{2}, \dots, \frac{N_{stft}}{2}, \tag{3.2}$$

where:

- $x(n - rN_{hop})w(n)$ is n-th sample of the r-th frame;
- $\omega_k = 2\pi f_s \frac{k}{N}$ is the k-th frequency bin;
- $X_r(\omega_k)$ is the r-th frame content at k-th frequency bin.

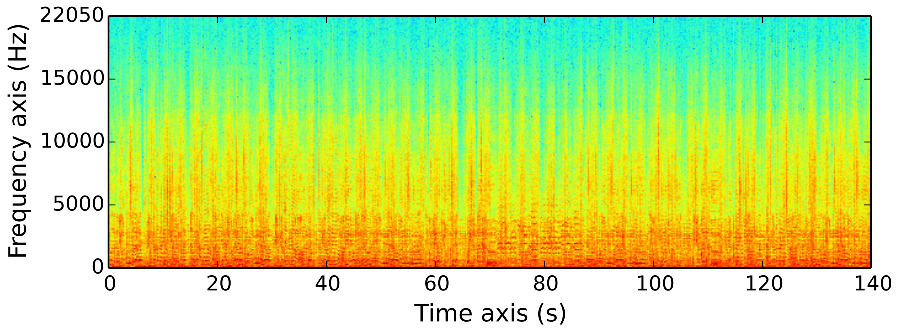


Figure 3.1: Spectrogram of the song "You are going to loose that girl" by The Beatles

It is worth to notice that there is a trade off on the frame length N_{stft}: the longer are the frames, the higher is the frequency resolution, and the lower is the temporal one. The frequency resolution of a discrete-time signal is given by:

$$\triangle f = \frac{F_s}{N_{stft}} \quad . \tag{3.3}$$

A spectrum is computed for each frame, providing a frequency versus time representation of the signal. Since the spectrum of real signal presents Hermitian symmetry, it is possible to discard the components related to the negative frequencies without loosing information on it. Since the phase is not as informative as the magnitude [43], generally only the latter is considered. A STFT where the magnitude is computed over only the components related to positive frequencies is referred to as *spectrogram*. An example of spectrogram is depicted if Figure 3.1.

3.1.2 Harmonic pitch class profiles

The *Harmonic pitch class profiles* (HPCP) aims to describe the harmonic content of a frame. The harmony is related to musical pitches, that have two dimensions: height and chroma [44]. The former refers to the octave to which a note belongs, while the latter tells where a note stands within that particular octave. The HPCP is often referred to as *chromagram* and *chroma*. The chromagram collects the distribution of the signal's energy across a predefined set of pitch classes, whose number is generally 12, 24 or 36 [44]. Since the vector represents the note distribution, its components are also referred to as bin. The common choice is a vector with 12 components, one for each semitone of the equal-tempered musical scale. In this way each bin contains the information about a semitone in a certain frame, discarding the octave to which it belongs. Various procedures have been proposed to compute the chromagram; in the following we show the method described in [44], where the chromagram is used for the task of chord recognition.

A constant Q-transform is computed over the spectrogram of a song. The constant Q-transform converts the linearly-spaced frequency scale into a filter bank of geometrically spaced filters, according to the formula:

$$f_k = f_0 2^{\frac{k}{B}}, \tag{3.4}$$

where B is the number of filters per octave, and f_0 is a reference frequency. The ratio between the center of a filter and its width is constant, and the result is a logarithmically-spaced frequency scale, which is close to the human auditory system [45].

The parameters of the constant Q-transform can be tuned in order to match the tempered scale, in which the frequencies relative to tones are geometrically spaced, such that the filterbank of the new scale is centered on the frequencies of the musical tones. Finally, the chromagram vector is obtained by summing up the contribution of each tone over the all the octaves:

$$CH(r,b) = \sum_{m=0}^{M-1} \left| X_{CQ}\left(r, b + mB\right) \right|, \tag{3.5}$$

where $|X_{CQ}|$ is the magnitude of the Q-transformed frame spectrum, b is the chromagram bin ($b = 1, ..., 12$ in our case), M the number of octaves of the spectrum, and $CH(r,b)$ the chroma coefficient relative to the bin b of the r-th frame.

Since the chromagram encodes musically meaningful concepts (the tones) rather than simple properties of the signal, it is considered a mid-level descriptor. In Figure 3.2 it is shown an example of a chromagram. Around second 60, it is possible to notice that the D tone is the most present. Moreover, since the harmonic pattern present around second 60 is the same at second 100, the two intervals probably belong to the same structural part.

3.1.3 Mel-frequency cepstral coefficients

The Mel-Frequency Cepstrum Coefficients (MFCCs) encode the timbral properties of the signal, and are commonly adopted in timbre-based structure analysis methods [9, 10, 46].

The MFCCs are computed from a modified version of the spectrogram: in particular, the amplitude is logarithmically transformed and the frequency

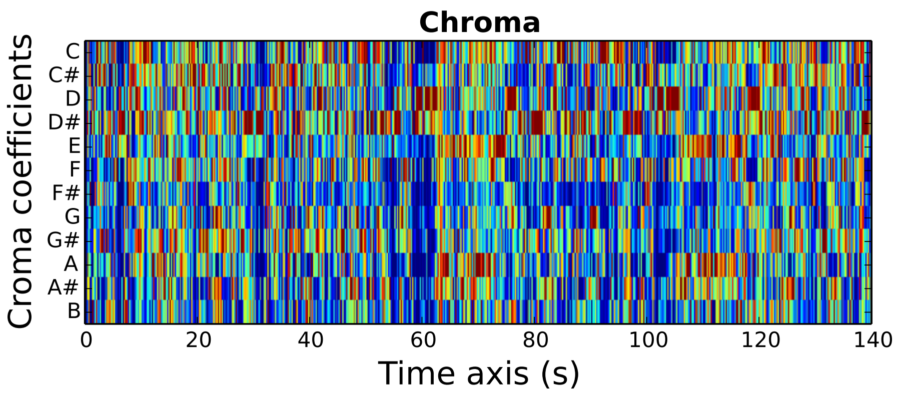


Figure 3.2: Chromagram of the song "You are going to loose that girl" by The Beatles

linear scale is converted to a logarithmically-spaced one, in order to approximately reflect the human's ear perception [47]. The conversion to the logarithmic frequency scale is obtained with the following equation:

$$f_{log} = 2595 \log_{10}(1 + f_{lin}/700). \qquad (3.6)$$

The spectrum is then pass-filtered by a mel-filter bank, depicted in Figure 3.3, and the energy within each filter is summed up.

The computation of the MFCC is performed on the resulting reduced Power Spectrum, by means of another Fourier-related signal processing tool, the Discrete Cosine Transform (DCT). Among the several variants of DCT, we show how to compute the DCT-II, as found in [48]:

$$C_x(k) = \sum_{n=0}^{N-1} 2x(n) \cos\left(\frac{\pi}{2N} k(2n+1)\right), \qquad k = 0, 1, ...N-1, \qquad (3.7)$$

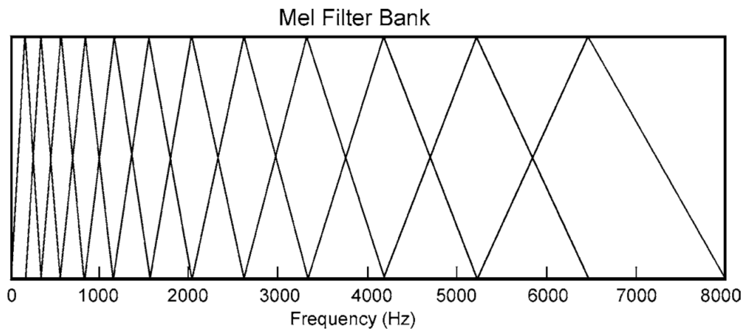


Figure 3.3: Filterbank needed to compute the MFCC

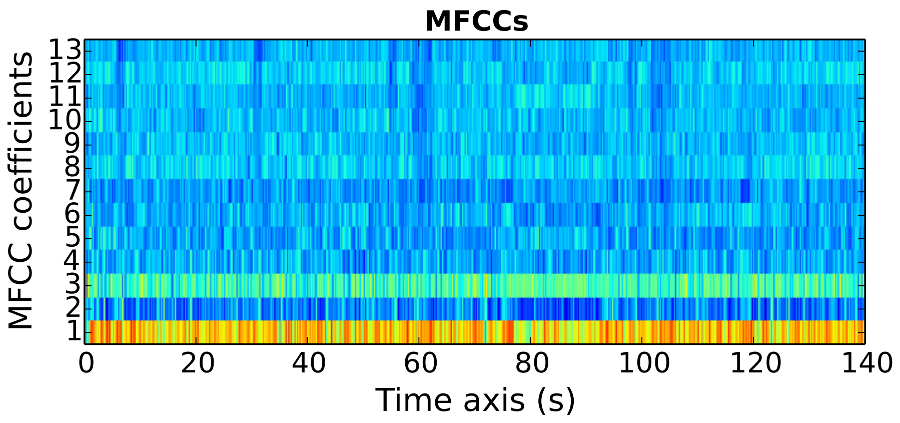


Figure 3.4: MFCC of the song "You are going to loose that girl" by The Beatles

where $C_x(k) \in \mathbb{R}$ is the k-th coefficients of the DCT computed on the discrete-time signal $x(n)$, and N is the number of samples of the transformed signal. Generally only the coefficients from 1 to 13 are kept, because they are considered the most informative [49], while the others are discarded. Differently from the chromagram in Figure 3.2, in the example of MFCC depicted in Figure 3.4 it is not easy to find a recurrent timbral pattern. However, we can notice that the energy of the signal is mostly distributed in bin 1 and in bin 3.

3.2 Music Segmentation

The music segmentation concerns two subtasks: the boundary detection and the labeling of similar parts. The boundaries define the sections in which the song is divided, while the labels are assigned to sections in order to highlight their similarity. In particular, the same label is assigned to sections that belong to the same cluster, i.e., the same structural part.

3.2.1 Self Similarity Matrix

The *Self Similarity Matrix* (SSM) is a square matrix collecting a measure of similarities for all the pair of frames in which a song is divided. The SSM has been introduced by Foote in [13] as a tool needed to perform music segmentation. The similarity between each couple of frames is evaluated by means of a distance function. High level of similarities (i.e., low measures of distance) must be assigned to similar frames, so that the SSM can be seen as an image representing structural information. Ideally, the entries of the

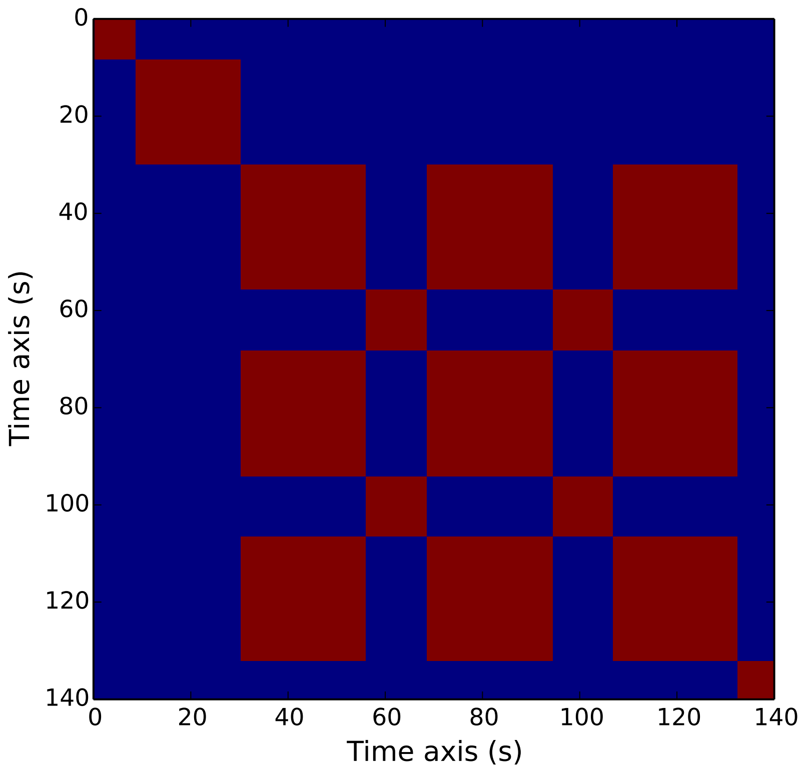


Figure 3.5: The ideal SSM for the song "You are going to loose that girl" by The Beatles

SSM are zero (minimum distance) if the corresponding frames belong to the same structural part, and are one (maximum distance) otherwise:

$$SSM(i,j) = \begin{cases} 0 & i\text{-th and } j\text{-th frames belonging to the same cluster} \\ 1 & \text{otherwise} \end{cases}$$

An example of ideal SSM, represented as an image, is given in Figure 3.5. It is possible to notice that the section around second 60 and the section around second 100 belong to the same structural part.

In order to compute the SSM relative to a song, it is necessary to convert it into a suitable sequence of descriptors (i.e., the feature extraction phase). Then, those descriptors are pairwise compared by means of a distance function. The three commonly adopted distance functions are:

$$D_{euclidean}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|,$$

$$D_{cosine}(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u}\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|},$$

$$D_{correlation}(\mathbf{u}, \mathbf{v}) = 1 - \frac{(\mathbf{u} - \mu_u) \cdot (\mathbf{v} - \mu_v)}{\|\mathbf{u} - \mu_u\|\|\mathbf{v} - \mu_v\|},$$

where $\|\cdot\|$ is the euclidean norm, $\mathbf{u}$ and $\mathbf{v}$ are two vectors (in our case the descriptors relative to a pair of frames), and μ_u and μ_v are the mean values of such vectors.

Once the distances have been computed for all the couple of frames, they are reorganized into matrix form. The generic entry of the SSM is given by:

$$SSM(i, j) = D(\mathbf{x}_i, \mathbf{x}_j), \tag{3.8}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the descriptors of i-th and j-th frames, and $D(\cdot)$ is a measure of distance (normalized between 0 and 1). If the distance measure is symmetric, the resulting SSM is symmetric too.

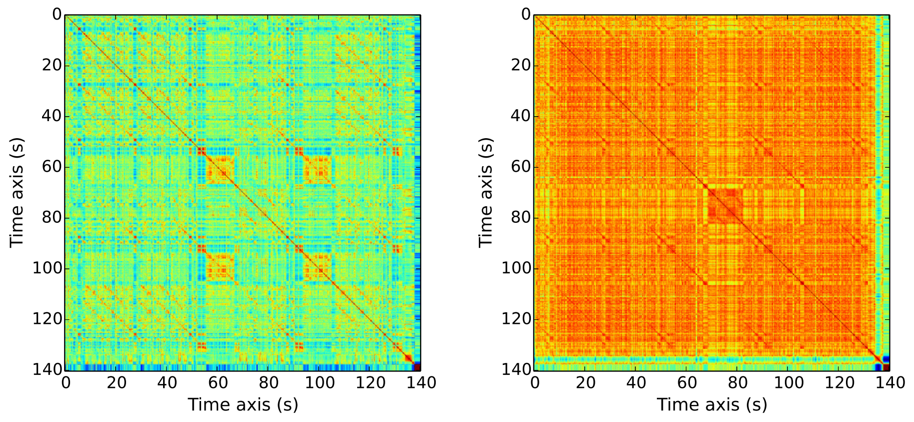
The SSM changes according to both the descriptors and the distance measure. In the following we show some SSMs computed with the above mentioned distances, for a chroma representation (Figure 3.6a, Figure 3.7a, Figure 3.8a) and for a MFCC representation (Figure 3.6b, Figure 3.7b, Figure 3.8b).

3.2.2 Novelty Curve

Given an ordered list of elements, the novelty curve (NC) is a tool used to measure how much two adjacent elements are dissimilar. In [13] Foote uses the novelty curve in the field of music segmentation, where such list is the feature sequence in which the song is transformed. The NC is linked to the concept of contrast, since it shows relevant peaks when the feature sequence abruptly changes. A common way to compute the NC involves the correlation of a kernel along the main diagonal of a SSM:

$$NC(i) = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m, n) SSM(i + m, i + n), \tag{3.9}$$

where $C \in \mathbb{R}^{L \times L}$ is the kernel. Depending on the value assumed by L, a different number of past and future samples can be included in the computation of the novelty curve. As far as the value of L is concerned, it is



(a) SSM computed over the chroma representation *(b) SSM computed over the MFCC representation*

Figure 3.6: Self Similarity Matrices of the song "You are going to loose that girl" by "The Beatles" computed with the euclidean distance

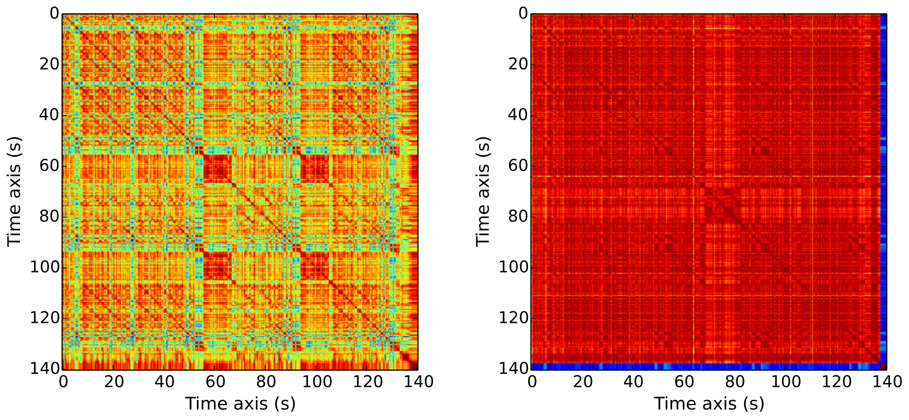
chosen in accordance with the desired temporal scale in the NC computation. As far as the type of kernel is concerned, the simplest kernel is the checkerboard one, which models the ideal shape of a boundary. An example of checkerboard kernel with $L = 6$ is given in Equation 3.10:

$$C = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \end{pmatrix}. \tag{3.10}$$

In order to give more relevance to the center of the kernel than to its edges, a smoother kernel, such as the Gaussian one, is commonly adopted.

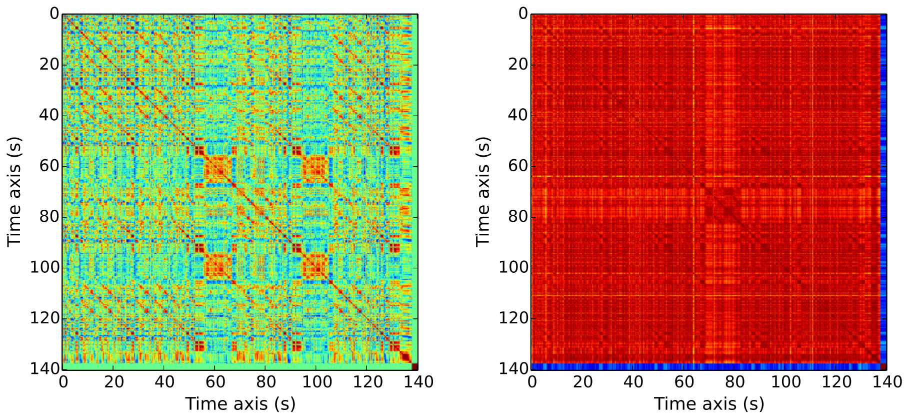
An alternative method obtains the NC directly from the feature sequence, by computing the finite differences of adjacent features.

