On Music Structure Analysis

Machine learning implementations of the Segmentation by Annotation approach

Author:
Leander van Boven

Supervisor:
Dr. M.P. (Marijn) Schraagen

Second Supervisor:
Prof. dr. C.J. (Kees) van Deemter

A thesis presented for the degree of
Bachelor of Science

Faculty of Humanities
University Utrecht
The Netherlands
June 2020

On Music Structure Analysis

Machine learning implementations of the Segmentation by Annotation approach

Leander van Boven (6215637)

Abstract

This thesis proposes a novel approach to Music Structure Analysis (MSA). This approach implements the Segmentation by Annotation (SbA) approach to MSA, using a convolutional neural network (CNN) and an artificial neural network using Long Short-Term Memory (LSTM) units. An overview of the current advances in music structure analysis is given as well as the use of the proposed architectures in similar research fields. A description of the testing methods is provided in which the proposed architectures show promising results on the custom ground truth used. This custom ground truth is a modified version of the segment function annotations found in the internet archives subset of the SALAMI dataset. The ground truth is modified by reducing the amount of unique functions from 26 to 9 because of the low occurrence of some labels in the original data and to improve the accuracy of the machine learning models. Due to this custom ground truth, comparison with the current state-of-the-art of music structure analysis proved quite hard. By comparing the SbA approach to the (more symbolic) Distance-based Segmentation and Annotation approach, a comparison between using machine learning and non-machine learning techniques can be made. Future research is proposed to enhance the segmentation by annotation approach as well as music structure analysis in general.

Keywords: Music Structure Analysis, Music Information Retrieval, Segmentation by Annotation, Convolutional Neural Networks, Long Short-Term Memory

The more we think we know about, the greater the unknown
We suspend our disbelief, and we are not alone
Mystic rhythms – capture my thoughts
Carry them away…

NEIL ELLWOOD PEART
Rush - Mystic Rhythms

Contents

Chapter 1

Introduction

Music is all around us. Everybody listens to it in a different way [17], only a few actually create it. Creating music takes a lot of effort; making sure all notes and tones form some kind of harmony and melody, and that all harmonies and melodies form a coherent piece, among many other difficult aspects that involve creating music. Although these aspects are all very difficult and only some people are actually good at it, computers are far worse in it.
Listening to music involves many subconscious processes that are influenced by, among other things, emotion. Therefore making a computer understand the concept of music is a difficult task. Due to the hierarchical properties of music [15, 46], one of the first steps of this process is understanding the musical structure of a piece of music. This will be the main focus of this thesis.

1.1 Musical Structure

The musical structure of a piece of music can be defined in many different ways; on a high level or low level, and can be of different meanings in different kinds of music.
Therefore for the remainder of this thesis I will refer to music as music in the genre of Western Popular Music [74] and make the following definitions:

Definition 1.1.1 (Musical Structure)**.** The *Musical Structure* of a piece of music is a series of *segments* (1.1.2) that describes the high-level structure of a piece of music.

Definition 1.1.2 (Segment)**.** A *Segment* is one consecutive piece of audio in a piece of music that has one *segment function* (1.1.3), with a start- and an end-boundary.

Definition 1.1.3 (Segment Function)**.** Following the definitions as they can be found in Benward and Saker [5], Part B: *The Structural Elements of Music*, a *Segment Function* is the function of a *segment*, this is either:

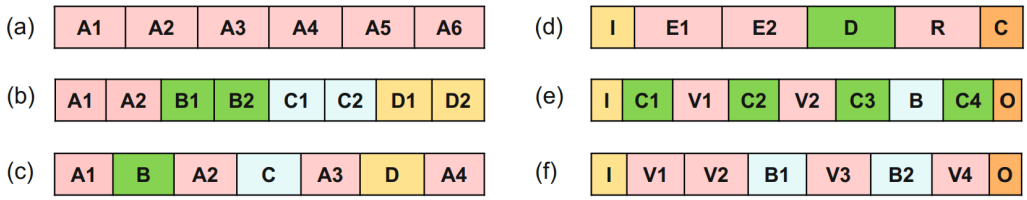


Figure 1.1: Examples for musical structures as encountered in Western music. **(a)** Strophic form. **(b)** Chain form with repetitions. **(c)** Rondo form. **(d)** Sonata form. **(e)** Beatles song "Tell Me Why." **(f)** Beatles song "Yesterday."
Reprinted from Müller [59, p. 173] (Figure 4.4)

- Chorus
- Verse
- Bridge
- Interlude / Transition
- Intro
- Outro
- Solo
- (Silence)
- (Background Noise)

I will refer to the extraction of the musical structure of a piece of music as *Musical Structure Analysis* or MSA. An example of musical structures can be found in Figure 1.1. The aim of this thesis is to produce a structure of a song similar to 1.1e and 1.1f, where **I** stands for intro, **V1** stands for Verse 1, etc.

1.2 Applications

The musical structure of a piece of music has many applications, within research and the industry. For example the musical structure can be used to only retrieve a certain part of a piece of music. This can then be used to preview only the chorus of a song when someone is browsing through a list of songs to find a certain song. Being able to quickly seek through a song to its verses or chorus can be of great addition to many music streaming, or other streaming, services.

More commercial applications could be to limit ones listening capabilities by restricting a user to listen only to the verse or chorus of a song, while blocking the other parts of a song behind some pay-wall. Less aggressive commercial applications could be to place an advert between for example the verse and chorus, when the user is using a free subscription to a music streaming service.

The scope of this thesis, however, will be to place the music structure problem in a more research driven, less commercial field, and the field of AI in particular.

1.2.1 Relevance to AI

Finding and analyzing the structure of a musical piece is an important part of music information retrieval and an important step in music processing and music analysis, both for humans and computers. Once an understanding of the basic structure of a piece of music is established, it can be used to establish more complex understandings about that piece of music.

One of these uses may be to use the musical structure to get more insight in how certain songs are equal to each other, or how certain structures are used for certain kinds of songs or in certain genres. This thus may be used as additional information when finding for example the genre of a piece of music.

Another way of using this musical structure could be to extract lower level musical patterns from a piece of music. This could be leading melodies, a riff[1], lyrics or the hook [16] of a song [13].

Additionally, knowledge about the structure of songs that are produced by humans can be used for generating songs using computers. This information can for example be used as additional information during the generation phase, or as validation information for already generated songs to validate whether these generated songs comply to human musical structure.

For humans, finding the musical structure is quite trivial, because they constantly and often unconsciously adapt themselves to the musical and acoustic properties of what they listen to. However the amount of different musical structures make computational structure analysis a challenging problem.

1.3 Segmentation by Annotation versus Distance-based Segmentation and Annotation

Extracting the musical structure of a piece of music can be done in a few ways, however almost all of these methods can be generalized into two general approaches: *Segmentation by Annotation* (SbA) and *Distance-based Segmentation and Annotation* (DSA).

Definition 1.3.1 (Segmentation by Annotation)**. Segmentation by Annotation** in the context of music structure analysis means first dividing a song into many small pieces (e.g. pieces of 10ms or every beat). Thereafter a *segment function* is assigned to each small piece using some kind of classification method (e.g. statistics, support vector machine, deep learning). Sequences of similarly annotated small pieces are then combined into *segments*, resulting in the musical structure of the song.

[1] A famous example of a (rock) riff can be found in *Smoke on the Water* played by Ritchie Blackmore of Deep Purple.

Definition 1.3.2 (Distance-based Segmentation and Annotation)**. Distance-based Segmentation and Annotation** in the context of music structure analysis means first finding segment boundaries using some kind of distance metric. Subsequently these segments are grouped into groups that are similar to each other in their structural role in the piece of music, using some kind of grouping method (e.g. nearest neighbors, support vector machines), resulting in each segment getting a capital letter denoting their function. Then, each capital letter is converted into a segment function, this can be either done using some kind of statistics, pattern matching or machine learning method.

Distance-based Segmentation and Annotation has been researched a lot in the context of music structure analysis and was for a long time the best approach. See chapter 2 for an overview of research in this areas. Segmentation by Annotation has been used a lot less for this exact application but more for separating and classifying pieces of speech, music and different kind of background noises in an audio stream. This is because speech, music and background noise all differ a lot more in their auditory features than different parts of music within the same piece. Because the use of the distance-based approach yielded increasingly better results, recent research focused on that. However with the increasing popularity and performance of machine learning in many disciplines like computer vision or natural language processing and production, applying machine learning to the music structure problem has become increasingly popular.

1.4 Combined Research

Although the DSA approach to MSA is become state-of-the-art, its results can always be improved. Therefore not only putting effort into improving the Segmentation by Annotation approach but also the Distance-based Segmentation and Annotation approach will benefit the Music Structure Analysis problem. This work is related to Kuiper [43] in such a way that this work focuses on the Segmentation by Annotation approach, and Kuiper focuses on giving an overview and improvement of the Distance-based Segmentation and Annotation approach.

1.5 Research Questions

The following research questions can be formulated:

Research Question 1 (Main RQ)**.** What is the feasibility of a machine learning implementation of the Segmentation by Annotation approach to Music Structure Analysis in Western Popular Music?

The sub-questions that are part of this main research question are:

Research Question 2 (Sub-RQ 1)**.** Which deep learning architecture, implementing the Segmentation by Annotation approach to Music Structure Analysis, yields the best relative results?

Research Question 3 (Sub-RQ 2)**.** How does the performance of the best performing deep learning architecture, implementing the Segmentation by Annotation approach to Music Structure Analysis, compare to the performance of implementations of the Distance-based Segmentation and Annotation approach, and a state-of-the-art implementation in particular?

To answer the main research question I will create different types of deep learning models, using different kinds of architectures. Then I will train these models on different combinations of musical features extracted from the songs present in the internet archives subset of the SALAMI dataset (see section 3.1 for an explanation of this dataset). I will use the annotations, produced by human annotators, of these songs as ground truth to determine the absolute performance of each implementation. The absolute performance of the best performing implementation can then be used to compare a machine learning application of the Segmentation by Annotation approach to MSA to the absolute performance of a state-of-the-art implementation of the Distance-based Segmentation and Annotation approach, determined in the same way. I will elaborate more on this in chapter 3.

1.6 Outline

In chapter 2 I will first explore previous work regarding the SbA and DSA approach. Here I will also discuss the current state-of-the-art and describe the background of the models I created. Lastly I will give an overview of the dataset I used, and a listing of possible acoustic features that can be extracted from this dataset.

In chapter 3 I introduce the architectures of the models I created. In this chapter I also describe the data preparation and feature selection that preceded the testing phase of this research.

Then, in chapter 4 I describe the training and evaluation setup and some preliminary results. A final test setup is described as well as the final results following from this final test.

Thereafter, chapter 5 discusses these results and explores reasons for these results. It concludes with a comparison between the segmentation by annotation approach and distance-based segmentation and annotation approach to music structure analysis.

This thesis ends with chapter 6. In this final chapter I propose future research that can be performed to improve both the segmentation by annotation approach to music structure analysis as well as music structure analysis in general. This chapter ends with an comparison of machine learning versus symbolic approaches to music structure analysis as well as artificial intelligence problems in general.

Chapter 2

Related Work

In this chapter I will give an overview of applications within branches of music information retrieval that are related to music structure analysis. Two machine learning architectures are introduced. Then previous work of the distance-based segmentation and annotation approach is reviewed and the current state-of-the-art to music structure analysis is showed. This chapter ends with an overview of datasets and acoustic features that can be used for music structure analysis.

2.1 Machine learning in Music Information Retrieval

Although machine learning has not been applied on a wide scale to the SbA approach to Music Structure Analysis, it is already being used and researched upon within Music Information Retrieval (MIR) and music generation. Because Music Information Retrieval involves many disciplines regarding music that extend well beyond the scope of this thesis, as does music generation, I will focus on applications of machine learning closely related to Music Structure Analysis. See *Fundamentals of Music Processing* [59] for an extensive introduction to basis and main disciplines of Music Information Retrieval.

2.1.1 Machine Learning for Automatic Audio Segmentation

One of the fields in which machine learning has been (successfully) applied, is classifying pieces of audio in an audio stream. The classes to be distinguished are *speech*, *music* and various kinds of *background noises* or *silence*. Each class can be made as specific as desired; one could be interested in finding only the parts within an audio stream where the same person is talking, or where only music is played. This is often referred to as *Automatic Audio Segmentation* (AAS).