After the computation of the NC, a commonly adopted boundary detection technique consists in searching peaks within it. Different strategies can be employed, but in general a peak is considered a boundary if it is a local maximum and if it is greater than a threshold. In Figure 3.9 we show a NC computed from the SSM in Figure 3.6a, together with the expected (i.e., the true) boundaries. It is easy to notice that the peaks of the curve do not always match with the ground truth boundaries.



(a) SSM computed over the chroma representation *(b) SSM computed over the MFCC representation*

Figure 3.7: Self Similarity Matrices of the song "You are going to loose that girl" by "The Beatles" computed with the cosine distance



(a) SSM computed over the chroma representation *(b) SSM computed over the MFCC representation*

Figure 3.8: Self Similarity Matrices of the song "You are going to loose that girl" by "The Beatles" computed with the correlation distance

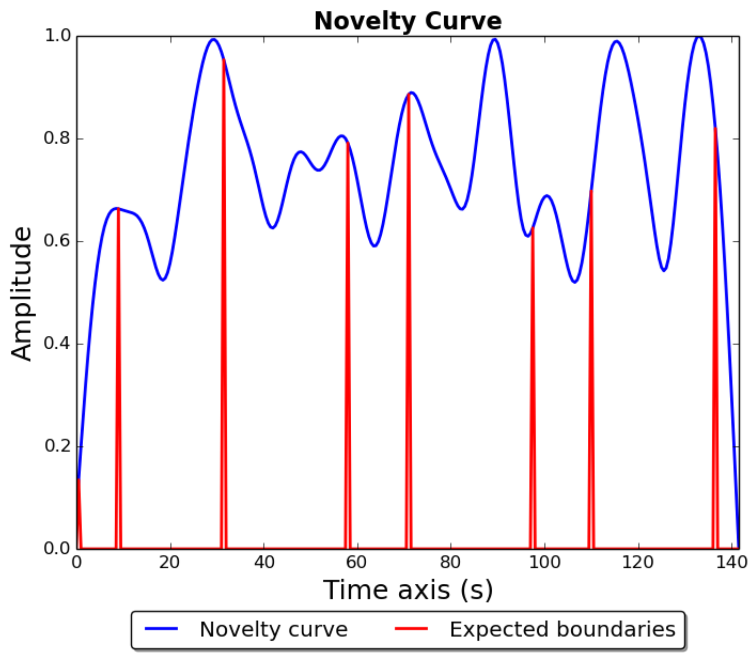


Figure 3.9: The novelty curve computer for the song "You are going to loose that girl" by The Beatles

3.3 Deep Learning Networks

3.3.1 Machine Learning

Machine learning is a research field that studies how to provide computers with the ability to learn from data without being explicitly programmed. The learning process can be of two different kinds: *unsupervised* and *supervised*. The former aims at finding the unknown structure of unlabeled data, while the latter uses a labeled set to model a certain relation between raw data and the corresponding label.

The dataset is commonly split into three parts: training set, validation set and test set. The first one is used to train the machine learning algorithm. The second one is used to perform an intermediate testing, which is done during the training, in order to validate the learnt model. Moreover, the hyperparameters (parameters of the machine learning model), can be tuned according to the performance obtained over the validation set. The third one is completely separated from the learning phase, since it is used only for evaluating the performance of the model on unseen data.

3.3.2 Neural Networks

The neural networks are machine learning techniques that make use of architectures of interconnected elements called *neurons*. Such nodes implement a simple, generally non linear transformation of a combination of their inputs. An example of performed operation is the application of a function to a weighted sum of inputs:

$$a = f(\sum_{i=1}^{N} w_i x_i + b), \tag{3.11}$$

where:

- a is the activation of the neuron, i.e., the value that it assumes;
- N is the number of input components;
- x_i is the value of the i-th input component;
- w_i is the weight associated with the connection between the i-th input and the reference neuron;
- b is the bias;
- f is generally a non-linear function. A commonly adopted non-linear function is the *sigmoid* one, which is computed as:

  $$\mathrm{sigm}(z) = \frac{1}{1 + e^{-z}},$$

  where $z \in \mathbb{R}$. An example of sigmoid function is depicted in Figure 3.10.

Neurons are interconnected to form networks. Depending on the way neurons are linked, as well as the operations that they perform, different topologies of neural networks can be realized; the usefulness of this kind of architectures relies on their ability to perform operations like finding the structure of input data, or classifying them according to some criteria.

In the next sections we focus on a particular category of neural networks, where neurons are arranged in separate layers. These kinds of construction are referred to as Deep Learning Architectures, and prove to be particularly effective in several machine learning tasks [38], due to their ability to learn rich and complex models of data.

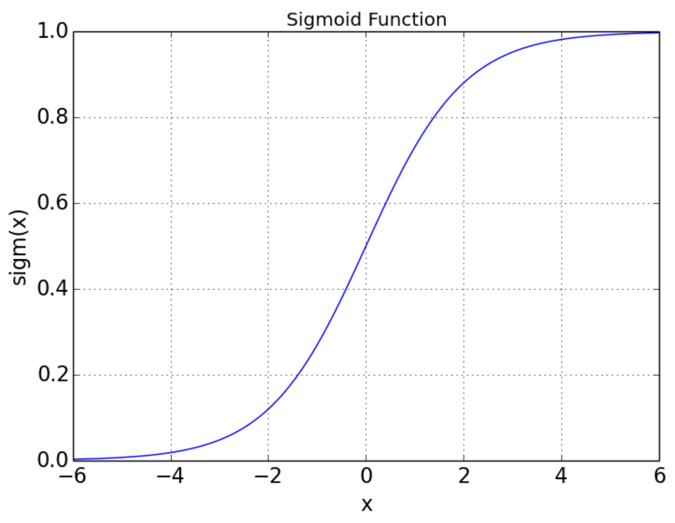


Figure 3.10: A representation of the sigmoid function, which is commonly used as neuron activation function

3.3.3 Deep Learning

Deep Learning is a class of neural network topologies that are composed by several layers of neurons. This type of arrangement aims at reproducing the human brain, which decomposes external stimuli in a hierarchical fashion [38]. The hierarchical representation also allows to decompose problems into subproblems, each associated with one or more abstract concepts. In order to mimic such capability, each layer of the deep learning architecture is used to process the information of the previous one, and the obtained representation of the data is made progressively more abstract.

Different deep learning architectures can be constructed, depending on the type of adopted layer. In this thesis, we use a Deep Belief Network (DBN), which is formed by the stacking of several Restricted Boltzmann Machines (RBM). The RBM is another type of neural network, which belongs to the family of the energy-based models.

3.3.4 Energy-based models

In the energy-based models, an energy function associates a scalar value to each configuration of the variables of interest. The function is generally designed in such a way that small energy values are assigned to configurations that are likely to occur, whereas high values are assigned to the implausible ones [50]. In this way the function can be used to model a probability distribution. A formal relation between energy function and probability distribution has been proposed in [38]:

$$P(\mathbf{x}) = \frac{e^{-\text{Energy}(\mathbf{x})}}{Z}, \tag{3.12}$$

where Energy($\mathbf{x}$) and $P(\mathbf{x})$ are the energy and the probability associated with a particular configuration $\mathbf{x}$, respectively. Z is a normalization factor, and is defined as:

$$Z = \sum_{\widetilde{\mathbf{x}}} e^{-\text{Energy}(\widetilde{\mathbf{x}})}, \tag{3.13}$$

where $\widetilde{\mathbf{x}}$ is a generic input, and the sum is run over the input space. In some situations, it is useful to introduce also hidden variables that can capture salient relationships among the input ones, with consequent improvement of the expressive power of the model. In such model, it is common to refer to the input variables as to the *visible* ones. The Equation 3.12 results slightly modified:

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-\text{Energy}(\mathbf{x},\mathbf{h})}}{Z}, \tag{3.14}$$

where $\mathbf{x}$ and Z still denotes the visible variables and the normalization factor respectively, while $\mathbf{h}$ denotes the hidden variables. Since we observe only the visible variables, we derive the marginal probability:

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) = \sum_{\mathbf{h}} \frac{e^{-\text{Energy}(\mathbf{x},\mathbf{h})}}{Z}, \tag{3.15}$$

with the sum that is over the space of all the hidden variables spanned by $\mathbf{x}$. In order to write a relation similar to 3.12, it is necessary to introduce the physic-inspired concept of free energy:

$$\text{FreeEnergy}(\mathbf{x}) = -\log \sum_{\mathbf{h}} e^{-\text{Energy}(\mathbf{x},\mathbf{h})}, \tag{3.16}$$

with the sum that is, again, run over the space of all the hidden variables spanned by $\mathbf{x}$. The probability of a generic configuration $\mathbf{x}$ can now be written as:

$$P(\mathbf{x}) = \frac{e^{-\text{FreeEnergy}(\mathbf{x})}}{\sum_{\widetilde{\mathbf{x}}} e^{-\text{FreeEnergy}(\widetilde{\mathbf{x}})}}, \tag{3.17}$$

where $\mathbf{x}$ is a generic visible input and the sum is over the input space.

A training phase is needed to learn the parameters that better models the above-mentioned probability distribution. If the structure of the model is known and the energy function belongs to a family of functions parameterized by the model parameters, it is common to perform an unsupervised training that maximizes the empirical log-likelihood of the training data:

$$\arg\max_{\boldsymbol{\theta}} \frac{1}{|D|} \sum_{\widetilde{\mathbf{x}} \in D} \log P\left(\widetilde{\mathbf{x}}\right), \tag{3.18}$$

where $\boldsymbol{\theta}$ are the parameters of the model, $\widetilde{\mathbf{x}}$ is a generic training data, D is the training set and $|D|$ is the number of elements in the training set. A very common optimization algorithm is the steepest descent one, an iterative numerical method that searches the minimum of a function following the direction of its negative gradient. However, the problem in Formula 3.18 is in the form of a maximization, so that the minimization is performed on the negative log-likelihood [38]:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{|D|} \sum_{\widetilde{\mathbf{x}} \in D} - \log P\left(\widetilde{\mathbf{x}}\right). \tag{3.19}$$

In the energy-based models with hidden units, the gradient is computed as [38]:

$$\frac{\partial \log P(x)}{\partial \boldsymbol{\theta}} = -\frac{\partial \log \text{FreeEnergy}(x)}{\partial \boldsymbol{\theta}} + \sum_{\widetilde{\mathbf{x}}} P(\widetilde{\mathbf{x}}) \frac{\partial \text{FreeEnergy}(\widetilde{\mathbf{x}})}{\partial \boldsymbol{\theta}}. \tag{3.20}$$

As we said, the steepest descent algorithm follows the direction of the negative gradient. However, since we are trying to model the distribution of a training set, the above gradient is replaced by the average of the gradient itself:

$$E_{\hat{P}}\left[\frac{\partial \log P(\mathbf{x})}{\partial \boldsymbol{\theta}}\right] = E_{\hat{P}}\left[\frac{\partial \text{FreeEnergy}(\mathbf{x})}{\partial \boldsymbol{\theta}}\right] - E_{P}\left[\frac{\partial \text{FreeEnergy}(\mathbf{x})}{\partial \boldsymbol{\theta}}\right], \tag{3.21}$$

where $\hat{P}$ is the distribution of the training data, and P is the distribution of the input variables (i.e., the probability distribution of the elements in the input space). Therefore, if we could infer the distribution of the input variables, and if we could tractably compute their free energies, we would obtain an estimator of the log-likelihood gradient. In the following section we describe how this problem is addressed when the energy-based model is the Restricted Boltzmann Machine.

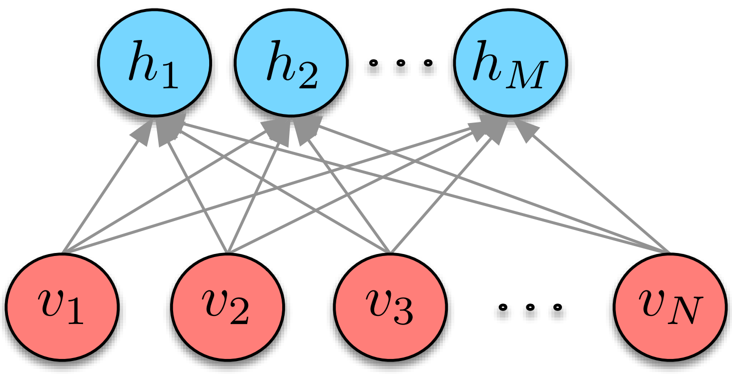


Figure 3.11: A representation of RBM

3.3.5 Restricted Boltzmann Machines

A Restricted Boltzmann Machine (RBM) is a stochastic neural network which is composed by a visible layer v and a hidden layer h. The two layers are fully connected, whereas no connection exists among the units in the same layer. A representation of RBM is shown in Figure 3.11. The RBM is trained (generally in unsupervised fashion) in order to become a generative model, i.e., a system able to model the joint probability of its variables. The hidden layer h provides a representation of the visible input v, and the values of both layers define a particular configuration of the network. If the RBM is successfully trained, it provides a closed-form representation of the distribution underlying the training data.

Since the RBM belongs to the family of energy-based models, it is characterized by an energy function. The energy function of the RBM (with N visible units and M hidden units, in the following example), is linear in its free parameters, and is defined as:

$$\text{Energy}(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{W} \mathbf{v}, \tag{3.22}$$

where $^\top$ indicates the transpose operation, $\mathbf{W} \in \mathbb{R}^{M \times N}$ is the matrix of weights of connection between the visible and hidden layers and $\mathbf{b} \in \mathbb{R}^{N \times 1}$, $\mathbf{c} \in \mathbb{R}^{M \times 1}$ are the bias vectors, respectively for the hidden and input layers. Since the RBM presents also the hidden variables, it is characterized by a free energy function:

$$\text{FreeEnergy}(\mathbf{x}) = -\mathbf{b}^\top \mathbf{v} - \sum_i \log \sum_{h_i} h_i e^{h_i(c_i + W_i x)}. \tag{3.23}$$

In [38] it is shown that, if the energy-based model is the RBM, the negative log-likelihood gradient can be written as:

$$\frac{\partial \log P(\mathbf{x})}{\partial \boldsymbol{\theta}} = -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}) \frac{\partial \text{Energy}(\mathbf{x},\mathbf{h})}{\partial \boldsymbol{\theta}} + \sum_{\mathbf{x},\mathbf{h}} P(\mathbf{x},\mathbf{h}) \frac{\partial \text{Energy}(\mathbf{x},\mathbf{h})}{\partial \boldsymbol{\theta}}, \tag{3.24}$$

where the first summation is run over the hidden space spanned by the input variable, while the second one is run over the whole input space. The energy functions are computed as in Equation 3.22, from which it is possible to analytically obtain the partial derivatives. Unfortunately, we can not do the same to obtain the probability distributions, and we need a way to sample them. A sample of $P(\mathbf{h}|\mathbf{x})$ can be analytically computed, due to the lack of connections between visible and hidden units in the RBM:

$$P(\mathbf{h}=\mathbf{1}|\mathbf{x}) = \prod_i P(h_i = 1|\mathbf{x}) = \prod_i \text{sigm}(\mathbf{c} + \mathbf{W}\mathbf{x}). \tag{3.25}$$

In order to compute a sample of $P(\mathbf{x},\mathbf{h})$, it is possible to employ a Monte-Carlo sampling procedure, such as the *Gibbs* sampling, which is particularly suitable for the RBM architecture [51].

More precisely, the joint distribution of N variables is obtained by running a Monte Carlo Markov Chain (MCMC) to convergence. Each node of the chain represents one of such variables, and the convergence is asymptotically guaranteed. In our case, these variables are represented by the set of visible and hidden units of the RBM, and the transition operator of the Markov Chain is the Gibbs sampling. In the generic i-th Gibbs steps, it is computed $P(x_i|x_{-i})$, which is the probability associated with the i-th random variable, conditioned to the values assumed by all the others.

The MCMC is initialized by picking an input $\mathbf{x_0}$ from the training distribution $\hat{P}$. Then, its hidden representation $\mathbf{h_0}$ is computed from the model distribution as:

$$\mathbf{h_0} = \text{sigm}(\mathbf{c} + \mathbf{W}\mathbf{x_0}).$$

Due to the symmetric structure of the RBM, another sample of the visible units is derived as:

$$\mathbf{x_1} = \text{sigm}(\mathbf{b} + \mathbf{W}^\top \mathbf{h_0}).$$

The procedure can be repeated for the desired number of Gibbs steps, but this is a very expansive operation. In order to overcome such problem, the Contrastive Divergence (CD-k) algorithm has been proposed in [52]. The CD-k algorithm introduces two approximations, which allow to obtain very good results even without waiting the convergence of the chain. The approximations are the following:

- the sample is obtained after k steps of sampling, i.e., without waiting the complete convergence of the MCMC. In particular, one step has shown to be sufficient to achieve good results (CD-1) [38];
- the expectation $E_P\left[\frac{\partial \text{FreeEnergy}(\mathbf{x})}{\partial \theta}\right]$ is replaced by a single sample, i.e., the free energy is computed only for one visible variable from the input space.

Thus, the steps of the CD-1 algorithm become:

1. given a visible input $\mathbf{x_0}$, compute $Q(\mathbf{h_0} = \mathbf{1}|\mathbf{x_0})$, i.e., the probability that the hidden nodes assume the value of 1, given the visible ones: $Q(\mathbf{h_0} = \mathbf{1}|\mathbf{x_0}) = \text{sigm}(\mathbf{c} + \mathbf{W}\mathbf{x_0})$;
2. from $Q(\mathbf{h_0} = \mathbf{1}|\mathbf{x_0})$ perform a sampling for each hidden units, in order to obtain a sample $\mathbf{h_0}$;
3. do the same operation for every visible nodes, in order to obtain a sample $\mathbf{x_1}$, starting from $\mathbf{h_0}$. $P(\mathbf{x_1} = \mathbf{1}|\mathbf{h_0}) = \text{sigm}(\mathbf{b} + \mathbf{W}^\top \mathbf{h_0})$;
4. then, find again a sample $\mathbf{h_2}$ by sampling $Q(\mathbf{h_1} = \mathbf{1}|\mathbf{x_1}) = \text{sigm}(\mathbf{c} + \mathbf{W}\mathbf{x_1})$;
5. finally, update the parameters of the network by means of the steepest descent algorithm:

$$\mathbf{W} \longleftarrow \mathbf{W} + \rho(\mathbf{h_0}\mathbf{x_0}^\top - Q(\mathbf{h_1}|\mathbf{x_1}))\mathbf{x_1}^\top,$$

$$\mathbf{b} \longleftarrow \mathbf{b} + \rho(\mathbf{x_0} - \mathbf{x_1}),$$

$$\mathbf{c} \longleftarrow \mathbf{c} + \rho(\mathbf{h_0} - Q(\mathbf{h_1}|\mathbf{x_1})),$$

   where ρ is the *learning rate* used in the steepest descent algorithm;
6. stop if the training has been performed for the desired number of epoch, otherwise go to step 2.