Theodorou, Mporas, and Fakotakis [78] gives an overview of different approaches and implementations to automatic audio segmentation. They describe a distance-based approach similar to the distance-based segmentation and annotation approach to MSA. One of the downsides of this approach is that it does not

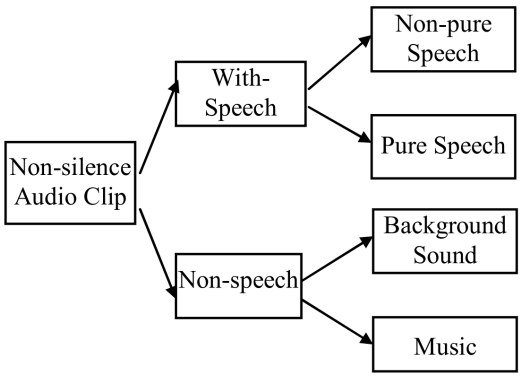


Figure 2.1: Example multi-class classification tree for multi-class broadcast domain data classification or *Automatic Audio Segmentation*.
Reprinted from Lu, Li, and Zhang [53] (Figure 2)

classify each segment, because it is limited to solely finding the segment boundaries. One other approach they describe is the Segmentation by Annotation approach. Within automatic audio segmentation, this approach works in a similar way as the SbA approach to music structure analysis, by subdividing an audio stream into small pieces and assigning a class to each small piece. Theodorou, Mporas, and Fakotakis mention this approach as being specifically suited for machine learning, because of the high performance of machine learning models on classification problems.

First uses of machine learning used for automatic audio segmentation used machine learning methods that did not employ deep learning, such as Gaussian Mixture Models [58, 41, 82, 12, 18], Support Vector Machines [66, 53, 81, 18, 64, 50] or Decision Trees [64, 12], often combined into a hybrid model with a Hidden Markov Model.

The main idea of these models is to first make a distinction between *speech* and *non-speech* using a certain classification model and than use other classification models to make further distinctions within speech like *pure speech* or *silence* and non-speech like *pure music* or *background noise* (Figure 2.1). The Hidden Markov Model in this context is used to combine the output of multiple models trained on different labels and assign the final label to the audio sample.

A recent report of more advanced machine learning methods, such as deep learning, being used within Automatic Audio Segmentation is from Gimeno et al. [28]. They describe an implementation of the SbA approach to automatic audio segmentation using a recurrent neural network, namely bi-directional long short-term memory artificial neural networks. Their aim was to make use of this architecture with multiple different configurations to find the best model. These different configurations consisted of different pooling layers between multiple long short-term memory layers and different input feature combinations.

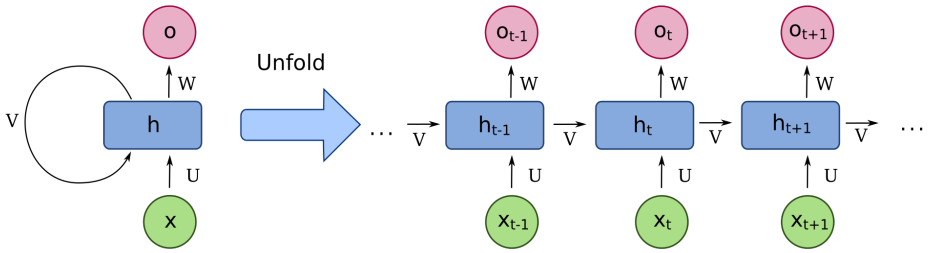


Figure 2.2: Basic structure of a fully recurrent neural network.

Recurrent Neural Networks

A recurrent neural network (RNN) was used because of its capabilities in processing temporal sequence data. This is because this type of artificial neural network has a recurrent architecture, meaning that the output of the previous sample is combined with the input of the current sample [20]. The connections of the neurons can be compared to a directed graph. Due to this connectivity, the internal state of the network at a certain point in its training or inference phase can be called memory, because of the still existing output of the previous sample. However, a downside of this architecture is that the previous sample is very present in the 'memory' while earlier samples are increasingly less present. These fully recurrent neural networks therefore have very good short term memory while lacking long term memory (figure 2.2).

Back-propagation (the algorithm used to train artificial neural networks) now has to be performed over time, this is called back-propagation through time (BPTT) [35, 67, 80]. This means that the output of each time step needs to be tracked, which can become quite unwieldy. Elman [22] found a way around this problem by truncating an unfolded fully RNN to just one time step. This way normal back-propagation can be used again for time sequence data. Elman networks are therefore also called simple recurrent (neural) networks (SRNN). Another type of SRN are Jordan networks [39].

(Bi-directional) Long Short-Term Memory Networks

As mentioned earlier, SRNN and FRNN suffer from very very good short-term memory while lacking long-term memory. To account for this Long Short-Term memory units were introduced [35]. These units were meant to replace the 'normal' neurons in a FRNN. LSTM units are different from normal neurons by its inner structure. Normal RNN units work like neurons in a multi-layer perceptron, however instead of only adding up all inputs (or outputs from previous layer), previous outputs are also added to this sum.

A LSTM unit contains an internal state which acts as its memory. An input,

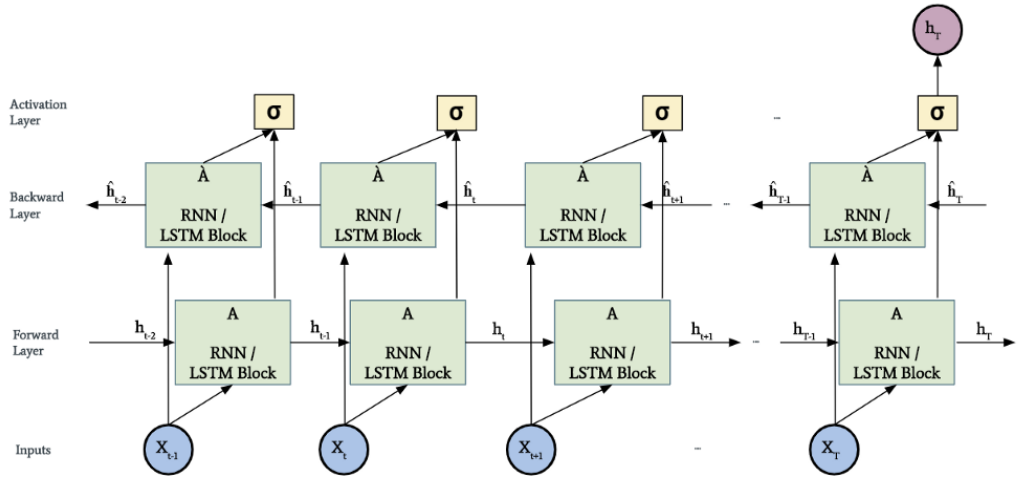


Figure 2.3: Example Bi-directional long short-term memory neural network with an output state per input and a single output after all inputs are processed.

output and forget gate decide how the internal state is changed. The way these are set up enables a LSTM to 'remember' important data and 'forget' less important data of all the data or current input. Li and Wu [48] found these properties especially suitable for Speech Recognition ('forget' noise, 'remember' phonemes), while Sak, Senior, and Beaufays [68] used these properties for acoustic modeling, because of its performance on time sequence data.

A LSTM network can receives a series of input. The network can then return one output per input sample (sequence-output) or return one output once all inputs are processed (single-output). If one is interested in predicting the word someone is typing, sequence-output can be used. If only the next word needs to be predicted after someone is done typing the previous word, single-output can be used.

A bi-directional (recurrent) neural network, first introduced by Schuster and Paliwal [72], is a special kind of recurrent neural networks. A B-RNN works by not only using the input data up to a certain frame, but also the input data from the end to that certain frame. Training thus is performed in both positive and negative time direction. Using both time directions during training enlarges the context of a recurrent unit and was therefore proven to increase the performance of a model. An example of a single-output bi-directional LSTM network can be found in Figure 2.3.

Going back to Gimeno et al.; by using a bi-directional RNN with LSTM units, they reported a relative improvement of 19.72% and 5.35% compared to the best results in the literature at that moment for their two datasets (Albayzín 2010 and 2012) respectively.

Applications of AAS models

Many applications of models capable of performing such segmentation and classifications are within the broadcast data domain. For example in a live radio broadcast these models can be used to automatically apply the right type of audio filtering onto the broadcast audio stream. This could be an audio filter focused on higher tones when someone is talking or an audio filter focused on lower tones when music is played. Also automatically distinguishing when a person's voice is transmitted through a microphone or background noise, can be used to automatically cut the audio feed of a microphone during a live broadcast.

Non-real time applications of these models are within audio indexing and retrieval of for example documentaries, podcasts or other broadcasts. Audio streams automatically segmented by these models can then be used to speed up the process of retrieving certain parts from documentaries or cut all parts where nobody is talking in a podcast.

By modifying these existing models and extending them from multi-class **audio** segmentation to multi-class **music** segmentation will also benefit this field by giving it more specific classes within the music class. Although these models aim to find different kind of patterns within the features used[1], the basic principles can be used to decrease the effort needed to create a good performing music segmentation model that uses machine learning.

2.2 Convolutional Neural Networks in Music Information Retrieval

Another type of machine learning commonly used in another field of computer science and adapted to be useable in Music Information Retrieval are Convolutional Neural Networks (CNN). CNN's, first introduced by LeCun et al. [44] for recognizing handwritten postal codes, and later most improved by Krizhevsky, Sutskever, and Hinton [42], is a type of artificial neural network that are specialized for processing –most often classifying– data that has a known grid-like topology. This, as were its first applications, generally is image data, because images have a very well-defined grid-like topology. While binary images being the most simple form of two-dimensional images with for each pixel only 2 possible values, more complex images, like multi-channel images (red-green-blue images, cyan-magenta-yellow-black images) or multi-view images (top view, bottom view, etc,) can be processed too, albeit in a slightly different way.

Convolutional networks are inspired by the neural architecture found in the human visual cortex by Hubel and Wiesel [37] in the V1 and V2 area [49]. Each neuron in the V1 area is sensitive to a sub-region of the visual field, called a *receptive field*. The activation of each neuron then depends on the occurrence of certain visual features (like edges in certain orientations) in the receptive field.

[1] I will elaborate on this in the Acoustic Features section below.

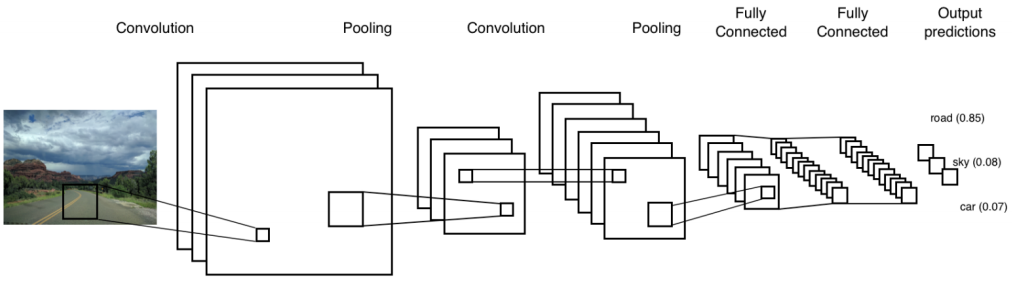


Figure 2.4: Basic structure of a convolutional neural network.

The receptive fields are tiled to cover the entire visual field. The current way to imitate this receptive field is to use convolutions, hence the name of this type of artificial neural network. A convolution, or kernel, is a filter that is applied on an image. This way, each neuron in the next layer is connected to a sub-region in the output or image of the layer before (Figure 2.4). During training, neurons of deeper convolutional layers learn to adapt their weights in such a way that the neuron activates on certain patterns found in the (convolved) image from the layer above, similar to how the neurons in the different visual areas in the visual cortex work.

Feature vectors at first don't seem to be multi-dimensional 'images' however when put in a time sequence one can construct a two-dimensional matrix from multiple one-dimensional feature vectors. This principle makes it possible to also apply convolutional neural networks to natural language processing (NLP) [52] and music information retrieval. In the case of NLP, each word in a sentence is first converted to a feature vector by either embedding it [27, 57], or using one hot encoding. The feature vectors of multiple words concatenated then can be used as two-dimensional input for a CNN, slightly imitating the way N-grams work [11].