Notice that the average gradient of the log-likelihood is composed by two terms, one directly related to the training examples and one related to the input space, which is sampled by means of the Contrastive-Divergence algorithm. The training phase is consequently based on sampled examples that are generated by a non-optimized model. However, the two kinds of samples are treated in a different way. In fact, in the training phase we attempt to shape the energy function such that it assigns lower values to training examples and higher values to samples from the model. The model

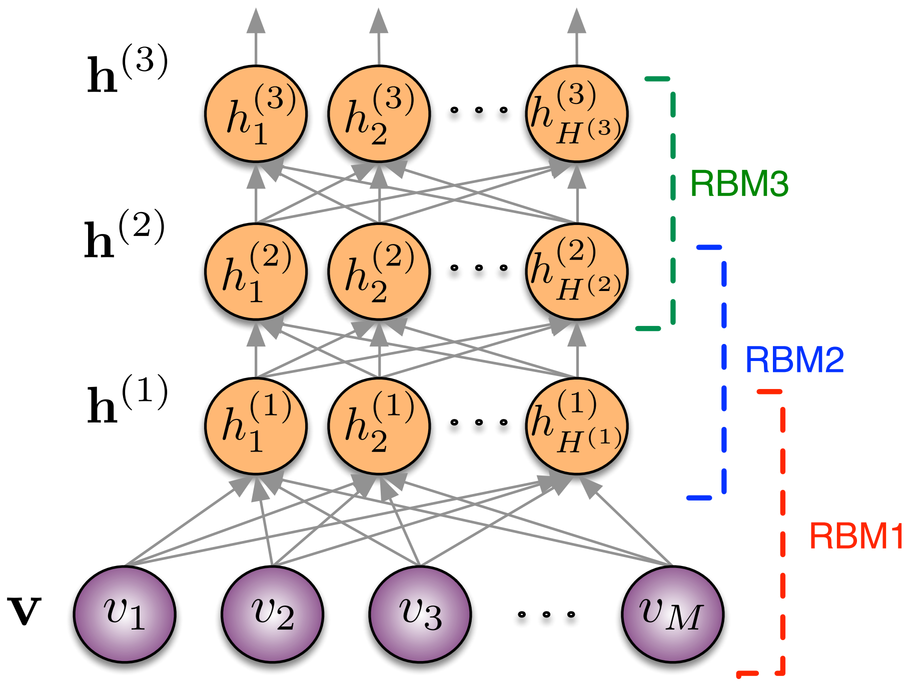


Figure 3.12: A representation of DBN

is trained in order to recognize training examples from sampled examples, and to model the probability distribution of the former with respect to the latter. This makes the RBM able to generate samples which are more similar to the training examples, and to provide an effective representation of the input.

In the following we show that the learning phase of the DBN involves a layer-wise unsupervised training of its RBM layers. Since a supervised learning phase for the DBN may be performed after the unsupervised one, ρ is often called *pre-learning rate*.

3.3.6 Deep Belief Networks

A Deep Belief Network (DBN) is a deep learning architecture which is composed by the stacking of several RBMs [53]. A representation of DBN is depicted in Figure 3.12. Differently from the simple RBM, the multi-layer structure makes the DBN able to extract more abstract features from the input data. Each RBM level takes as input the hidden layer of the previous one. In this way, it is possible to learn features from features and to derive an higher-level representation. The DBN models the joint distribution between the visible layer and its representation, which is composed by the hidden layers of the network, as:

$$P(v = h^{(0)}, h^{(1)}, \dots, h^{(L)}) = \left(\prod_{k=0}^{L-2} P(h^{(k)} | h^{(k+1)}) \right) P(h^{(l-1)}, h^{(l)}), \quad (3.26)$$

where $h^{(0)} = v$ is the input data, and it is the bottom of the DBN. The layer $k + 1$ stands on the top of the layer k, and L is the number of DBN layers. The recursive nature of the above formula is motivated by the lack of connections between nodes of the same layer, that is a property inherited from the RBM.

The DBN can be trained both in unsupervised and supervised fashion. With the former approach, the network becomes able to extract only general features from the input data. The latter is used to obtain target-oriented features, i.e., a representation of the input data that is suitable for a specific application. The supervised training is generally performed after the unsupervised one. In fact, in [52] it is shown that performing a supervised learning phase starting from a random initialization of the parameters makes the training more likely to get stuck into a local minimum of the cost function. However, if the model is initially trained in unsupervised fashion, the parameters of the network are driven in a better region of the parameter space, i.e., a region from which the network can more easily converge to a good solution.

Unsupervised learning for a DBN

The DBN is formed by the stacking of several RBM layers. Each RBM is a generative model that can be trained in unsupervised way. In [51], the authors show that a DBN can be trained from the bottom (visible layer) to the top (last hidden layer) in a layer-wise fashion, which means that the output of a layer becomes the input of the following one. After the training, the DBN becomes a more powerful generative model than the simple RBM.

A more structured description of the unsupervised training is given below:

1. the representation $h^{(l)}$ is computed ($h^{(0)} = v$ when $l = 0$);
2. the l-th layer is trained with the CD-1 algorithm explained in Section 3.3.5, and the parameters $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are updated;
3. if the training has been performed for the desired number of iterations, the procedure stops and the input of the $l+1$-layer is computed

from the hidden layer $h^{(l)}$, generally being its mean activations or, alternatively, its samples.

4. The procedure is repeated for each layer.

The main advantage of the unsupervised method is to make the DBN able to extract salient features from data, with a training performed over unlabeled datasets, which are largely available. The extracted features can be used in several tasks, since they are still rather general, but have an higher level of abstraction with respect to those obtained with one simple RBM. The overall unsupervised learning procedure is shown in Figure 3.13.

Supervised learning for a DBN

The purpose of the supervised learning is to obtain a set of network parameters that makes the DBN able to extract target-oriented features from the input data. Differently from the unsupervised approach, a labeled training set is required in the supervised approach. A cost function is defined depending on the particular task that the DBN is designed to accomplish. In a classification problem, for example, the cost function measures the distances between the estimated and the true classes, for a given labeled training set. Details about how we apply the supervised training to the music segmentation problem are given in Chapter 4.

The cost function is minimized by means of iterative numerical methods, and the Steepest Gradient Descent (SGD) presented before is commonly adopted also here. The minimum is searched moving along the function in the direction of its negative gradient, and the size of such movement, called *learning rate*, is chosen taking into account the trade-off between speed and accuracy of the convergence. A large step size speeds up the process, as it reaches more rapidly a region close to the solution, but it likely prevents the convergence to the optimum. The minimization of the cost function is

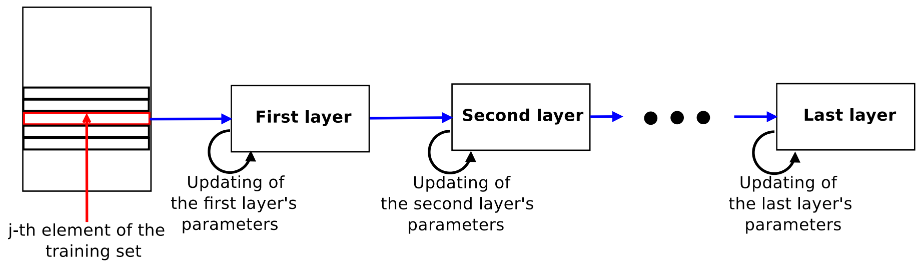

First layer
Second layer
Last layer
Updating of the first layer's parameters
Updating of the second layer's parameters
Updating of the last layer's parameters
j-th element of the training set

Figure 3.13: Diagram of the unsupervised learning

done by modifying the parameters of the network, and this process is called *fine-tuning*.

The phases of the fine-tuning are presented in the following:

1. the network parameters are initialized, either randomly or with an unsupervised pre-training phase;

2. a representation of the generic visible input $\mathbf{v}$ is computed with the current network parameters $\boldsymbol{\theta}$:

$$\begin{aligned} \mathbf{h}^{(1)} &= \mathcal{T}^{(1)}(\mathbf{v}), \\ \mathbf{h}^{(2)} &= \mathcal{T}^{(2)}(\mathbf{h}^{(1)}), \\ &\vdots \\ \mathbf{h}^{(L)} &= \mathcal{T}^{(L)}(\mathbf{h}^{(L-1)}), \end{aligned}$$

   where $\mathcal{T}^{(l)}$ is a function that computes the l-th layer representation from the previous one. $\mathcal{T}^{(l)}$ only depends on the parameters $\boldsymbol{\theta}^{(l)}$ of the corresponding layer, due to the properties of the DBN;

3. a label $\hat{y} = f(h^{(1)}, \cdots, h^{(L)})$ is estimated from the obtained representation by means of a function f, and a cost function $J(y, \hat{y})$ measures the difference between the estimated label and the true one;

4. the derivatives with respect to all the network parameters (gradient of the cost function) are computed;

5. all the network parameters are modified. The updating rule for the generic l-th layer parameter θ at the step k is given by:

$$\theta_k^{(l)} \leftarrow \theta_{k-1}^{(l)} - \rho \frac{\partial J(y, \hat{y})}{\partial \theta^{(l)}}, \qquad \forall l = 1, \cdots, L,$$

   where ρ is the learning rate;

6. the procedure is repeated from step 2, for the next training example.

The procedure can be repeated for a desired number of epochs, as for the unsupervised learning.

3.3.7 The problem of overfitting

In a supervised approach, a machine learning algorithm is trained over a set of labeled data, in order to predict its output values. However, the algorithm may be excessively trained, resulting in a poor capability of being general with unseen data. This problem is commonly referred to as *overfitting* and every machine learning algorithm employs several techniques in order to address it.

Both in the unsupervised and supervised approaches, it is possible to repeat the training for several iterations (epochs) in order to improve the power of the model. However, the more epochs are employed, the more likely the model overfits the training data. For this reason, it is common to stop the training when the model has not perfectly fit the training data, in order to be more general. Two main strategies have been proposed for this so-called "early-stop condition":

- the learning stops after a fixed number of epochs;
- the prediction performance are periodically computed over the validation set, and the learning stops when the error over the validation set starts to decrease.

An early-stopped training phase makes the model more robust to the noise present in the training set, and consequently more able to extract only relevant relationships from data.

In this work we train a DBN architecture in both unsupervised and supervised fashion, using a fixed number of epochs as early-stop condition. We successively use the trained DBN to extract a sequence of descriptors from a song. The obtained descriptors are then employed to perform the music segmentation task.

Chapter 4

System Overview

In this chapter we describe our approach to the music segmentation task. The main phases of the music segmentation procedure are the feature extraction, the boundary detection and the labeling of similar sections. The common pipeline of a music segmentation algorithm is shown in Figure 4.1.

The feature extraction phase is performed by means of a deep learning network applied to a frequency versus time representation of the audio signal. In Section 4.1, we initially describe how such representation is obtained, and then we describe the deep learning architecture used to extract relevant features from it.

Our purpose is to compare the extraction of handcrafted features (chromagram and MFCC, in particular) with the features obtained with a deep learning approach. We test our features with several music segmentation algorithms proposed in the literature.

4.1 Feature Extraction

In the feature extraction phase, a sequence of suitable descriptors are extracted from the audio signal. Initially, we compute the spectrogram, which is a frame-wise representation of the audio signal in the frequency domain.

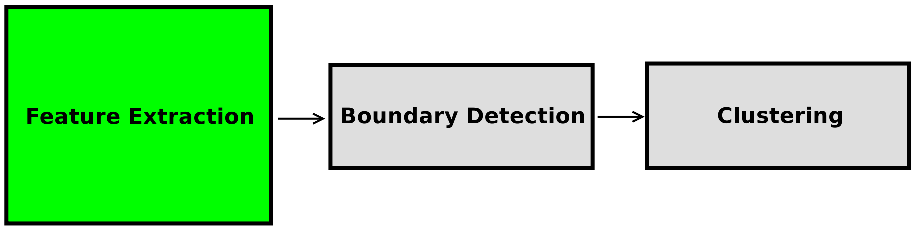


Figure 4.1: The main phases of the music segmentation task

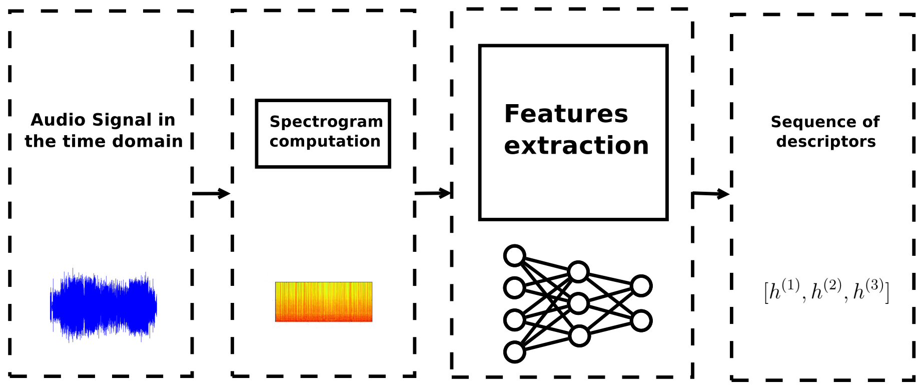


Figure 4.2: Pipeline of the feature extraction phase

Each frame is then used as visible layer of a DBN in order to extract its salient features. An illustration of the process is given in Figure 4.2.

In Section 4.1.1 we describe the parameters that characterize the spectrogram, and we show how they affect the quality of the music segmentation task. In Section 4.1.2 we discuss the training of the DBN, both the unsupervised and the supervised ones.

4.1.1 Input processing

In this section we describe the possible inputs of the DBN, aiming at learning which is the best set of parameters in the computation of the spectrogram. Then, we use such information to compute the performance with all the available algorithms. We refer to Chapter 5 for the relative considerations.

Initially we convert the audio signal from the stereo to the mono format, by means of a simple average applied to the two channels. The audio signal is characterized by its sampling frequency F_s, which is related to the maximum frequency F_{max} that can be preserved in the conversion between the analog and the digital domain. The Shannon sampling theorem, in fact, states that $F_{max} = F_s/2$.

Since in general $F_s = 44100Hz$, the maximum preserved frequency is $F_{max} = 22050Hz$. Since the components relative to the high frequencies may not be as relevant as the lower ones for the music segmentation task [20], we modify F_{max} by reducing F_s to $11025Hz$. Such operation, which is called *downsampling*, presents the advantage of substantially reducing the computational effort. We compute the performance of the system, using both $F_s = 44100Hz$ and $F_s = 11025Hz$, and we show their qualitative comparison in Section 5.4.1.

From the (downsampled) audio signal, the STFT is computed. The parameters that characterize it are:

- *Window function*: it is the function used in the windowing process;
- *Frame length*: it is the number of samples per frame (we refer to the frame length as N_{stft});
- *Hop-size*: it is the shift size of the windowing process, in samples.

The window function is chosen according to the desired refinement of the spectrum [56]. Typical choices are the *rectangular* and the *Hamming* windows. The spectrograms computed with the former window are rather coarse, while those computed with the latter present an higher frequency resolution [56].

The frame length and the hop-size determine the degree of temporal dynamic (i.e., the evolution of the musical properties along the time) captured by the descriptors, as well as the frequency resolution of the frames, due to the trade-off between temporal and frequency resolution described in Section 3.1.1. We try two combinations of these parameters. In the first case the frame length and the hop-size coincide, while in the second one the frame length is larger than the hop-size, so that the frames overlap and the features can capture both the global and the local properties of the audio signal.

After the computation of the STFT, the spectrogram of the song is derived, and it is represented by a matrix $\mathbf{S} \in \mathbb{R}^{N_s \times \frac{N_{stft}}{2}+1}$, where N_s is the number of frames of the generic s-th song. All the values of the matrix are then converted in dB ($\hat{\mathbf{S}} = 20 \log_{10}(\mathbf{S})$) to resemble the human auditory system, and normalized between 0 and 1. The purpose of the normalization is to discard the information on the power of the signal, which may strongly influence the training phase.

4.1.2 Deep Learning Network

4.1.2.1 Unsupervised Learning

The unsupervised learning aims at discovering the structure of the training data. In this work we use the DBN that is able, thanks to its layer-wise architecture, to capture the hidden relations among the input data with an high level of abstraction. After the unsupervised training, the network is able to extract a general (i.e., not target-oriented) representation of unseen data.

Training dataset creation

The songs used to train the DBN are chosen from the available databases. The training dataset is then formed as a list of spectrograms, one for each training song.

The music segmentation task produces an high-level representation of a song (the division in its sections and the labeling) and, as a consequence, the labels related to the raw data are song-dependent. However, as explained in Section 3.3, the unsupervised learning does not require a labeled set, and the representations of all the songs can be merged. Thus, all the spectrograms, which are frame-wise representations of the songs, are stacked to form the overall dataset. The result is a matrix of dimensions $N_{TOT} \times N_{stft}$, where N_{TOT} is the total number of frames.

Deep Learning architecture

The deep learning architecture that we use is a Deep Belief Network, which is characterized by:

- M visible inputs;
- L: the number of DBN layers (i.e., the depth of the network);
- $H^{(l)}$: the number of nodes of the l-th layer, $\qquad l = 1, ..., L.$

The dimension of the network is a crucial point of the learning process. Given the depth of the network, a DBN with few nodes per layer requires less computational effort than one with a larger number of nodes per layer, but may also not be able to capture relevant features from data. We test several network topologies, and we refer to Chapter 5 for the relative considerations.

The parameters that concern the training are:

- *pre-learning rate*: it is the learning rate of the Steepest Descent algorithm;
- *number of epochs* N_{ep}: it is the number of iterations of the learning phase.

In order to avoid over-fitting (see Section 3.3.7), we use a fixed number of epochs N_{ep} as early-stop condition.

After the initialization of its parameters, the DBN is trained in an unsupervised layer-wise fashion, with the CD-1 algorithm explained in Section 3.3.6. At the end of the unsupervised learning phase, the system is able to capture a rather general representation of data.

4.1.2.2 Supervised Learning

The supervised learning aims at finding a relation between input data and corresponding label. In our method the data are all the possible pairs of frames of the song, and the labels are measures of their similarity. In this work we train a DBN in supervised fashion after an unsupervised pre-training phase.

Training dataset creation

The music segmentation algorithms are often based on the comparison of all the pairs of frames. Thus, we define the supervised training set in such a way that it contains the information of the similarity for each couple of frames. More specifically, the supervised training set is formed as a set of elements, each one composed by the spectrogram and the ideal SSM of a song: the former is represented by a matrix of dimension $N_K \times N_{stft}$, where N_K is the number of frames. The latter is represented by a matrix of dimension $N_K \times N_K$, whose generic entry (i, j) is equal to 0 if the i-th and the j-th frames belong to the same structural part (i.e., they are similar), and it is equal to 1 otherwise. Notice that such matrix is the ideal SSM described in Section 3.2.1.

Cost function definition and training

A crucial point of the supervised learning procedure concerns the set-up of an adequate fine-tuning strategy. Since several music segmentation algorithms are based on the computation of the SSM, we propose two types of fine-tuning, aiming to obtain a set of features from which it is possible to derive the ideal SSM. The first one is based on the computation of the euclidean distance between the pairs of descriptors, whereas the second one is based on a classification layer placed at the top of a deep learning network.

The fine-tuning is realized by means of the Steepest Descent algorithm described in Section 3.12, which minimizes a cost function through the iterative updating of the network weights. As far as the training parameters are concerned, there are some rather formal differences with respect to those of the unsupervised learning. In particular:

- the pre-training learning rate is replaced by the *learning rate*;
- the minimization is repeated for a fixed number of epochs, which may be different from that used in the unsupervised pre-training.

The cost function that we adopt is:

$$J(y, \hat{y}) = (y - \hat{y})^2, \tag{4.1}$$

where y is the reference label, whereas $\hat{y}$ is the estimated one. In order to obtain $\hat{y}$, we initially train a DBN in unsupervised fashion. A new deep learning architecture is successively built by duplicating the DBN, and the vector $\mathbf{u_{ij}}$, which encodes the similarity between the i-th and the j-th frames, is computed. Its generic k-th entry is:

$$u_{ij}(k) = h_i^{(L)}(k) - h_j^{(L)}(k), \tag{4.2}$$

where $h_i^{(L)}(k)$ and $h_j^{(L)}(k)$ are the k-th components of the last layer representations of the i-th and j-th frames, respectively. From the obtained vector $\mathbf{u_{ij}}$, the label is predicted according to the two different fine-tuning techniques that we propose.

The first one computes the estimated label as:

$$\hat{y}_{ij} = \sqrt{\sum_{k=1}^{H(L)} u_{ij}^2(k)}. \tag{4.3}$$

$\hat{y}_{ij}$ is then normalized between 0 and 1. Notice that such label is the normalized version of the euclidean distance computed between the descriptors. An illustration of the process is depicted in Figure 4.3.

The second one computes the estimated label as:

$$\hat{y}_{ij} = \mathrm{sigm}(\mathbf{w_c} \mathbf{v_{ij}} + \mathbf{b_c}), \tag{4.4}$$

where $\mathbf{w_c}$ and b_c are the weight vector and the bias (which is a scalar), respectively, of a one-node classification layer that is added on the top of the new network. In this second configuration, $\hat{y}$ is the value assumed by the classification node, and it is a measure of the dissimilarity between the two input frames (i.e., the value is 0 when they are similar). Figure 4.4 shows the overall procedure.

The SSM is usually obtained through the computation of the distance between each couple of extracted descriptors, as explained in Section 3.2.1. The second finetuning strategy allows also to compute the SSM by exploiting the classification layer. The predicted value, in fact, can be seen as the SSM entry associated with the pair of frames given as input. Thus, in order to create the SSM, the finetuning based on the classification layer also tries to model a similarity function.

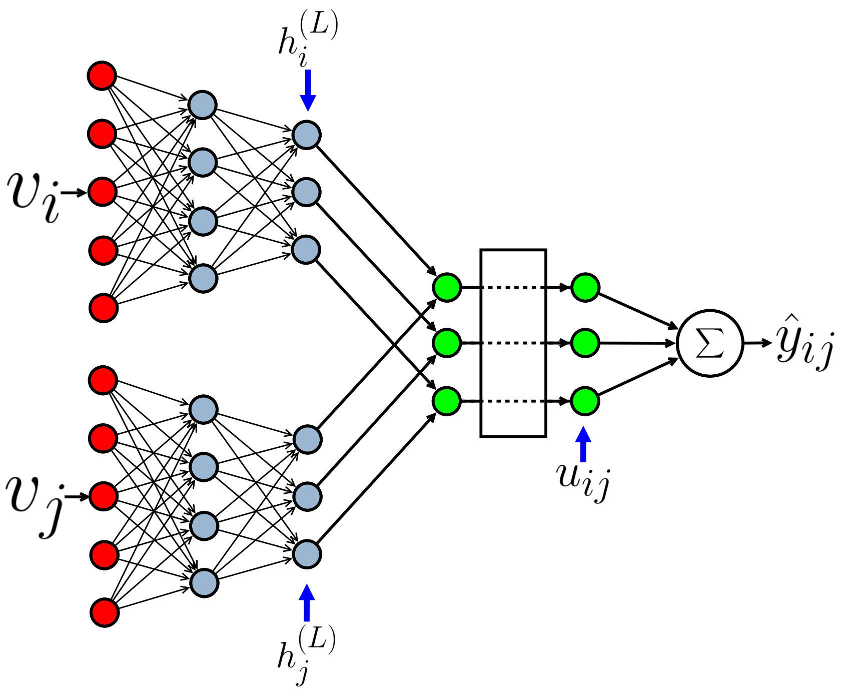


Figure 4.3: Architecture defined to estimate the similarity between two input frames for the finetuning based on the euclidean distance

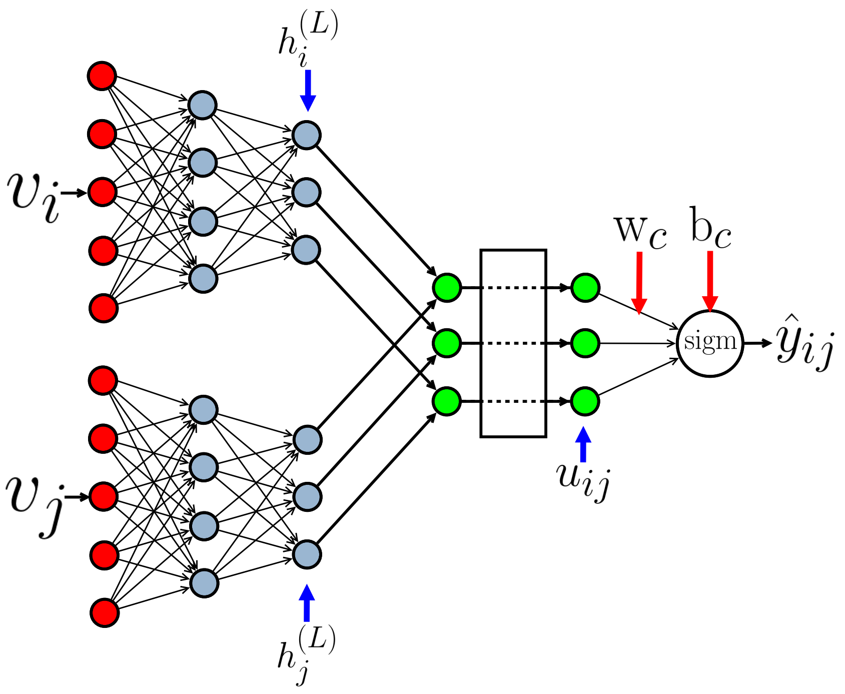


Figure 4.4: Architecture defined to estimate the similarity between two input frames for the finetuning based on the classification layer

4.2 Music Segmentation

The music segmentation task is divided in the two subtasks of boundary detection and clustering. The boundary detection is the operation that spots the boundaries of sections within the song, i.e., the time instants when a section ends and the following begins. The clustering is the operation that groups together the sections that belongs to the same structural part. The music segmentation does not perform a classification, so that the assigned labels do not have a semantics (like *Chorus* or *Verse*).

The clustering is not always performed after the boundary detection. In some cases, for example, the boundary positions are derived from the cluster assignments. In this work we use the DBN-based features with four state-of-the-art music segmentation algorithms: Foote [49], Structural Features (SF) [19], C-NMF [20] and Spectral Clustering (SC) [57](see Table 4.1).

4.2.1 Foote

The algorithm proposed by Foote in [49] is a novelty-based music segmentation algorithm, which works on the Self Similarity Matrix. The algorithm performs only the boundary detection task.

The main idea of this algorithm consists in performing a correlation of a kernel along the main diagonal of the Self Similarity Matrix computed over a suitable representation of the song. The result of the correlation is the novelty curve. The procedure has been explained in Section 3.2.2, as it represents a common way to compute the novelty curve. The kernel is designed in such a way that the correlation takes higher values in the proximity of a boundary. In the algorithm described in [49] the kernel is the Gaussian one. Once the novelty curve has been computed and properly filtered, the peaks are searched in it.

The advantage of the Foote's algorithm lies in its simplicity, but in contrast, the performance are very poor. The inefficiency is mostly due to the way the novelty curve is computed. The result, in fact, is a rather noisy

	Boundary Detection	**Clustering**
Foote	Yes	No
Structural Features	Yes	No
C-NMF	Yes	Yes
Spectral Clustering	Yes	Yes

Table 4.1: Summary of the tasks performed by the algorithms

curve, which prevents an effective peak picking phase. In particular, we have noticed in our experiments that the algorithm presents oversegmentation. We refer to Chapter 5 for a quantitative evaluation of the problem.

4.2.2 Structure Features

SF, which has been proposed by Joan Serrà et al. in [19], is a novelty-based algorithm that works on the time-lag matrix, which is a modified version of the Self-Similarity Matrix, to perform only the boundary detection phase.

In order to derive the SSM, the feature sequence is initially modified in such a way that it includes some local temporal dynamic. This operation is performed by replacing each descriptor of the sequence with the concatenation of its adjacent descriptors. All the elements of the sequence are successively pairwise compared and a square matrix is derived. The values assumed by the entries of such matrix are obtained by means of the *K-nearest neighbors*, which is an unsupervised machine learning algorithm. In particular, the generic entry of such matrix assumes the value of 1 if and only if the corresponding frames are K-nearest neighbors, whereas the value is 0 otherwise. Such matrix is then converted to its time-lag version. The time-lag is a matrix where the rows represent the frame indexes, while the columns represent the shift, in number of frames, with respect to the index of the row. The following equations aim to clarify the relation between SSM and its corresponding time-lag matrix:

$$\mathbf{SSM}(i,j) = D(\mathbf{x_i}, \mathbf{x_j}),$$

$$\mathbf{TimeLag}(i,k) = D(\mathbf{x_i}, \mathbf{x_{i+k}}),$$

where D is a distance function, whereas $\mathbf{x_i}, \mathbf{x_j}$ and $\mathbf{x_{i+k}}$ are generic frame descriptors. k is the shift, i.e., the *lag*.

The resulting matrix can be seen as a sequence of descriptors, from which the novelty curve is computed. The boundaries are successively extracted from the curve, according to a thresholding mechanism: a peak of the curve is considered a boundary if it is a local maximum and if it is above a, generally not constant, threshold.

4.2.3 C-NMF

The C-NMF music segmentation algorithm, which has been proposed by Nieto in [20], exploits the capabilities of the Convex Non-Negative Matrix

Factorization (C-NMF) to yield a part-based representation of the Self-Similarity Matrix. Such representation is then used to obtain the structure of a song.

The SSM is computed over a sequence of descriptors by means of the correlation distance (see Section 3.2.1), and each element $SSM(i,j)$ is replaced by a value equal to $2^{SSM(i,j)}$, successively normalized between 0 and 1, and then factorized. Roughly speaking, a factorization technique allows to find the approximation of a generic matrix $\mathbf{X}$, which can be written as:

$$\mathbf{X} \approx \mathbf{FG},$$

where $\mathbf{X} \in \mathbb{R}^{N \times M}$ is the matrix to approximate, and $\mathbf{F} \in \mathbb{F}^{N \times R}$ and $\mathbf{G} \in \mathbb{R}^{R \times M}$ are the decomposition matrices, with $R \in \mathbb{N}$ being the rank of the decomposition. After the decomposition, each element of $\mathbf{X}$ can be written as:

$$\mathbf{X}(i,j) \approx \sum_{k=1}^{R} \mathbf{A}_k(i,j), \tag{4.5}$$

where $\mathbf{A}_k = \mathbf{F_k G_k^\top}$, with $\mathbf{F_k}$ and $\mathbf{G_k^\top}$ being the k-th column of the matrix $\mathbf{F}$ and the k-th row of the matrix $\mathbf{G}$, respectively.

The Convex-Non Negative Matrix Factorization (C-NMF) is applied to the SSM, such that $\mathbf{SSM} \approx \mathbf{FG}$, where $\mathbf{F} \in \mathbb{R}^{N \times R}$ and $\mathbf{G} \in \mathbb{R}^{R \times N}$. The C-NMF is the evolution of the Non Negative Matrix Factorization (NMF) proposed in [14], which forces the decomposition matrices to be positive. Due to the properties of the NMF, each addend of the summation presented above is expected to encode the information of a structural part, which means that a part-based representation of the initial matrix can been obtained. Moreover, the convexity constraint is fulfilled, i.e., each column of $\mathbf{F}$ is a linear combination of the columns of $\mathbf{X}$, where the weights of the linear combination sum to one. For a detailed description of the C-NMF algorithm we refer to [58].

After the factorization, the SSM is approximated by the summation of R matrices, each one encoding the information of a structural part. The k-means clustering algorithm is applied to the rows of each of these matrix, with k = 2. A boundary is estimated when two successive frames are classified as belonging to different classes. Once the potential boundaries for all the matrices have been obtained, they are combined within a distance window, so that the close boundaries are merged in their average location.

Thanks to the part-based representation of the initial matrix, it is possible to know how the energy of each previously found section is spread across all the structural parts, and to describe the sections in a feature space. All the sections are then compared with the Euclidean distance, and classified as belonging to one of the structural parts. The drawback of the algorithm is that the number of clusters has to be previously estimated to obtain the best performance.

4.2.4 Spectral Clustering

The algorithm, proposed by Brian McFee et al. in [57], associates the representation of a song with a graph, in which the nodes represent the frames and the edges measure their similarity. Spectral clustering techniques are then used to find the structure of the song.

SC uses both the chroma and the MFCC as descriptors. The former is used to detect similarities between distant frames, while the latter is used to detect local consistency (i.e., two near frames are generally characterized by the same timbre). In order to do a comparison with the traditional features, we replace both the chroma and the MFCC with the DBN-based descriptors.

All the N elements of the feature sequence are compared by means of the *K-nearest neighbors* algorithm (see Section 4.2.2), and a binary matrix $\mathbf{R} \in \{0, 1\}^{N \times N}$ is obtained. Through the application of signal processing techniques, $\mathbf{R}$ is enhanced to obtain an adjacency matrix, i.e., a matrix in which the ij-th entry is 1 if and only if the i-th and the j-th nodes of the graph associated with the song are connected.

The adjacency matrix is successively modified in order to better describe the relations between pair of nodes (i.e., pair of frames), and the matrix $\hat{\mathbf{R}}$ is formed. In particular, $\hat{\mathbf{R}}$ is not binary anymore, since the values of the edges become weighted in order to better model the relations among frames. The process performed to weigh the edges is specifically designed to work with both the chroma and the MFCC. Thus, among the algorithms that we use, SC is the most dependent on the traditional features.

The k-means clustering algorithm is then applied to the eigenvectors of $\hat{\mathbf{R}}$, where k is set to be equal to M, i.e., the number of clusters. The value of M is estimated according to the desired level of details. Since in the music segmentation task it is required a detail at the level of frames, M is chosen as the value that minimizes a cost function related with the frame-level annotation entropy (we refer to [57] for the details). Clustering and boundary detection are jointly performed, since the boundaries are simply estimated as being the changing points of the cluster assignment.

Chapter 5

Experimental Results

In this chapter we introduce and discuss the evaluation metrics and the results. In Section 5.1 we show in details the main metrics used to evaluate a music segmentation algorithm. Then, we describe the datasets that we used to train the DBNs and to test the performance (see Section 5.2), as well as the tools that we exploited to implement our system (see Section 5.3). In Section 5.4 we discuss the parameters used in the features extraction phase. Then, in Section 5.5 we present the numerical results obtained with the traditional and with the DBN-based descriptors, both the unsupervised and the supervised ones. Finally, in Section 5.6 we present some final considerations about the obtained results.

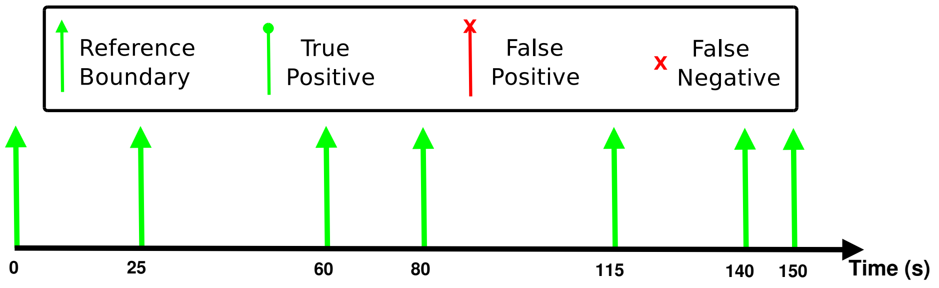
5.1 Evaluation metrics

In this section we show the metrics that are commonly used to evaluate the performance of music segmentation systems. There are two families of metrics: the first one evaluates the boundary detection, whereas the second one evaluates the clustering.

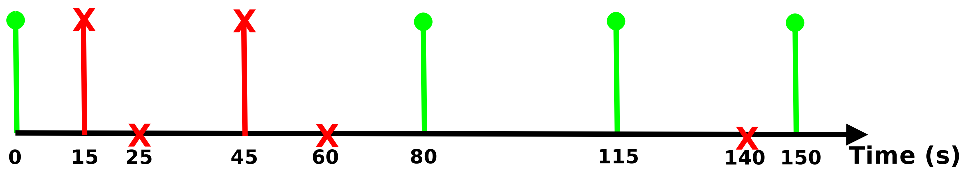
5.1.1 Metrics for the boundary detection evaluation

As far as the boundary detection is concerned, the similarity between estimated boundaries and the ground truth is evaluated with metrics derived from the *Hit Rate* [59], which measures the performance by considering an estimated boundary as correctly retrieved if it is within a certain interval of time from the correspondent boundary in the ground truth.