Basili, Serafini, and Stellato [3] were the first to apply machine learning on automatic genre classification of songs. However, they only used Naïve-Bayes, Voting Feature Intervals, Decision Trees and Nearest-Neighbors classifiers. Li, Chan, and Chun [47] actually were the first to use convolutional neural networks for automatic genre classification. They were therefore the first to use convolutional neural networks in music information retrieval, and make use of the one-dimensional feature vector concatenation into two-dimensional 'images'. Although they reach over 85% accuracy when using a dataset with only 3 genres, their accuracy heavily decreases to under 30% when they use a dataset with 6 genres.

2.2.1 Convolutional Neural Networks for onset detection

The next subdomain of MIR –related to MSA– where convolutional neural networks are used with great success, is within musical onset detection.

Musical Onset Detection is the process of automatically extracting the point of time of the musical onsets within a piece of music. A (musical) onset is the

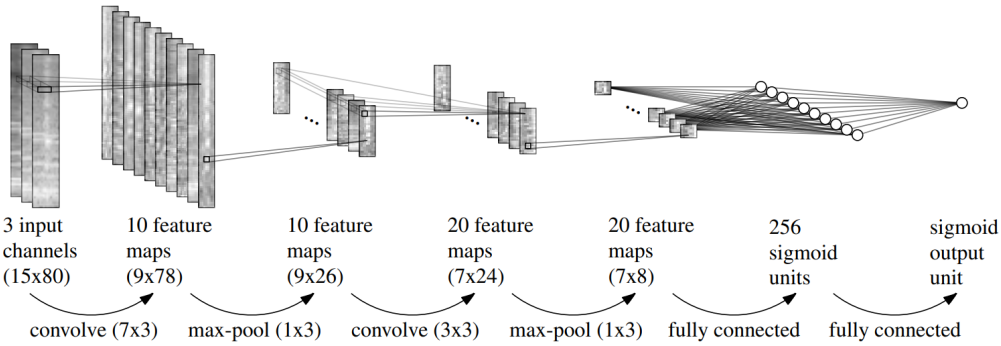


Figure 2.5: Final CNN created for onset detection, starting from a stack of three spectrogram excerpts, ending with one output unit indicating the chance of an onset.
Reprinted from Schlüter and Böck [69] (Figure 2).

beginning of a musical event, most often the beginning of a musical note, but may also be other musical events [4, 83]. It was first introduced as a contest by Paul Brossier and Pierre Leveau for the MIREX 2005 [2]. From this moment it has been an annually recurring contest of the MIREX.

Again, using machine learning was on equal performance or outperformed state-of-the-art at that time [23]. However, in contrast to the current models at that time, this new approach was able to perform real time onset detection by actually predicting the onsets. The proposed model, in multiple ways improved by Böck et al., using a recurrent neural network with long short-term memory (LSTM) units produced a 0.840 F-measure on real time onset detection compared to 0.826 and 0.866 F-measure on non-real time onset detection by state-of-the-art models at that time [7].

After this success with the application of recurrent neural networks for onset detection, Schlüter and Böck [70] propose an onset detection method using a convolutional neural network. Not only did this model perform better than the (improved) bi-directional recurrent neural networks at that time (0.885 F-measure compared to 0.873), it also required less (manual) pre-processing, since CNN's are able to 'learn' this pre-processing in their first layer(s). A year later Schlüter and Böck [69] report improvements made on the initially proposed model, bringing its F-measure above 0.9 to 0.903 respectively (Figure 2.5). Schlüter and Böck explain the high performance of CNN's on onset detection by their accuracy in finding oriented edges in images. Musical onsets show up as 'edges' in spectrograms of a piece music and a CNN is therefore perfectly suited for finding these edges and thus the onsets[3].

After these successes with the application of CNN's within musical onset detection, Ullrich, Schlüter, and Grill made the link to musical structure analysis

[2] See the MIREX website for their proposal. (link)

[3] In practice this is a lot less trivial, however Schlüter and Böck [69] have put great effort into trying to explain it.

and its similarity to musical onset detection. Their efforts to also apply convolutional neural networks on this music information retrieval problem resulted in a new state-of-the-art for music structure analysis, which I will describe below.

2.3 Music Structure Analysis

Music Structure Analysis, as described in the introduction, is the process of finding important parts in a piece of music, on many possible hierarchical levels. As per my definition I will focus on the functional high level segments in a piece of music.

In many songs, a few of these high level segments have the same function, and are therefore grouped together. This thus resulted in the two approaches to this problem described in the introduction. Due to the many differences between different songs, even within the same genre, it has always been very difficult to assign a function to a segment. First research to this problem therefore initially focused on finding the segment boundaries first and then group similar segments together. Similar segments were then given the same capital letter denoting their similar function without specifying it further (as verse, chorus, etc.).

2.3.1 Distance-based Segmentation and Annotation approach to Music Structure Analysis

> **N.B.** A more in depth overview of the Distance-based Segmentation and Annotation approach can be found in Kuiper [43].

The DSA approach can be summarized into a few different sub-approaches; a novelty-, homogeneity- and repetition-based approach.

Self-Similarity Matrix

All methods rely on the same main principle, a self-similarity matrix (SSM). A self-similarity matrix, first introduced by Foote [25], is a matrix calculated by calculating the distance of each feature vector to the other feature vectors. Lower distance values mean that the two feature vectors compared are more similar to each other than two feature vectors with a higher distance value. This will result in a $n \times n$ matrix, where n is the amount of feature vectors of the song. The distances are then normalized to a similarity value. If the distance between two feature vectors is equal to 0, their similarity is equal to 1, all other distance are normalized between a similarity value of 0 and 1. From the way self-similarity matrices are calculated, a diagonal line of similarities with a value of 1 will be seen. This line represents the similarity between each feature vector and itself, which is obviously 1.

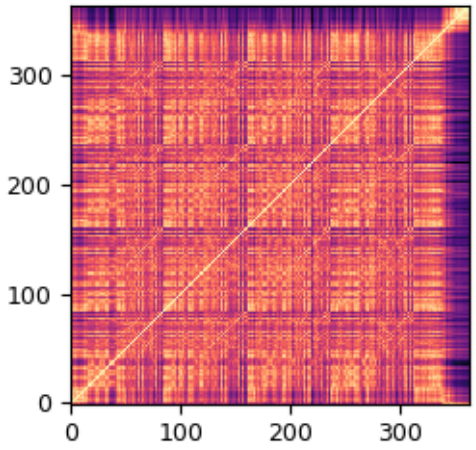


Figure 2.6: Example of a Self-Similarity Matrix, created on *Mrs. Robinson* by Simon & Garfunkel.
Reprinted from Kuiper [43] (Figure 2.2a).

Repetition-based Approach

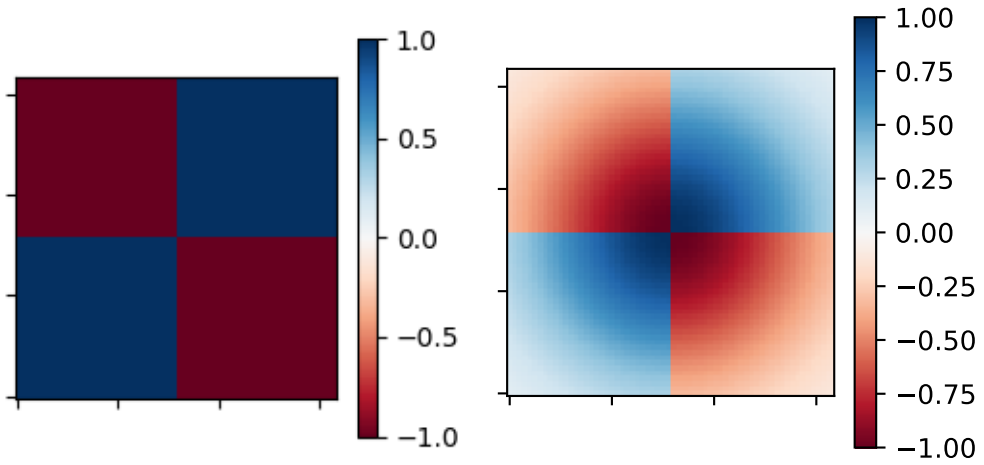
The repetition based approach to DSA makes use of this property of SSM's by detecting more diagonal lines in the SSM. Another diagonal line means that a certain part of the song is repeated elsewhere. The start and end of these repetitions can than be used to determine the location of a segment boundary. One technique used to do this is called *Structure Features* [73]. A (cyclic) time lag matrix is constructed from a SSM. This time lag matrix shows horizontal or vertical lines (depending on the exact process used), a line shows for the frame the line occurs in, the amount of time (in frames) it takes before that frame is repeated (in terms of the feature used to create the time lag matrix).

Novelty-based Approach

The novelty based approach makes use of another property of SSM's: blocks. A block in a SSM denotes a section of consistent features for the duration of the block. Foote [24] was the first to come up with a way to detect the transition of one block into another along the main diagonal of the SSM. Their method used a (Gaussian) checkerboard kernel. A most basic checkerboard kernel can be constructed using a 2×2 matrix, like this:

$$\mathbf{K} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

It functions similar to how a *Sobel Operator* works in image processing to detect edges. However, where the *Sobel Operator* uses a horizontal and vertical operator to detect horizontal and vertical edges, a checkerboard kernel is specifically designed to detect diagonal edges.



(a) Normal checkerboard kernel (b) Gaussian tapered checkerboard kernel

Figure 2.7: An example of a normal and Gaussian tapered checkerboard kernel. Reprinted from Kuiper [43] (Figure 2.3).

Since this is a 2×2 matrix, only 2 different feature vectors will be taken into account, it is therefore more common in music structure analysis to use larger checkerboard kernels, like 64×64. To give more importance to closer feature vectors, the checkerboard matrix can be tapered with a Gaussian function (Figure 2.7).

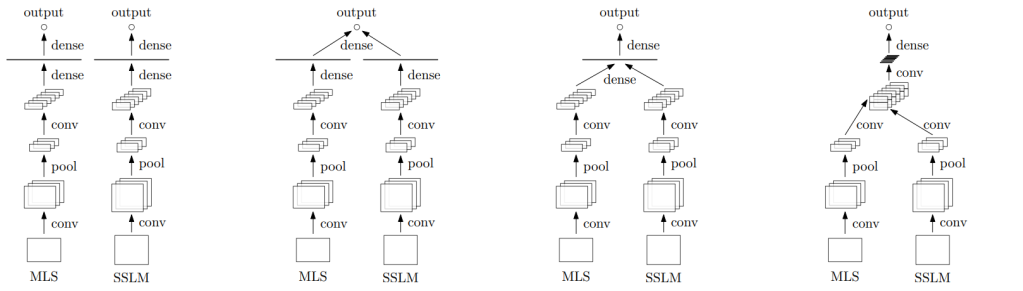
By applying a checkerboard kernel over the diagonal of a SSM, an 'edge activation' can be calculated for each feature vector. From this a *novelty curve* can be constructed, denoting the amount of novelty (in terms of the acoustic features used) between two adjacent feature vectors. By, for example, applying adaptive thresholding, the peaks in this novelty curve can be extracted. The location of each peak then stands for the location of a segment boundary.

Homogeneity-based Approach

The homogeneity-based approach to DSA has been researched a lot less. A most recent research implements this approach using a Hidden Markov Model. Each state stands for a homogeneous piece of music, and the chance to go to the next state determines the chance of a segment boundary.

2D fourier Magnitude Coefficients

Once all segment boundaries are detected, the segments can be extracted. Then all segments need to be labeled. One of the most common ways is to use some kind of clustering method, and then assigning a label to each cluster. Each segment, however, can be of different length. A way to represent each segment in such a way that a distance metric could be applied was therefore needed. The use of



(a) late fusion of outputs (b) fusion in output unit (c) fusion in hidden dense layer (d) fusion in convolutional layer

Figure 2.8: Four different CNN architectures for combining two input features. Reprinted from Grill and Schluter [29].

the magnitude of the 2D Fourier Transform, originally a technique used in image processing, was therefore first introduced by Ellis [21]. Nieto and Bello [61] were the first to apply the 2D Fourier Magnitude Coefficients for segment clustering.