In order to define all the metric related with the Hit-Rate, it is necessary to provide some definitions:



(a) Ground truth



(b) Estimated boundaries

Figure 5.1: An example to understand the retrieval metrics

- a correctly retrieved boundary is a *True Positive*;
- a wrongly retrieved boundary is a *False Positive*;
- a not detected boundary is *False Negative.*

The example shown in Figure 5.1 can help to clarify the concepts of True Positive, False Positive and False Negative. Two measures can then be defined:

$$Precision = \frac{|TruePositives|}{|TruePositives + FalsePositives|};$$

$$Recall = \frac{|TruePositives|}{|TruePositives + FalseNegatives|},$$

where $|\cdot|$ indicate the cardinality of a set.

The Precision is the ratio of the correctly retrieved boundaries over the total number of detected boundaries, whereas the Recall is the ratio of the correctly retrieved boundaries over the total number of the ground-truth boundaries. Precision and Recall assume values within the range $[0, 1]$.

Since there is a trade-off between Precision and Recall, the Hit-Rate F-measure has been introduced as a balancing metric. The F-measure is defined as:

$$F_\alpha = (1+\alpha^2)\frac{Precision \cdot Recall}{\alpha^2 \cdot Precision + Recall},$$

where $\alpha \in [0, \infty]$ is a parameter chosen in accordance with the relevance given to Precision or Recall. When $\alpha \in [0, 1]$ more relevance is given to the Precision, while when $\alpha \in [0, \infty]$ more relevance is given to the Recall. In [60] it is shown that an high value of Precision is perceptually more relevant than an high value of Recall for the evaluation of segmentation algorithms. Although in [60] it has been proven that $\alpha = 0.58$ is the value that properly balances the perceptual evaluation of Precision and Recall, in this work we set $\alpha = 1$, since it is the commonly adopted choice in literature. Notice that, in the case of perfect matching, the Hit-Rate F-measure assumes the value of 1.

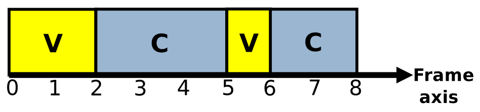
The Hit-Rate metrics are the F-measure, the Precision and the Recall computed with an interval of tolerance of $\pm 0.5s$ and $\pm 3s$ with respect to the boundaries of the ground-truth. These intervals are commonly adopted in the literature. The performance within the $3s$ interval are always higher (or, at least equal) than those within the $0.5s$ interval. In fact, if a boundary is correctly retrieved within a certain range, it is also correctly retrieved within a larger one.

5.1.2 Metrics for the clustering evaluation

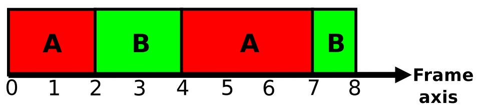
A cluster is a set of sections that belong to the same structural part. In order to evaluate the performance, two different structures are compared: the ground-truth one and the estimated one. The former is usually characterized by semantic labels (such as Chorus and Verse), whereas the latter is characterized by non-semantic labels (since music segmentation is not a classification task). An example of clustering is shown in Figure 5.2, in which both the ground-truth structure (see Figure 5.2a) and the estimated one (see Figure 5.2b) are presented.

The clustering metrics are designed in such a way to allow the comparison between the two structures, even though their labels are of different kinds. In order to define the clustering metrics, two sets are introduced. The set F_a contains all the pair of frames that belong to the same structural part in the ground-truth. The set F_e contains all the pair of frames that belong to the same structural part in the estimation. The more similar the two sets are, the better is the clustering algorithm.

The sets F_a and F_e can be represented in a 2D structure, which collects every pair of frames and assigns the value of 1 when the two frames are considered belonging to the same structural part, and the value of 0 otherwise.



(a) Expected clustering



(b) Estimated clustering

Figure 5.2: An example of expected and estimated clustering

In Figure 5.3, we show a 2D representation of the ground-truth and the estimated structures. Figure 5.3a is the 2D representation of the structure shown in Figure 5.2a, whereas Figure 5.3b is the 2D representation of the structure shown in Figure 5.2b. The grey regions assume the value of 0, whereas the other regions assume the value of 1 (we use the labels only to show when the sections belong to the same structural part).

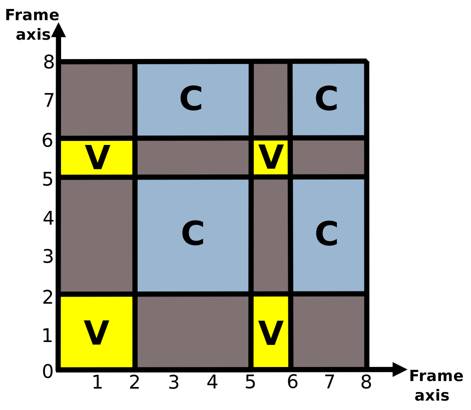
From the sets F_a and F_e, we define the clustering metrics, which are the pairwise Precision (PP), the pairwise Recall (PR) and the pairwise F-measure (PF):

$$PP = \frac{\mid F_a \cap F_e \mid}{\mid F_e \mid},$$

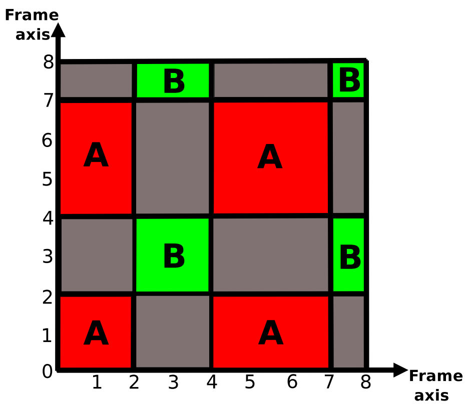
$$PR = \frac{\mid F_e \cap F_a \mid}{\mid F_a \mid},$$

$$PF_\alpha = 2\alpha \cdot \frac{PP \cdot PR}{PP + PR}.$$

The set $\mid F_a \cap F_e \mid$ contains all the pairs of frames that belong to the same structural part both in the ground-truth and in the estimations. A graphical representation of this intersection is presented in Figure 5.4. As for their boundary counterparts, also PP, PR and PF assume a value in the range $[0, 1]$, where 1 is assigned to PF in case of perfect matching between estimation and reference.



(a) 2D representation of the reference structure. The set F_a is represented by the non-grey regions



(b) 2D representation of the estimated structure. The set F_e is represented by the non-grey regions

Figure 5.3: 2D representation of expected and estimated clustering

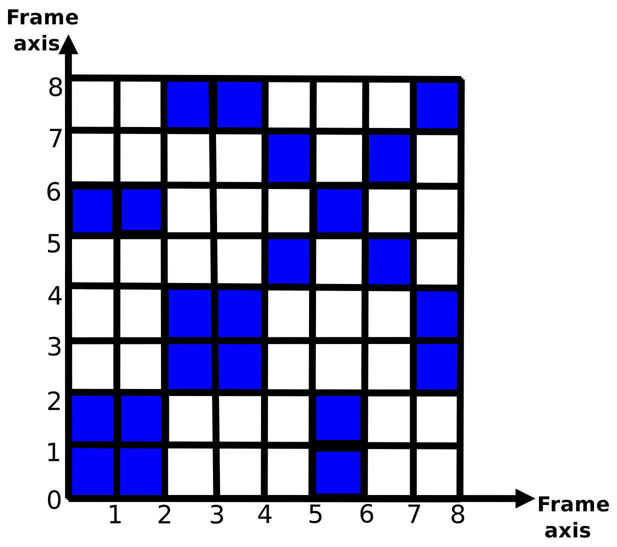


Figure 5.4: 2D representation of the set $\mid F_a \cap F_e \mid$

5.2 The datasets

In order to evaluate our approach, we made use of different datasets well known in the literature, which are composed by a set of music pieces and the correspondent annotation for the structural analysis. The annotations include the positions of the boundaries and the semantic labels of the associated sections, for each song of the dataset.

In order to avoid that the training conditions the test, we used two different datasets for the training and for the test phases. In particular, we trained the DBNs using the *SALAMI* database, while the performance of the system were computed over the *Beatles* database.

The SALAMI dataset

The *Structural Analysis of Large Amounts of Musical Information* database (SALAMI) [61] includes songs of various genres, like pop, jazz, classical and others. The SALAMI annotations have been provided by experts in Music Theory and Composition [61].

Since the database is not completely available on-line, we performed the training of our system with a part of it, which is composed of 267 songs. The largest part of the songs has a sampling frequency $F_s = 44100Hz$, but also other sampling frequencies are present ($48000Hz$, $32000Hz$, $22050Hz$). In our experiments, we re-sampled all the songs with the same sampling frequency (see Section 5.4). Only 4 songs are in the *mono* format, while all the others, which are in the *stereo* format, are converted to mono as explained in Section 4.1.1.

The Beatles dataset

The Beatles dataset is composed by 174 songs from the discography of the Beatles. The performance are usually evaluated over two sets of annotations, one provided by the Queen Mary University of London[1], and the other one provided by the Tampere University of Technology[2]. Since the second database is the most commonly used in literature, we adopted it for the test. All the songs in the dataset are characterized by a sampling frequency $F_s = 44100Hz$, and all are in the *stereo* format, which have been converted to mono. We present the list of all the songs by the Beatles in Appendix A.

5.3 Details of the implementation

In this section we describe the tools that we used to implement our music segmentation system. We initially describe our implementation of a Deep Belief Network, which is used to extract the descriptors from the audio signal. Then, we describe the music segmentation framework that we used to test the performance of our features.

[1]`http://www.cs.tut.fi/sgn/arg/paulus/beatles_sections_TUT.zip`
[2]`http://isophonics.net/content/reference_annotations`

Deep learning architecture

We implemented the deep learning architectures using the Python programming language [62]. The DBN is a Python class, which is based on a list of RBM classes. Some Python libraries have been particularly important in the development of this work: Numpy and Scipy [63], which provide powerful numerical computing and signal processing tools, and Theano [64], which allows to define symbolical expressions.

We extracted the DBN-based descriptors with 5 DBN architectures, all with a depth of 3 layers, and the 5 DBNs differ in the number of nodes (see Table 5.1). We tried different combinations of nodes, in order to understand how the number of nodes per layer influences the ability to extract relevant features from the audio signals.

All the networks presented above have been trained in unsupervised fashion as described in Section 3.3.6. The unsupervised training phase has been performed over all the SALAMI dataset, and it is characterized by the following parameters: *pre-learning rate* $= 10^{-5}$ and $N_{ep} = 5$. In a second stage the DBN has also been trained in supervised fashion, using only the smallest network, i.e., the DBN(75,50,25) in Table 5.1, since the fine-tuning is a computationally-expensive operation whose complexity depends on the number of nodes of the network and on the number of elements in the training set. More specifically, if N is the number of songs of a dataset and $\overline{N}_s$ is the mean number of frames per each song, the unsupervised training dataset is composed by $N \cdot \overline{N}_s$ frames, whereas the supervised training set is composed by $\approx N \cdot \overline{N}_s^{\,2}$ frames, because it is formed by all the pairs of frames for each song.

Since $\overline{N}_s$ is generally $\approx 10^3$, fewer songs are required to make the supervised training set as large as the unsupervised one. Thus, we performed the fine-tuning using only 20 songs out of the 267 of the entire SALAMI

Network Architecture	$\mathbf{H^{(1)}}$	$\mathbf{H^{(2)}}$	$\mathbf{H^{(3)}}$
DBN(1000,250,25)	1000	250	25
DBN(100,250,25)	100	250	25
DBN(500,250,200)	500	250	200
DBN(500,750,25)	500	750	25
DBN(75,50,25)	75	50	25

Table 5.1: List of employed DBNs, where $H^{(i)}$ is the number of nodes at the i-th layer of the DBN

database. The supervised training is characterized by the following parameters: *learning rate* $= 10^{-4}$ and $N_{ep} = 1$.

Music segmentation framework

In Section 4.2 we described the algorithms used to evaluate the performance of our system. Those algorithms are present in the literature, and collected in a framework[3] developed by Oriel Nieto. There is a difference between the description of the C-NMF algorithm (see Section 4.2.3) and its implementation in the framework. The factorization is performed on the matrix representing the feature sequence, instead of on the SSM. In fact, from our preliminary experiments, we notice that this choice guarantees higher performance.

The framework allows to compute a representation based on three type of features: the chromagram, the MFCC (described in Section 3.1) and the Tonnetz (a descriptor that captures the tonal content of the song). Since the latest is not frequently used in literature, we did not use it for the evaluations.

We have modified the framework in order to include our feature extraction phase, which allows to obtain the DBN-based features, i.e., the representation derived from the DBN (its hidden layers, either taken separately or jointly). The results are computed using the MirEval Python library [65].

5.4 Experimental setup of the feature extraction phase

In this section we aim at investigating how the hyper-parameters of the feature extraction phase affect the final performance. Such parameters are: the sampling frequency, the length of the frames, the hop-size and the windowing function. The sampling frequency is related with the maximum frequency components that is preserved, while all the other parameters are related with a degree of frequency resolution and of temporal dynamic. Thus, the main aspects that we investigate are: which frequencies are the most important when performing music segmentation, and which are the most advisable time/frequency resolution and temporal dynamic. Notice that the parameters may not be independent, and an exhaustive search would require to try all their possible combinations. However, this is a computationally-expansive operation, and we consequently approximate the space parameters

[3]`https://github.com/urinieto/msaf`

Table 5.2: Parameters of the experimental setup

Parameter	Value
Sampling Frequency	$11025Hz$; $44100Hz$
Frame Length	$\approx 50ms$; $\approx 190ms$
Hop-size (ms)	$\approx 50ms$; $\approx 190ms$
Windowing function	Rectangular; Hamming

by choosing the best parameter at each step, in an heuristic fashion. The parameters that we try and their value are shown in Table 5.2.

We compute the performance using only two algorithms: SF for the boundary detection, and C-NMF for the clustering. In fact, in this phase we only want to obtain a good set of parameters, and then use it to compute the results with all the available algorithms (see Section 5.5.1).

We present some test-cases, each one aiming to show a comparison of the performance when a single parameter of the feature extraction phase is changed. The comparison is presented by means of histograms that show the average performance of three metrics, i.e., the F-measure $0.5s$, the F-measure $3s$ and the Pairwise F-measure. The average is computed over the 5 DBNs trained in unsupervised fashion only, using as descriptor the concatenation of the three DBN layers.

5.4.1 Sampling Frequency

In this experiment we show the impact of the sampling frequency F_s on the performance. The sampling frequencies that we tested are $F_s = 44100Hz$, since it characterizes the largest number of songs, and $F_s = 11025Hz$, which is commonly adopted in the literature. The fixed parameters are:

- rectangular windowing function;
- frame length $\approx 50ms$;
- hop-size $\approx 50ms$.

The Figure 5.5 clearly shows that the performance are always higher when the downsampling is performed. This confirms the assumption described in 4.1.1, for which the components relative to the high frequencies may be not as important as the low ones for the music segmentation task. Moreover, they can be affected by noise from encoding and instruments. For this reason, since the sampling frequency has to be a fixed parameter in the following experiments, from now on we always compute the performance with $F_s = 11025Hz$.

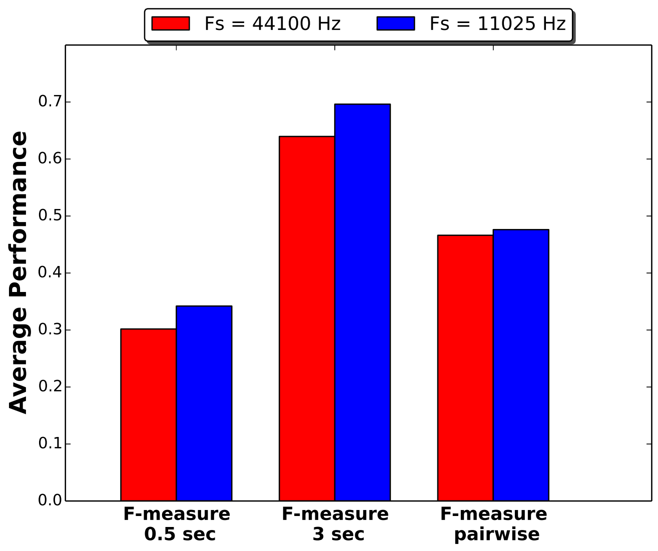


Figure 5.5: Histogram of the performance for varying sampling frequency

5.4.2 Windowing function

In this experiment we understand the influence of the windowing function on the average performance. We adopt the rectangular and the Hamming windows, since they differ in the frequency resolution of the resulting representation. The fixed parameters are:

- frame length $\approx 190ms$;
- hop-size $\approx 190ms$;
- $F_s = 11025Hz$.

Differently from the previous test-case, we use a larger frame length to increase the frequency resolution. The Hamming window guarantees a finer representation (i.e., higher frequency resolution) than the rectangular one. From Figure 5.6 it is possible to notice that the performance computed with the Hamming function are always lower than those obtained with the rectangular one. Thus, we deduce that a coarse representation is more advisable when performing music segmentation.

5.4.3 Frame length

In this experiment we show the impact of the frame length on the performance. We choose the sizes that are commonly adopted in the literature,

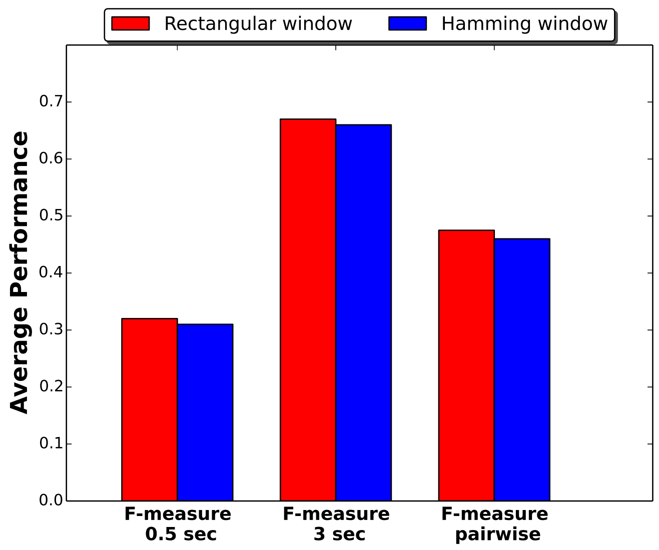


Figure 5.6: Histogram of the performance for varying windowing function

i.e., $\approx 50ms$ and $\approx 190ms$. With the second value we obtain both an higher frequency resolution and an higher temporal dynamic. The fixed parameters are:

- rectangular windowing function;
- hop-size $\approx 50ms$;
- $F_s = 11025Hz$.

In Figure 5.7, differently from the previous test-case, there is not an evident difference in the performance. The results may suggest that the properties that depend on the frame length, i.e., the temporal dynamic (see Section 4.1.1) and the frequency and time resolution (see Section 3.1.1), are not decisive when performing music segmentation. However, it is also possible that the obtained descriptors are strongly influenced by the hop-size, which is still fixed. In particular, the time resolution that is lost when the frame length is higher, is somehow preserved by the short hop-size. Thus, in order to understand the real influence of the hop-size, in the following experiment we make it varying, while the frame length is kept constant.