Clustering segments this way proved to be quite accurate. One big problem however, was the amount of clusters that had to be used, since this could vary per song.

2.3.2 Current State-of-the-Art

Due to the successes of CNN's within onset detection and its similarity to finding the segment boundaries –the start of a new high level segment can be compared to a strong onset viewed in a big time window– , Ullrich, Schlüter, and Grill therefore put their effort into creating a convolutional network that is capable of finding these boundaries, based on their model created for onset detection. They presented their first results in 2014 [79]. They report an advance of state-of-the-art of that time for MSA, F-measure of 0.33 to 0.46 for 0.5s tolerance and F-measure of 0.52 to 0.62 for 3s tolerance.[4] A few different input features were tested (MFCC, Chroma, Mel spectrograms), and finally 5 different models trained on Mel spectrograms were bagged [8] together for their final model.

To account for misses of boundaries between non-local musical cues, such as segment repetitions, Grill and Schluter [29] present an improvement on the initial convolutional network by combining the Mel-scaled Log-magnitude Spectrograms (MLS) with Self-Similarity Lag Matrices (SSLM) as input. They test different models that each combine these inputs at a different moment in the model (Figure 2.8). Their best model, fusing the inputs in the convolutional layer (Figure 2.8d), advanced the F-measure from 0.46 [79] to 0.52.

Grill and Schlüter [30] further expanded this model by dividing the SSLM into a near (14 second time context) and far (88 second time context) variant, each used as different feature combined with a MLS as their input for their models. They

[4] A 0.5 or 3 second tolerance means that a segment boundary is counted as a hit if it lies within 0.5 or 3 seconds of the ground truth. Refer to Kuiper [43] (section 2.2.2) for more elaboration on this subject.

also added another neuron to the output layer giving it two neurons in total. One of these neurons was trained on lower level annotation available in their dataset, while the other neuron was trained on the high level annotation. They show that using two annotation levels increases the F-measure of over 0.3 on 0.5s tolerance.

They present their final model in their MIREX submission for the MIREX 2015 Music Structural Segmentation task [31].

2.4 SALAMI Dataset

Many datasets have been created and used for music structure analysis. An extensive listing with most well-known data-sets can be found on the website of the International Society of Music Information Retrieval (ISMIR). There are a total of 16 data-sets listed there that cover musical structure. Only 5 of these data-sets cover western popular music and feature a song total above 100, these are the INRIA:Eurovision, INRIA:Quaero, QMUL:Beatles, RWC and SALAMI datasets respectively.
Although these datasets contain the annotations, and some of them also the features, of the songs, none of these actually provide the audio of the annotated songs. The only dataset that did provide a link to the audio files is the SALAMI dataset. The SALAMI dataset, or *Structural Analysis of Large Amounts of Musical Information* dataset, is an unprecedented large dataset that contains 2400 structural annotations [75]. This dataset contains, among many other subsets, an *internet archives* subset. The internet archives subset contains the annotations of songs publicly available on the internet together with a link to an mp3 file with the audio.

2.5 Acoustic Features

To represent the audio in a more meaningful way, many features have been used or proposed in the past, in this section I will inspect a few of most popular features that are being used or have been used in music information retrieval and music structure analysis in particular. Further explanation of these features among citations to great other sources about these features can be found in [65].

2.5.1 Chroma

The chroma, or 'color', of a song closely relates to the (often twelve) different pitch classes in music. There are multiple chroma types, each one calculated in a different way to represent each pitch class in a slightly different way. The first step for each chroma type however, is to first create a *spectrogram*. A spectrogram is created by applying the discrete fourier transform (DFT) on a slice, or *window*, of the audio. By repeatedly applying the DFT on the window, while it is being slid or hopped through the audio, one can create a representation of the intensity of each

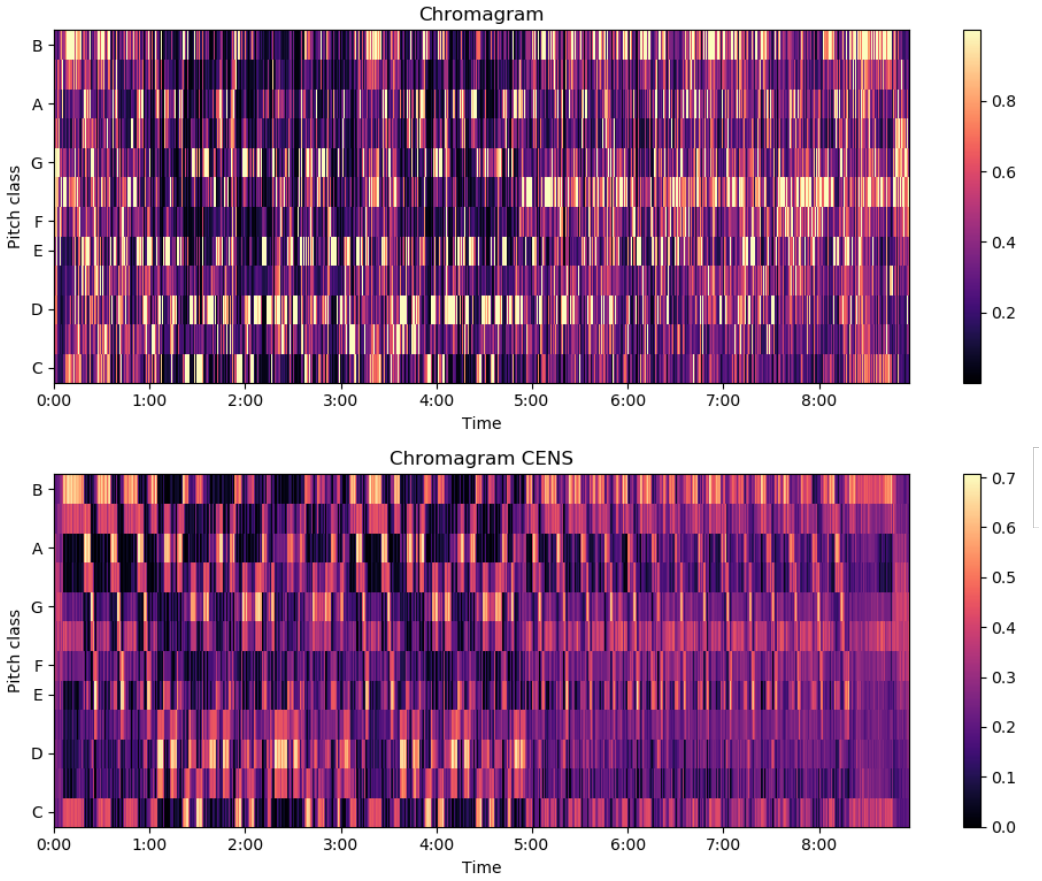


Figure 2.9: PCP chromagram compared to CENS chromagram, both constructed with 12 bins, 2048 FFT window size and 512 hop length.