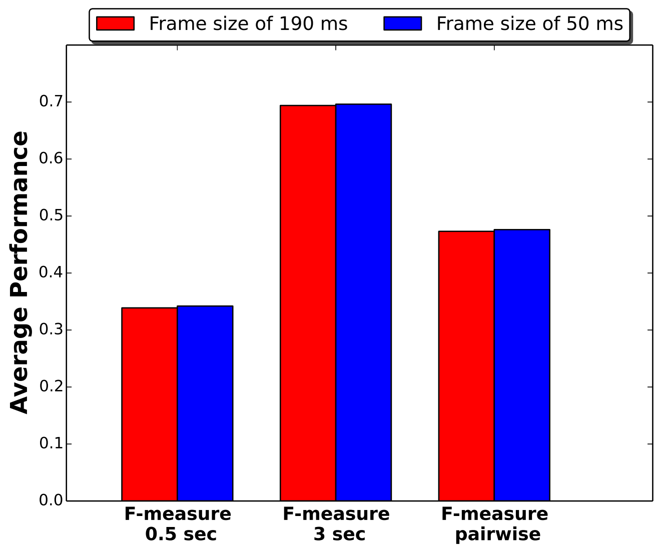


Figure 5.7: Histogram of the performance for varying frame length

5.4.4 Hop-size

In this experiment we show the impact of the hop-size on the performance. We choose the hop-sizes of $\approx 50ms$ and $\approx 190ms$, as commonly done in the literature. The fixed parameters are:

- rectangular windowing function;
- frame length $\approx 190ms$;
- $F_s = 11025Hz$.

Figure 5.8 shows that the choice of the hop-size is an important aspect of the feature extraction phase. In particular, for the boundary detection the performance are higher when the hop-size is $\approx 50ms$ (i.e., when the frames overlap). Since the variation of the hop-size (as well as the frame length) influences the time resolution of the obtained representation, we deduce that an higher time precision (guaranteed by a short hop-size) has a positive effect on the boundary detection phase. Moreover, we can confirm the conclusion drawn in the previous section, i.e., that the time resolution also depends on the hop-size. As far as the labeling is concerned, we do not notice any relevant difference.

The quantitative results for each one of the presented test-cases are provided in Appendix B, where it is possible to notice that the best set of parameters is: sampling frequency $F_s = 11025Hz$, frame length $\approx 50ms$,

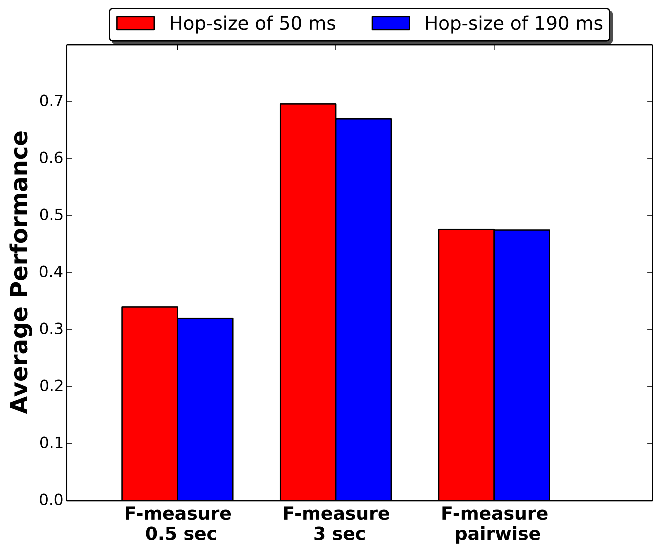


Figure 5.8: Histogram of the performance for varying hop-size

hop-size $\approx 50ms$ and rectangular window. Thus, we use this setting to compute the performance for all the algorithms. From the above experiments we can deduce that the components relative to the higher frequencies should be discarded and that a coarse representation (i.e., high frequency resolution) is preferable than a finer one. Moreover, the temporal dynamic is the parameter that less affect the performance.

5.5 Numerical Results

In this section we present a quantitative evaluation of the performance, computed with all the algorithms described in Section 4.2. In this way we can make a comparison between the traditional features (chroma and MFCC) and the DBN-based descriptors.

The results shown in Section 5.5.1 are computed with the features learnt in unsupervised fashion only, while the results shown in Section 5.5.3 are computed with the features learnt with a supervised training performed after the unsupervised one. We remind that we use the notation DBN(x,y,z) to refer to the generic DBN with x,y and z nodes in the first, second and third layer, respectively. We present the results organized in tables, where the best F-measures are shown in bold.

Table 5.3: Hit-Rate F-measure, Precision and Recall with interval of tolerance of 0.5 s and 3 s, computed for all the boundary detection algorithms.

BD	Descriptor	F0.5s	P0.5s	R0.5s	F3s	P3s	R3s
Foote	Chroma	0.1573	0.0928	0.5560	0.2682	0.1592	0.9243
	MFCC	**0.1688**	0.1008	0.5637	**0.2824**	0.1692	0.9252
	DBN 1000, 250, 25	0.1965	0.1217	0.5476	0.3320	0.2065	0.9096
	DBN 100, 250, 25	0.1796	0.1092	0.5501	0.3059	0.1866	0.9226
	DBN 500, 250, 200	0.1949	0.1205	0.5428	0.3330	0.2069	0.9100
	DBN 500, 750, 25	**0.1975**	0.1224	0.5455	**0.3343**	0.2080	0.9097
	DBN 75, 50, 25	0.1766	0.1060	0.5678	0.2925	0.1763	0.9247
SF	Chroma	**0.3373**	0.3190	0.3670	**0.7034**	0.6653	0.7653
	MFCC	0.3351	0.3165	0.3657	0.6826	0.6467	0.7405
	DBN 1000, 250, 25	**0.3559**	0.3356	0.3887	0.7009	0.6637	0.7611
	DBN 100, 250, 25	0.3324	0.3184	0.3549	0.6882	0.6605	0.7337
	DBN 500, 250, 200	0.3457	0.3279	0.3755	0.7013	0.6679	0.7564
	DBN 500, 750, 25	0.3460	0.3273	0.3765	**0.7028**	0.6668	0.7608
	DBN 75, 50, 25	0.3302	0.3127	0.3589	0.6879	0.6531	0.7450
C-NMF	Chroma	0.2619	0.2381	0.3060	**0.5527**	0.5050	0.6421
	MFCC	**0.2993**	0.2901	0.3047	0.5342	0.5288	0.5659
	DBN 1000, 250, 25	0.2991	0.3024	0.3091	0.5817	0.5845	0.6041
	DBN 100, 250, 25	0.2984	0.3060	0.3021	0.5796	0.5912	0.5887
	DBN 500, 250, 200	0.3003	0.3062	0.3082	0.5821	0.5896	0.6000
	DBN 500, 750, 25	0.2884	0.2938	0.2952	**0.5831**	0.5916	0.5982
	DBN 75, 50, 25	**0.3079**	0.3114	0.3173	0.5753	0.6004	0.5974
SC	Chroma + MFCC	**0.3669**	0.3840	0.3698	**0.6210**	0.6497	0.6258
	DBN 1000, 250, 25	0.3470	0.3758	0.3325	0.6090	0.6578	0.5845
	DBN 100, 250, 25	0.3154	0.3534	0.2938	0.5931	0.6611	0.5540
	DBN 500, 250, 200	0.3437	0.3746	0.3279	0.6107	0.6620	0.5843
	DBN 500, 750, 25	**0.3479**	0.3772	0.3328	**0.6112**	0.6610	0.5856
	DBN 75, 50, 25	0.3303	0.3635	0.3092	0.5976	0.6556	0.5612

5.5.1 Unsupervised Training

We used the parameters of the feature extraction phase that proved to be the best-performing ones (see Section 5.4): $F_s = 11025Hz$, frame length of $\approx 50ms$, hop-size of $\approx 50ms$ and rectangular window.

Boundary detection

As far as the boundary detection is concerned, the results presented in Table 5.3 show that in general the descriptors extracted with a deep learning network represent a good alternative to the hand-crafted ones.

The DBN-based features performs significantly better than the traditional ones with the Foote algorithm. The best performance of the traditional features is obtained with the MFCC, and our method outperforms it with all the employed DBNs. In particular, we obtain the best results with the DBN(500,750,25). The F-measure at 0.5 seconds is 0.1975, and it is higher with respect to that of the MFCC (i.e., 0.1688). With the aforementioned DBN the F-measure at 3 seconds is 0.3343 against the 0.2824 of the MFCC.

Also the SF algorithm has proved to efficiently work with the DBN-based descriptors, even though the difference with respect to the hand-crafted ones is not as evident as in the Foote algorithm. Again, all the DBNs obtain results comparable with the state of the art, which is in this case represented by performance computed with the chroma. We obtain the highest F-measure at 0.5 seconds (i.e., 0.3559) with the DBN(1000,250,25). Such value significantly outperforms the result obtained with the chroma (i.e., 0.3373). The F-measure at 3 seconds computed with the chroma (i.e., 0.7034) is slightly higher than 0.7028, which has been obtained by the DBN(500,750,25).

The state of the art of the C-NMF algorithm is represented by the MFCC for the F-measure at 0.5 seconds, and by the chroma for the F-measure at 3 seconds. Almost all the DBNs outperform the traditional features, and the DBN(75,50,25) obtains an F-measure at 0.5 seconds of 0.3079, which is higher than the performance of the MFCC (i.e., 0.2993). The best F-measure at 3 seconds (i.e., 0.5831) is higher than the performance of the chroma (i.e., 0.5527), and it has been obtained by the DBN(500,750,25).

The SC algorithm is the only case in which the DBN-based features never outperform the state of the art. We remind that the SC algorithm is specifically designed to work with the MFCC and the chroma, and it is not suitable for the DBN-based features. Nevertheless, we obtain an F-measure at 0.5 seconds of 0.3479 against the 0.3669 of the hand-crafted features, using the DBN(500,750,25). As far as the F-measure at 3 seconds is concerned, we obtain a value of 0.6112, which is slightly lower with respect to the traditional ones (i.e., 0.6210).

To summarize the overall results, the highest F-measure at 0.5 seconds assumes the value of 0.3669 with the SC algorithm (using both MFCC and chroma as descriptors), while the highest F-measure at 3 seconds assumes the value of 0.7034 with the SF algorithm (using the chroma as descriptor).

However, the highest performance obtained with the DBN-based descriptors are close to the traditional ones. In fact, we obtain an F-measure at 0.5 seconds of 0.3559 with the SF algorithm, using the DBN(1000,250,25). Then, with the SF algorithm we also obtain an F-measure at 3 seconds of 0.7028, using the DBN(500,750,25).

Thus, even though the highest performance are obtained with the traditional features, the DBN-based ones are able to obtain results close to the state of the art. Moreover, the performance of the DBN-descriptors are generally significantly higher than those relative to the traditional features, particularly when the DBN is large (i.e., many nodes per layer). Hence, since the employed algorithms have been specifically designed to work with low/mid-level features, we believe that also the DBN-based descriptors extract a reliable low-level representation of the audio signal. In particular, we think that the DBNs are able to extract both the timbral and the harmonic patterns, but also other salient properties, which are not captured by the chroma and the MFCC.

Clustering

We computed the performance with the C-NMF and with the SC clustering algorithms, and we present the results in Table 5.4. Differently from the SC algorithm, C-NMF can be used to cluster the sections obtained with other algorithms, i.e., Foote and SF. In this way, we aim at showing that the clustering is strongly influenced by the boundary detection phase.

The clustering applied to the sections obtained with Foote using all the DBNs significantly outperforms the state of the art (i.e., PF = 0.5150, obtained with the chroma). The highest value (PF = 0.5462) is reached with the DBN(500,250,200).

When SF is used to detect the boundaries, the DBN descriptors obtain results slightly lower than the state of the art (represented by a PF of 0.4821 using the chroma). The highest value computed with our features is a PF of 0.4804, which is obtained with the DBN(100,250,25). Notice that such value is higher than the PF obtained using the MFCC (i.e., 0.4786).

When the C-NMF algorithm is used to both detect boundaries and to label the sections, the results are remarkably higher than the traditional ones (i.e., PF of 0.5467 with the chroma, and of 0.5221 with the MFCC). All the DBNs outperform the state of the art, and we obtain the highest value (i.e., PF = 0.5905) with the DBN(75,50,25).

Finally, also the performance computed with the SC algorithm are higher when any among our descriptors is employed. The best result (PF = 0.6562) is obtained with the DBN(1000,250,25), which outperforms the state of the art, i.e., PF = 0.6379 computed using both the chroma and the MFCC.

The DBN-based descriptors proved to be generally better than the hand-crafted features in the clustering task. Moreover, differently from the boundary detection case, there is not a significant difference between small and large networks.

Table 5.4: Pairwise F-measure, Precision and Recall computed for all the boundary detection and clustering algorithms.

BD	CL	Descriptor	PF	PP	PR
Foote	C-NMF	Chroma	**0.5150**	0.5956	0.4784
		MFCC	0.5001	0.5475	0.4924
		DBN 1000, 250, 25	0.5380	0.6012	0.5224
		DBN 100, 250, 25	0.5341	0.5898	0.5229
		DBN 500, 250, 200	**0.5462**	0.6013	0.5346
		DBN 500, 750, 25	0.5393	0.5995	0.5224
		DBN 75, 50, 25	0.5365	0.6083	0.5112
SF	C-NMF	Chroma	**0.4821**	0.5229	0.4750
		MFCC	0.4786	0.5241	0.4703
		DBN 1000, 250, 25	0.4736	0.5239	0.4649
		DBN 100, 250, 25	**0.4804**	0.5314	0.4708
		DBN 500, 250, 200	0.4782	0.5286	0.4676
		DBN 500, 750, 25	0.4771	0.5246	0.4721
		DBN 75, 50, 25	0.4713	0.5261	0.4658
C-NMF	C-NMF	Chroma	**0.5467**	0.5854	0.5421
		MFCC	0.5221	0.5415	0.5478
		DBN 1000, 250, 25	0.5752	0.6042	0.5951
		DBN 100, 250, 25	0.5745	0.5880	0.6045
		DBN 500, 250, 200	0.5740	0.5932	0.5997
		DBN 500, 750, 25	0.5735	0.5872	0.6071
		DBN 75, 50, 25	**0.5905**	0.6124	0.6110
SC	SC	Chroma + MFCC	**0.6379**	0.5800	0.7553
		DBN 1000, 250, 25	**0.6562**	0.6061	0.7608
		DBN 100, 250, 25	0.6394	0.6020	0.7232
		DBN 500, 250, 200	0.6521	0.5994	0.7596
		DBN 500, 750, 25	0.6551	0.6020	0.7620
		DBN 75, 50, 25	0.6561	0.6033	0.7624

Trade-off between Precision and Recall

Precision and Recall can be seen as indexes of under/over segmentation. In fact, the higher is the number of detected segments, the more likely the boundaries are false positives, and at the same time, the more unlikely they are false negatives. Thus, high values of Precision are generally motivated by a low number of detected boundaries (i.e., undersegmentation), whereas high values of Recall are generally motivated by an high number of detected boundaries (i.e., oversegmentation). As a consequence, when the Precision tends to decrease, the Recall tends to increase. We show a graphical evaluation of such trade-off in Figure 5.10. Then, in Figure 5.9 we show the values of pairwise Precision and Recall, for all the algorithms. All the values are the average of the results computed with all the networks.

In SF, C-NMF and SC Precision and Recall are well balanced, with the Recall that is generally slightly higher than the Precision. Contrarily, the Foote algorithm presents an high Hit-Rate Recall (the Hit-Rate Recall at 3 seconds is generally greater than 0.9) and a low Hit-Rate Precision. The poor performance obtained with Foote confirms that there is oversegmentation.

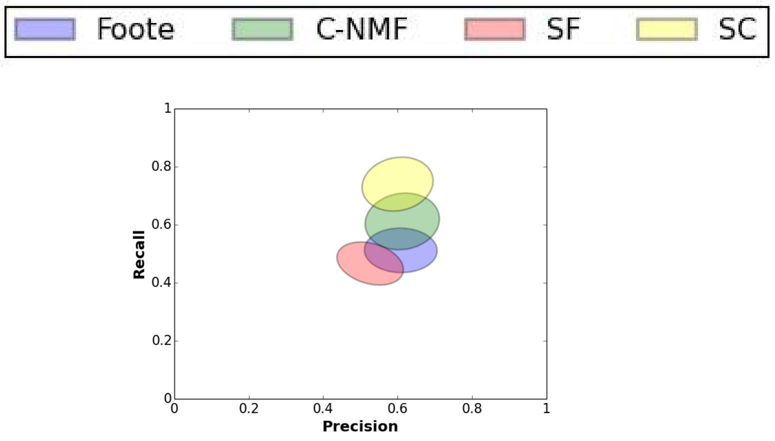
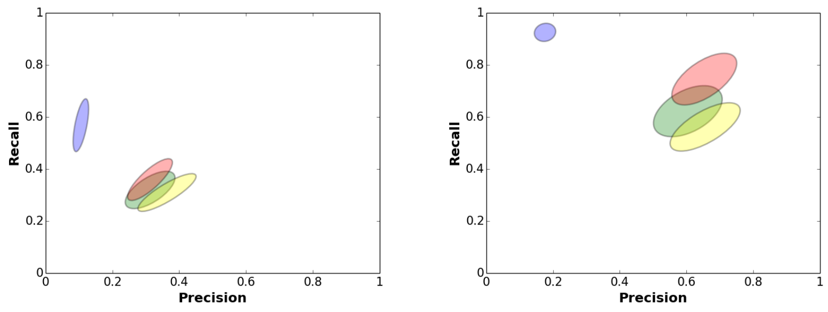


Figure 5.9: Average values of the pairwise Precision and Recall



(a) Trade-off at 0.5 sec *(b) Trade-off at 3 sec*

Figure 5.10: Trade-off between Hit-Rate Precision and Recall

5.5.2 DBN layer used as descriptor

In all the previous test-cases we used the concatenation of the three DBN layers as descriptor. In this experiment we use each DBN layer separately, in order to show how they influence the performance. We use SF to perform the boundary detection and C-NMF to perform the clustering. The results are presented in Table 5.5.

We remind that the higher is the DBN layer, the more abstract are the features that it is able to extract (see Section 3.3.3). Thus, we expect that the first layer captures a representation more close to the acoustic properties of the audio signal, while the third one captures a more abstract representation. In this way, we aim at understanding what is the degree of abstractness required when performing music segmentation. The fixed parameters are:

- rectangular windowing function;
- hop-size $\approx 190ms$;
- frame length $\approx 190ms$;
- $F_s = 11025Hz$.

As far as the boundary detection is concerned, the performance computed using the single layers as descriptors never reach the state of the art, which is represented by the results obtained with the chroma. However, we outperform the state of the art using all the layers of the DBN(500,250,200), obtaining an F-measure at 0.5 seconds of 0.3457 against 0.3426 and an F-measure at 3 seconds of 0.7012 against 0.6964 (both relative to the chroma). The results relative to the MFCC are generally outperformed even when the first and the second layer are employed alone.

As far as the clustering is concerned, the results do not consistently depend on the employed descriptor. We obtain the highest pairwise F-measure value with the first layer of the DBN(500,750,25), i.e., 0.4854, against the 0.4784 of the chroma.

The results also show that the lowest performance are generally obtained using the last layer as unique descriptor. However, we notice we have better results when the last layer is large, since it is more representative. The F-measure at 0.5 seconds obtained with the DBN(500,250,200) is even higher when computed with the last layer (i.e., 0.3221) then when computed with the first one (i.e., 0.3197).

Similarly to the test-cases described in Section 5.4, we show in Figure 5.11 an histogram that allows to graphically evaluate the comparison of the performance. The deeper is the layer, the lower the average performance are. More precisely, the differences between the first and the second layers are small, while the performance become poorer when the third layer is employed. Thus, it seems that the music segmentation task does not require a too abstract representation, or simply that the algorithms that we use are designed to work with lower-level features.

Moreover, it is interesting to notice that the average performance computed with the first layer alone are comparable with those obtained when all the layers are considered. The F-measure at $3s$ and the PF, in particular, are even higher with the first layer. This confirms the discussion about the degree of abstractness required in the task of music segmentation.

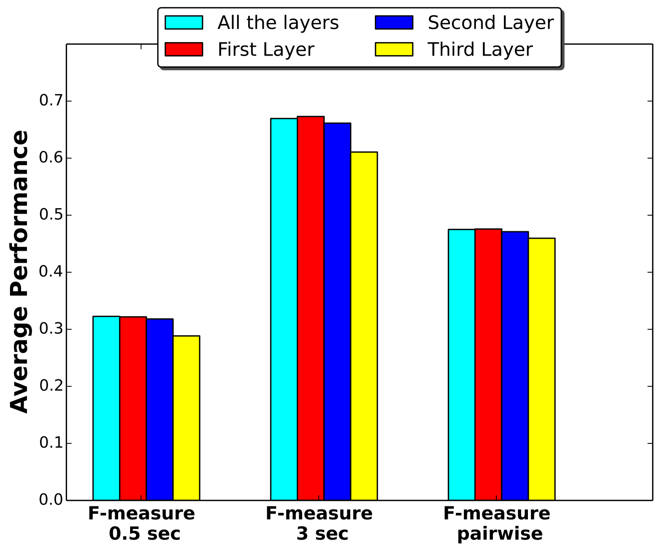


Figure 5.11: Histogram of the performance for varying DBN layer

Table 5.5: Comparison of the Hit-Rate and pairwise F-measures obtained using different combination of DBN layers as descriptor, employing SF and C-NMF as boundary detection and clustering algorithms, respectively

DBN architecture	**Descriptor**	**F 0.5 s**	**F 3 s**	**PF**
	Chroma	0.3426	0.6964	**0.4784**
	MFCC	0.3189	0.6731	0.4743
DBN 1000 , 250 , 25	All Layers	0.3352	0.6813	0.4738
	First Layer	0.3329	0.6828	0.4703
	Second Layer	0.3363	0.6852	0.4724
	Third Layer	0.2835	0.6079	0.4512
DBN 100 , 250 , 25	All Layers	0.3323	0.6882	0.4803
	First Layer	0.3122	0.6616	0.4729
	Second Layer	0.2980	0.6404	0.4629
	Third Layer	0.2796	0.5941	0.4673
DBN 500 , 250 , 200	All Layers	**0.3457**	**0.7012**	0.4782
	First Layer	0.3197	0.6828	0.4835
	Second Layer	0.3386	0.6815	0.4814
	Third Layer	0.3221	0.6721	0.4689
DBN 500 , 750 , 25	All Layers	0.3369	0.6892	0.4761
	First Layer	0.3307	0.6851	**0.4854**
	Second Layer	0.3281	0.6861	0.4775
	Third Layer	0.2796	0.5941	0.4673
DBN 75 , 50 , 25	All Layers	0.2979	0.6421	0.4783
	First Layer	0.3134	0.6526	0.4668
	Second Layer	0.2897	0.6135	0.4615
	Third Layer	0.2835	0.6079	0.4512

5.5.3 Supervised Training

In this section we present the results relative to the supervised training. The parameters are:

- rectangular windowing function;
- $F_s = 11025 Hz$;
- frame length $\approx 190 ms$;
- hop-size $\approx 190 ms$;
- employed network: DBN(75,50,25).

The parameters are related to the test-case described in Section 5.5.2, but not the best-performing ones. In fact, since the computational effort for the fine-tuning process is extremely demanding, we chose a set of parameters in order to have a good performance and a small number of frames. For the same reason, given that also a small network performs significantly well, we trained only the smallest network in supervised fashion.

As described in Section 4.1.2.2, we apply two different fine-tuning approaches: in the first one the cost function is based on the euclidean distance computed between the descriptors of a pair of frames; the second one is more general, since the comparison between pairs of frames is performed by means of a classification layer placed on the top of a deep learning architecture.

We remind that we have set-up the fine-tuning with the goal of obtaining the ideal SSM described in Section 3.2.1. Both the cost functions that we proposed have been defined over the last layer of the DBN. Thus, we consider important to show also how the performance change, before and after the fine-tuning, when the single last layer is employed as descriptor. Moreover, since we have also proposed a different way to compute the SSM, i.e., exploiting the classification layer, we show the results obtained with the Foote algorithm when this SSM is employed instead of the traditional one, i.e., that obtained by means of the euclidean distance. In the following list, we present the possible inputs for the music segmentation algorithms that we choose:

- $\mathbf{H^{(all)}}$, i.e., all the DBN layers in case of unsupervised training;
- $\mathbf{H^{(all)}_{class}}$, i.e., all the DBN layers in case of training performed in supervised fashion and cost function based on the classification node;
- $\mathbf{H^{(all)}_{eu}}$, i.e., all the DBN layers in case of training performed in supervised fashion and cost function based on the euclidean distance;
- $\mathbf{H^{(3)}}$, i.e., only the last DBN layer in case of unsupervised training;
- $\mathbf{H^{(3)}_{eu}}$, i.e., only the last DBN layer in case of training performed in supervised fashion and cost function based on the euclidean distance;
- $\mathbf{SSM_{class}}$, i.e., the SSM computed with the classification layer, only for the Foote algorithm.

In Table 5.6 we provide the quantitative evaluation of the performance obtained with the fine-tuning, both for the boundary detection and the clustering.

The results show that the fine-tuning efficiently works only in some situations. In order to understand why, we have to remind that the supervised training has been realized by providing an information of similarity between each couple of frames, and may be inefficient when applied to algorithms that are not based on the comparison of frames.

Boundary Detection

The Foote algorithm (see Section 4.2.1) is based on the computation of the SSM, and, as expected, the performance are higher after the fine-tuning. In fact, we notice a slightly improvement on the F-measure at 0.5 seconds, and a more relevant improvement on the F-measure at 3 seconds.

In the case of supervised training based on the euclidean distance, for example, when all the layers are employed the F-measure at 0.5 seconds reaches the value of 0.1640 against the 0.1602 obtained with the unsupervised features. Similar considerations can be done for the case of single last layer used as descriptor, and also for the performance after the fine-tuning based on the classification layer.

The difference between the performance is more remarkable as far as the F-measure at 3 seconds is concerned. We obtain, for example, an F-measure of 0.2806 against the previously obtained 0.2658.

The fine-tuning based on the classification layer is able to learn a powerful similarity function (see Section 4.1.2.2). In fact, the performance computed with the alternative SSM, i.e., based on the classification node, are remarkably higher than in the unsupervised case. The F-measure at 0.5 seconds assumes the value of 0.2081 against the previously obtained result, i.e., F-measure of 0.1602. Similarly, the F-measure at 3 seconds assumes the value of 0.4012 and outperforms the previously obtained 0.2658.

The SF algorithm (see Section 4.2.2) is based on the comparison of all the pairs of frames, but not by means of the euclidean distance. As a consequence, only the fine-tuning based on the classification layer improves the unsupervised descriptors. For example, when all the layers are used as descriptors, the F-measure at 3 seconds assumes the value of 0.6421 in the unsupervised case, and the value of 0.6007 in the supervised one. Contrarily, the supervised learning based on the classification layer improve the results. For example, the F-measure at 0.5 seconds is 0.3070, whereas the previously-obtained is 0.2979.

The C-NMF algorithm (see Section 4.2.3) does not compute a comparison among all the pairs of frames. In fact, the results are generally lower after the supervised training. For example, when the fine-tuning based on the

Table 5.6: Hit Rate F-measure with interval of tolerance of 0.5 s and 3 s, and pairwise F-measure, before and after the supervised training, computed for all the algorithms.

BD	CL	Descriptor	F 0.5 s	F 3 s	PF
Foote	C-NMF	**Chroma**	0.1549	0.2594	**0.5103**
		MFCC	**0.1596**	**0.2693**	0.4980
		$\mathbf{H}^{(\mathbf{all})}$	0.1602	0.2658	**0.5326**
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{class}}$	0.1626	0.2806	0.5178
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{eu}}$	0.1640	0.2781	0.5152
		$\mathbf{H}^{(\mathbf{3})}$	0.1511	0.2524	0.5097
		$\mathbf{H}^{(\mathbf{3})}_{\mathbf{eu}}$	0.1542	0.2590	0.5089
		$\mathbf{SSM}_{\mathbf{class}}$	**0.2081**	**0.4012**	0.4776
SF	C-NMF	**Chroma**	**0.3426**	**0.6964**	0.4784
		MFCC	0.3189	0.6731	0.4743
		$\mathbf{H}^{(\mathbf{all})}$	0.2979	0.6421	**0.4783**
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{class}}$	**0.3070**	**0.6451**	0.4656
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{eu}}$	0.2881	0.6007	0.4741
		$\mathbf{H}^{(\mathbf{3})}$	0.2695	0.5613	0.4521
		$\mathbf{H}^{(\mathbf{3})}_{\mathbf{eu}}$	0.2672	0.5673	0.4580
C-NMF	C-NMF	**Chroma**	0.2621	**0.5305**	**0.5368**
		MFCC	**0.2752**	0.5209	0.5198
		$\mathbf{H}^{(\mathbf{all})}$	0.2818	**0.5686**	**0.5612**
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{class}}$	**0.2840**	0.5511	0.5582
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{eu}}$	0.2741	0.5292	0.5351
		$\mathbf{H}^{(\mathbf{3})}$	0.2678	0.5535	0.5193
		$\mathbf{H}^{(\mathbf{3})}_{\mathbf{eu}}$	0.2701	0.5376	0.5319
SC	SC	**MFCC + Chroma**	**0.3553**	**0.6096**	**0.6422**
		$\mathbf{H}^{(\mathbf{all})}$	0.3049	0.4922	0.5689
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{class}}$	**0.3081**	**0.5183**	**0.5933**
		$\mathbf{H}^{(\mathbf{all})}_{\mathbf{eu}}$	0.2918	0.4753	0.5390
		$\mathbf{H}^{(\mathbf{3})}$	0.2733	0.4264	0.5167
		$\mathbf{H}^{(\mathbf{3})}_{\mathbf{eu}}$	0.2705	0.4128	0.5011

classification layer is used, the F-measure at 3 seconds is significantly lower when the fine-tuning based on the euclidean distance is adopted, since it decreases from 0.5686 to 0.5292.

The SC algorithm (see Section 4.2.4) is based on a comparison among frames that does not involve the computation of the euclidean distance. As a consequence, the performance are always lower when the supervised learning based on the euclidean distance is performed. Contrarily, the performance are increased when the other fine-tuning technique, i.e., that based on the classification layer, is employed. For example, the F-measure at 3 seconds, i.e., 0.5183 is higher than in the unsupervised case, i.e., 0.4922.

Clustering

We used two different algorithms to perform the clustering task, i.e., C-NMF and SC. As said, the former one is not based on the comparison among frames, whereas the second one is based on a comparison of frames that does not involve the computation of the euclidean distance.

Since the fine-tuning has been setup aiming to provide an information about the similarity among the pair of frames, we do not expect improvement in the results obtained with the C-NMF algorithm. In fact, independently on the algorithm used to detect the boundaries, the clustering is always worse after the supervised training. An example are the performance of the clustering applied to the sections found with the Foote algorithm, using all the layers as descriptor. In this case, the pairwise F-measure decreases from 0.5326 to 0.5152.

As far as the SC is concerned, we notice a consistent improvement only when the supervised learning based on the classification layer is performed. In fact, the pairwise F-measure is increased from 0.5689 to 0.5933. Contrarily, the performance decrease when the other fine-tuning techniques (i.e., based on the euclidean distance) is employed. An example is the pairwise F-measure that is decreased from 0.5689 to 0.5390, when all the layers are used.

5.6 Final considerations

Finally, we can conclude that the DBN-based descriptors proved to be a good alternative to the hand-crafted ones. Our approach reaches and generally outperforms the state of the art, both in the boundary detection and in the clustering. As far as the supervised learning is concerned, we obtained significant improvements when the fine-tuning well integrates with the employed music segmentation algorithm.

Chapter 6

Conclusions and Future Developments

6.1 Conclusions

In this thesis, we proposed an alternative way to represent music content for the music segmentation task. We used the DBN to derive an abstract representation from the audio signal. In this way, we are able to avoid two well-known issues of hand-crafted features: the selection of the properties of the audio signal to describe and the design of specific signal processing operations that are performed to obtain the sought properties.

The choice of the properties to extract, in fact, requires a deep knowledge of the task to perform. In the task of music segmentation, for example, the properties that have been considered the most important ones in the literature are the harmony and the timbre. However, other salient properties may be relevant when performing music segmentation, and, as a consequence, the traditional approach may lack completeness. Moreover, even when all the salient properties of the audio signal are known, the traditional signal processing operations may not be able to properly extract them.

In this thesis we trained several DBNs in unsupervised fashion, and we computed the performance with several algorithms presented in literature. Although such algorithms have been designed to work with the traditional features (the chroma to describe the harmony and the MFCC to describe the timbre, in particular), we obtained results comparable with the state of the art, and better in some cases.

Moreover, in order to derive an application-dependent set of learnt features, we setup a supervised learning procedure with the aim of deriving the ideal Self-Similarity Matrix, according to two different fine-tuning strategies.

The first one is based on the computation of the euclidean distance between the descriptors of each couple of frames. The second one is based on a classification layer put on the top of a deep learning architecture that classifies the frames as belonging to the same structural part or not. With the second approach, the fine-tuning do not only tune the parameters of the network, but it also learns a similarity function that allows to build a new type of SSM. The supervised learning is generally able to improve the performance of the algorithms that are based on the comparison of the frames. The most remarkable improvement with respect to the state of the art is obtained with the SSM computed exploiting the classification layer, confirming the ability of the neural networks to learn a complex representation of the input data.

6.2 Future Developments

6.2.1 Feature extraction

To the best of our knowledge, a set of descriptors extracted with a deep learning approach has never been employed as input for a music segmentation algorithm. Since we obtained very promising results, we think that there is considerable room for improvement in this direction. In particular, we propose to expand the training databases and to exploit other deep learning architectures.

Dataset Expansion

The deep learning networks require a massive amount of data to be efficiently trained. In this thesis we used all the available SALAMI database to perform an unsupervised pre-training phase, and a small part of it to perform also the supervised one (only for one DBN architecture). Since it is possible to train a DBN in unsupervised fashion with non-annotated datasets (see Section 3.3.6), which are largely available on the Web, a possible development consists in training a network with a very large number of songs for the unsupervised phase, and then to fine-tune the learnt parameters with one of the available annotated datasets.

Deep Learning architectures

In this thesis we employed only one type of deep learning architecture, the Deep Belief Network, with a depth of three layers. Since the results confirm the efficiency of the deep learning techniques, we intend to increase the depth of the network, as well as to use different types of layers (which are used to build other deep architectures).

As far as the depth of the network is concerned, the results indicate that the performance are lower when the employed descriptor is the last layer of the DBN, and we have consequently deduced that the music segmentation task does not properly work with a too abstract representation. However, it is possible that the music segmentation algorithms require a degree of abstractness that the adopted DBNs have not been able to reach. Thus, knowing that the depth of the network is a critical aspect of the deep learning techniques [38], we intend to experiment networks with more than three layers.

As far as the type of layer is concerned, examples of other possible networks are the Convolutional Neural Network (CNN), which is commonly used in the field of image and video recognition, and the Recurrent Neural Network (RNN), which is characterized by the introduction of a memory term that allows to take into consideration the temporal dependencies among the input data.

The CNN allows to treat the spectrograms as images and to find relevant patterns within them, using the convolutional layers to detect if the locally learnt features are also useful for other regions. The CNN has been adopted in the music processing for the task of musical onset detection in [66], and for the boundary detection subtask in [42]. We also intend to use the CNN in the music segmentation task, but to perform the feature extraction phase.

In the RNN the hidden units assume a value that is a function of the current and of the past input data. Such capability may be useful in the music analysis, where the dependencies among musical events is of crucial importance. The RNN has been used to perform musical improvisation in [67].

6.2.2 Music Segmentation algorithms

Since the employed algorithms have been designed to work with an handcrafted representation of music, a possible improvement consists in developing a music segmentation system that is more suitable for the DBN-based descriptors. The purpose is to create a system that performs both the feature extraction phase and the music segmentation, i.e., boundary detection and clustering.

In this work, we setup the supervised learning that better integrate with the available algorithms. A possible development consists in realizing the music segmentation algorithm itself as a deep learning architecture, which may properly exploit its supervised training.

Appendices

Appendix A

List of songs in the Beatles dataset

We present now all the songs from the Beatles discography, which has been used to test the performance of the music segmentation algorithms.

Year	Album	Title
1963	Please Please Me	I Saw Her Standing There
1963	Please Please Me	Misery
1963	Please Please Me	Anna(Go To Him)
1963	Please Please Me	Chains
1963	Please Please Me	Boys
1963	Please Please Me	Ask Me Why
1963	Please Please Me	Please Please Me
1963	Please Please Me	Love Me Do
1963	Please Please Me	P. S. I Love You
1963	Please Please Me	Baby Its You
1963	Please Please Me	Do You Want To Know A Secret
1963	Please Please Me	A Taste Of Honey
1963	Please Please Me	There's A Place
1963	Please Please Me	Twist And Shout
1963	With The Beatles	It Won't Be Long
1963	With The Beatles	All I've Got To Do
1963	With The Beatles	All My Loving
1963	With The Beatles	Don't Bother Me
1963	With The Beatles	Little Child
1963	With The Beatles	Till There Was You

1963	With The Beatles	Please Mister Postman
1963	With The Beatles	Roll Oover Beethoven
1963	With The Beatles	Hold Me Tight
1963	With The Beatles	You Really Got A Hold On Me
1963	With The Beatles	I Wanna Be Your Man
1963	With The Beatles	Devil In Her Heart
1963	With The Beatles	Not A Second Time
1963	With The Beatles	Money
1964	A Hard Day’s Night	A Hard Day’s Night
1964	A Hard Day’s Night	I Should Have Known Better
1964	A Hard Day’s Night	If I Fell
1964	A Hard Day’s Night	I’m Happy Just To Dance With You
1964	A Hard Day’s Night	And I Love Her
1964	A Hard Day’s Night	Tell Me Why
1964	A Hard Day’s Night	Can’t Buy Me Love
1964	A Hard Day’s Night	Any Time At All
1964	A Hard Day’s Night	I’ll Cry Instead
1964	A Hard Day’s Night	Things We Said Today
1964	A Hard Day’s Night	When I Get Home
1964	A Hard Day’s Night	You Can’t Do That
1964	A Hard Day’s Night	I’ll Be Back
1964	Beatles For Sale	No Reply
1964	Beatles For Sale	I’m A Loser
1964	Beatles For Sale	Baby’s In Black
1964	Beatles For Sale	Rock Roll Music
1964	Beatles For Sale	I’ll Follow The Sun
1964	Beatles For Sale	Mr. Moonlight
1964	Beatles For Sale	Kansas City Hey Hey Hey Hey(Medley)
1964	Beatles For Sale	Eight Days A Week
1964	Beatles For Sale	Words Of Love
1964	Beatles For Sale	Honey Don’t
1964	Beatles For Sale	Every Little Thing
1964	Beatles For Sale	I Don’t Want To Spoil The Party
1964	Beatles For Sale	What You’re Doing
1964	Beatles For Sale	Everybody’s Trying To Be My Baby
1965	Help!	Help!
1965	Help!	The Night Before

1965	Help!	You’ve Got To Hide Your Love Away
1965	Help!	I Need You
1965	Help!	Another Girl
1965	Help!	You’re Going To Lose That Girl
1965	Help!	Ticket To Ride
1965	Help!	Act Naturally
1965	Help!	It’s Only Love
1965	Help!	You Like Me Too Much
1965	Help!	Tell Me What You See
1965	Help!	I’ve Just Seen A Face
1965	Help!	Yesterday
1965	Help!	Dizzy Miss Lizzy
1965	Rubber Soul	Drive My Car
1965	Rubber Soul	Norwegian Wood(This Bird Has Flown)
1965	Rubber Soul	You Won’t See Me
1965	Rubber Soul	Nowhere Man
1965	Rubber Soul	Think For Yourself
1965	Rubber Soul	The Word
1965	Rubber Soul	Michelle
1965	Rubber Soul	What Goes On
1965	Rubber Soul	Girl
1965	Rubber Soul	I’m Looking Through You
1965	Rubber Soul	In My Life
1965	Rubber Soul	Wait
1965	Rubber Soul	If I Needed Someone
1965	Rubber Soul	Run For Your Life
1966	Revolver	Taxman
1966	Revolver	Eleanor Rigby
1966	Revolver	I’m Only Sleeping
1966	Revolver	Love You To
1966	Revolver	Here There Everywhere
1966	Revolver	Yellow Submarine
1966	Revolver	She Said She Said
1966	Revolver	Good Day Sunshine
1966	Revolver	And Your Bird Can Sing
1966	Revolver	For No One
1966	Revolver	Doctor Robert

1966	Revolver	I Want To Tell You
1966	Revolver	Got To Get You Into My Life
1966	Revolver	Tomorrow Never Knows
1967	Sgt Pepper's Lonely Hearts Club Band	Sgt. Pepper's Lonely Hearts Club Band
1967	Sgt Pepper's Lonely Hearts Club Band	With A Little Help From My Friends
1967	Sgt Pepper's Lonely Hearts Club Band	Lucy In The Sky With Diamonds
1967	Sgt Pepper's Lonely Hearts Club Band	Getting Better
1967	Sgt Pepper's Lonely Hearts Club Band	Fixing A Hole
1967	Sgt Pepper's Lonely Hearts Club Band	She's Leaving Home
1967	Sgt Pepper's Lonely Hearts Club Band	Being For The Benefit Of Mr. Kite!
1967	Sgt Pepper's Lonely Hearts Club Band	Within You Without You
1967	Sgt Pepper's Lonely Hearts Club Band	When I'm Sixty- Four
1967	Sgt Pepper's Lonely Hearts Club Band	Lovely Rita
1967	Sgt Pepper's Lonely Hearts Club Band	Good Morning Good Morning
1967	Sgt Pepper's Lonely Hearts Club Band	Sgt. Pepper's Lonely Hearts Club Band(Reprise)
1967	Sgt Pepper's Lonely Hearts Club Band	A Day In The Life
1967	Magical Mystery Tour	Magical Mystery Tour
1967	Magical Mystery Tour	The Fool On The Hill
1967	Magical Mystery Tour	Flying
1967	Magical Mystery Tour	Blue Jay Way
1967	Magical Mystery Tour	Your Mother Should Know
1967	Magical Mystery Tour	I Am The Walrus
1967	Magical Mystery Tour	Hello Goodbye
1967	Magical Mystery Tour	Strawberry Fields Forever
1967	Magical Mystery Tour	Penny Lane
1967	Magical Mystery Tour	Baby You're A Rich Man
1967	Magical Mystery Tour	All You Need Is Love
1968	TheWhite Album CD1	Back In The U S S R
1968	TheWhite Album CD1	Dear Prudence
1968	TheWhite Album CD1	Glass Onion
1968	TheWhite Album CD1	Ob- La- Di Ob- La- Da
1968	TheWhite Album CD1	Wild Honey Pie
1968	TheWhite Album CD1	The Continuing Story Of Bungalow Bill
1968	TheWhite Album CD1	While My Guitar Gently Weeps
1968	TheWhite Album CD1	Happiness Is A Warm Gun
1968	TheWhite Album CD1	Martha My Dear
1968	TheWhite Album CD1	I'm So Tired

1968	TheWhite Album CD1	Blackbird
1968	TheWhite Album CD1	Piggies
1968	TheWhite Album CD1	Rocky Raccoon
1968	TheWhite Album CD1	Don't Pass Me By
1968	TheWhite Album CD1	Why Don't We Do It In The Road
1968	TheWhite Album CD1	I Will
1968	TheWhite Album CD1	Julia
1968	TheWhite Album CD2	Birthday
1968	TheWhite Album CD2	Yer Blues
1968	TheWhite Album CD2	Mother Nature's Son
1968	TheWhite Album CD2	Everybody's Got Somethingto Hide Except Me And Mr Monkey
1968	TheWhite Album CD2	Sexy Sadie
1968	TheWhite Album CD2	Helter Skelter
1968	TheWhite Album CD2	Long Long Long
1968	TheWhite Album CD2	Revolution1
1968	TheWhite Album CD2	Honey Pie
1968	TheWhite Album CD2	Savoy Truffle
1968	TheWhite Album CD2	Cry Baby Cry
1968	TheWhite Album CD2	Good Night
1969	Abbey Road	Come Together
1969	Abbey Road	Something
1969	Abbey Road	Maxwell's Silver Hammer
1969	Abbey Road	Oh! Darling
1969	Abbey Road	Octopus's Garden
1969	Abbey Road	I Want You(She's So Heavy)
1969	Abbey Road	Here Comes The Sun
1969	Abbey Road	Because
1969	Abbey Road	You Never Give Me Your Money
1969	Abbey Road	Sun King
1969	Abbey Road	Mean Mr. Mustard
1969	Abbey Road	Polythene Pam
1969	Abbey Road	She Came In Through The Bathroom Window
1969	Abbey Road	Golden Slumbers
1969	Abbey Road	Carry That Weight
1969	Abbey Road	The End
1969	Abbey Road	Her Majesty
1970	Let It Be	Twoof Us

1970	Let It Be	Dig a Pony
1970	Let It Be	Acrossthe Universe
1970	Let It Be	I Me Mine
1970	Let It Be	Dig It
1970	Let It Be	Let It Be
1970	Let It Be	Maggie Mae
1970	Let It Be	I've Gota Feeling
1970	Let It Be	One After909
1970	Let It Be	The Longand Winding Road
1970	Let It Be	For You Blue
1970	Let It Be	Get Back

Table A.1: The Beatles discography

Appendix B

Details of the experimental setup

We present here the quantitative results relative to the experiments described in Section 5.4. Notice that the computation of the Chroma and the MFCC is always performed with the application of a rectangular window. Thus, we can not compare the DBN-based descriptors computed with the Hamming window with the state of the art.

Descriptor	**F 0.5 s**	**F 3 s**	**PF**
$F_s = 44100$	**Frame Length $\approx$ 50ms**	**Hop-size $\approx$ 50ms**	**Rectangular window**
Chroma	**0.3369**	**0.7120**	**0.4771**
MFCC	0.3266	**0.6784**	0.4682
DBN 1000 , 250 , 25	0.3022	0.6675	0.4703
DBN 100 , 250 , 25	**0.3088**	0.6562	0.4635
DBN 500 , 250 , 200	0.3031	0.6284	0.4572
DBN 500 , 750 , 25	0.3068	0.6681	**0.4756**
DBN 75 , 50 , 25	0.2879	0.5777	0.4645
$F_s = 11025$	**Frame Length $\approx$ 50ms**	**Hop-size $\approx$ 50ms**	**Rectangular window**
Chroma	**0.3373**	**0.7034**	0.4712
MFCC	0.3351	0.6826	**0.4760**
DBN 1000 , 250 , 25	0.3559	0.7009	0.4736
DBN 100 , 250 , 25	0.3324	0.6882	**0.4804**
DBN 500 , 250 , 200	0.3457	0.7013	0.4783
DBN 500 , 750 , 25	**0.3461**	**0.7028**	0.4772
DBN 75 , 50 , 25	0.3302	0.6879	0.4713
$F_s = 11025$	**Frame Length $\approx$ 190ms**	**Hop-size $\approx$ 50ms**	**Rectangular window**

Chroma	0.3371	**0.7044**	0.4789
MFCC	**0.3399**	0.6895	0.4763
DBN 1000 , 250 , 25	**0.3480**	**0.7059**	0.4718
DBN 100 , 250 , 25	0.3227	0.6847	0.4631
DBN 500 , 250 , 200	0.3351	0.6968	0.4771
DBN 500 , 750 , 25	0.3448	0.6972	**0.4796**
DBN 75 , 50 , 25	0.3435	0.6849	0.4741

$\mathbf{F_s = 11025}$	**Frame Length ≈ 190ms**	**Hop-size ≈ 190ms**	**Rectangular window**
Chroma	**0.3426**	**0.6964**	**0.4784**
MFCC	0.3189	0.6731	0.4743
DBN 1000 , 250 , 25	0.3352	0.6813	0.4738
DBN 100 , 250 , 25	0.3323	0.6882	**0.4803**
DBN 500 , 250 , 200	**0.3457**	**0.7012**	0.4782
DBN 500 , 750 , 25	0.3369	0.6892	0.4761
DBN 75 , 50 , 25	0.2979	0.6421	0.4783

$\mathbf{F_s = 11025}$	**Frame Length ≈ 190ms**	**Hop-size ≈ 190ms**	**Hamming window**
DBN 1000 , 250 , 25	0.3214	0.6793	**0.4733**
DBN 100 , 250 , 25	0.3045	0.6376	0.4655
DBN 500 , 250 , 200	0.3239	**0.6854**	0.4671
DBN 500 , 750 , 25	**0.3266**	0.6849	0.4719
DBN 75 , 50 , 25	0.3054	0.6299	0.4635

Bibliography

[1] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis." in *ISMIR*, 2010, pp. 625–636.

[2] M. J. Bruderer, M. F. McKinney, and A. Kohlrausch, "Structural boundary perception in popular music." in *ISMIR*, 2006, pp. 198–201.

[3] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 236–240.

[4] O. Nieto and J. Bello, "Music segment similarity using 2d-fourier magnitude coefficients," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 664–668.

[5] K. Jensen, "Multiple scale music segmentation using rhythm, timbre, and harmony," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 159–159, 2007.

[6] W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, "An efficient mfcc extraction method in speech recognition," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on.* IEEE, 2006, pp. 4–pp.

[7] J. T. Foote, "Content-based retrieval of music and audio," in *Voice, Video, and Data Communications.* International Society for Optics and Photonics, 1997, pp. 138–147.

[8] T. Li and M. Ogihara, "Toward intelligent music information retrieval," *Multimedia, IEEE Transactions on*, vol. 8, no. 3, pp. 564–574, 2006.

[9] Y. Itoh, A. Iwabuchi, K. Kojima, M. Ishigame, K. Tanaka, and S.-W. Lee, "Automatic music boundary detection using short segmen-

tal acoustic similarity in a music piece," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2008, p. 6, 2008.

[10] M. L. Cooper and J. Foote, "Automatic music summarization via similarity analysis." in *ISMIR*, 2002.

[11] S. Dubnov, "Unified view of prediction and repetition structure in audio signals with application to interest point detection," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 327–337, 2008.

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

[13] J. Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM, 1999, pp. 77–80.

[14] F. Kaiser and T. Sikora, "Music structure discovery in popular music using non-negative matrix factorization." in *ISMIR*, 2010, pp. 429–434.

[15] M. Müller and F. Kurth, "Towards structural analysis of audio recordings in the presence of musical variations," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 163–163, 2007.

[16] D. Turnbull, G. R. Lanckriet, E. Pampalk, and M. Goto, "A supervised approach for detecting boundaries in music using difference features and boosting." in *ISMIR*, 2007, pp. 51–54.

[17] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 452–455.

[18] J. Serra, M. Muller, P. Grosche, and J. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *Multimedia, IEEE Transactions on*, vol. 16, no. 5, pp. 1229–1240, Aug 2014.

[19] J. Serra, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised detection of music boundaries by time series structure features," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[20] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 236–240.

[21] B. Logan and S. Chu, "Music summarization using key phrases," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. II749–II752.

[22] J. T. Foote and M. L. Cooper, "Media segmentation using self-similarity decomposition," in *Electronic Imaging 2003.* International Society for Optics and Photonics, 2003, pp. 167–175.

[23] V. Klema and A. J. Laub, "The singular value decomposition: Its computation and some applications," *Automatic Control, IEEE Transactions on*, vol. 25, no. 2, pp. 164–176, 1980.

[24] J.-J. Aucouturier and M. Sandler, "Segmentation of musical signals using hidden markov models," *Preprints-Audio Engineering Society*, 2001.

[25] G. Peeters, A. La Burthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis." in *ISMIR*, vol. 2, 2002, pp. 94–100.

[26] L. Barrington, A. Chan, and G. Lanckriet, "Modeling music as a dynamic texture," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 602–612, March 2010.

[27] M. Müller, *Information retrieval for music and motion.* Springer, 2007, vol. 2.

[28] L. Lu, M. Wang, and H.-J. Zhang, "Repeating pattern discovery and structure analysis from acoustic music data," in *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval.* ACM, 2004, pp. 275–282.

[29] B. S. Ong *et al.*, "Structural analysis and segmentation of music signals," 2007.

[30] J. Wellhausen and M. Höynck, "Audio thumbnailing using mpeg-7 low-level audio descriptors," in *ITCom 2003.* International Society for Optics and Photonics, 2003, pp. 65–73.

[31] M. Goto, "A chorus-section detecting method for musical audio signals," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 5. IEEE, 2003, pp. V–437.

[32] Y. Shiu, H. Jeong, and C.-C. Kuo, "Similar segment detection for music structure analysis via viterbi algorithm," in *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006, pp. 789–792.

[33] J.-J. Aucouturier and M. Sandler, "Finding repeating patterns in acoustic musical signals: Applications for audio thumbnailing," in *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*. Audio Engineering Society, 2002.

[34] C. Rhodes, M. A. Casey *et al.*, "Algorithms for determining and labelling approximate hierarchical self-similarity," 2007.

[35] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning-a new frontier in artificial intelligence research [research frontier]," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 4, pp. 13–18, 2010.

[36] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics." in *ISMIR*. Citeseer, 2012, pp. 403–408.

[37] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1561/2200000006

[38] ——, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[39] M. Buccoli, P. Bestagini, M. Zanoni, A. Sarti, and S. Tubaro, "Unsupervised feature learning for bootleg detection using deep learning architectures," in *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*, 2014.

[40] E. M. Schmidt and Y. E. Kim, "Learning emotion-based acoustic features with deep belief networks," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6082328

[41] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks." in *ISMIR*. Utrecht, The Netherlands, 2010, pp. 339–344.

[42] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014), Taipei, Taiwan*, 2014.

[43] A. Kanagasundaram, D. Dean, and S. Sridharan, "Jfa based speaker recognition using delta-phase and mfcc features," in *SST 2012 14th Australasian International Conference on Speech Science and Technology*, 2012.

[44] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proc. of the International Computer Music Conference*, 2006.

[45] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[46] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 318–326, 2008.

[47] F. Kaiser, "Music structure segmentation," Ph.D. dissertation, Universitätsbibliothek der Technischen Universität Berlin, 2012.

[48] X. Shao and S. G. Johnson, "Type-ii/iii dct/dst algorithms with reduced number of arithmetic operations," *Signal Processing*, vol. 88, no. 6, pp. 1553–1564, 2008.

[49] J. Foote, "A similarity measure for automatic audio classification," in *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, 1997.

[50] Y. LeCun, S. Chopra, and R. Hadsell, "A tutorial on energy-based learning," 2006.

[51] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[52] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[53] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Journal Neural Computation (JNC)*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[54] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and models.* Springer Science & Business Media, 2007, vol. 22.

[55] M. McCandless, "The mp3 revolution," *Intelligent Systems and their Applications, IEEE*, vol. 14, no. 3, pp. 8–9, May 1999.

[56] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[57] B. McFee and D. P. Ellis, "Analyzing song structure with spectral clustering," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[58] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 45–55, 2010.

[59] H. M. Lukashevich, "Towards quantitative measures of evaluating song segmentation." in *ISMIR*, 2008, pp. 375–380.

[60] O. Nieto, M. M. Farbood, T. Jehan, and J. P. Bello, "Perceptual analysis of the f-measure for evaluating section boundaries in music."

[61] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations." in *ISMIR*, vol. 11, 2011, pp. 555–560.

[62] G. Van Rossum and F. L. Drake Jr, *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[63] T. E. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[64] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.

[65] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir eval: A transparent implementation of common mir metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference, Taipei, Taiwan*, 2014.

[66] J. Schluter and S. Bock, "Improved musical onset detection with convolutional neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 6979–6983.

[67] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 2002.