frequency over time. This technique is called the *Short Time Fourier Transform* (STFT).

The window size in this context is called the Fast Fourier Transform or **FFT window size** (often 2048 or 4096 audio samples). Each time the window is moved, the amount of audio samples it moves is called the **hop length** (often 512 or 1024). If a FFT window size of 4096 is used and a hop length of 1024, one can see that there is 75% overlap between each output of the DFT.

The *Pitch Class Profile* (PCP) [45] is one of the most low-level chroma representations. The STFT spectrogram is converted to an intensity of each 12 pitch classes (C, $C\sharp$, D, etc.) on the equal-tempered scale. If 12 bins are used, each bin represents a semitone, if a multiple of 12 bins are used, each bin represents an equal fraction of a semitone. The PCP has primarily been used to compute the similarity between two songs, however more computation and analysis is needed to extract higher-level patterns from the PCP.

The *Chroma Energy Normalized Statistics*, or CENS, chroma representation [60] is another chroma representation commonly used in audio matching, audio retrieval and music similarity. This feature is more popular in this fields because

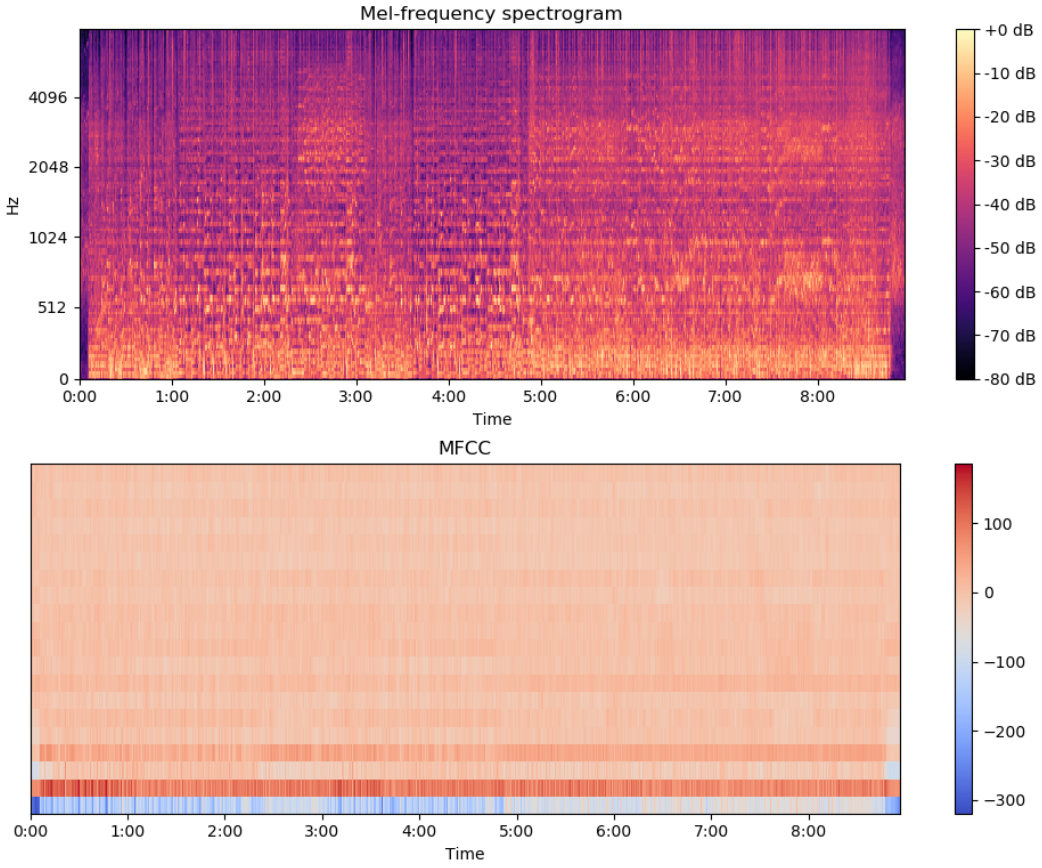


Figure 2.10: Mel-frequency spectrogram compared to Mel Frequency Cepstral Coefficients with 20 bins.

of its robustness to audio dynamics, timbre and articulation. This robustness is obtained by taking statistics over large windows, therefore smoothing local deviations in tempo, articulation and music ornaments such as trills and arpeggiated chords. A downside to this smoothing is that at some points in time it can be hard to determine which pitch class is the most dominant.

A comparison between a PCP chromagram and CENS chromagram can be found in Figure 2.9.

2.5.2 Timbre

Another way of representing audio is by its timbre. Timbre has no direct definition, it however is generally described as *'The perceived sound quality of a musical note, sound or tone'*. Timbre was introduced to distinguish between two instruments, since two instruments playing the same tone, will have the same chroma value.

While the Fourier Transform is able to extract the intensity of each frequency, it has a few flaws. That is why the Mel Scale was created. The mel scale, originally

introduced by Stevens, Volkmann, and Newman [76], is a perceptual scale that scales each pitch judged by listeners to be in equal distance from each other. They introduced this scale because the human auditory system is not equally sensitive to each audio frequency, for example the human auditory system is most sensitive to the 2000 to 5000 Hz range [26], while the screams of a baby are around the 3500 Hz region.

Using the mel scale we can create so-called Mel-scaled spectrograms, or cepstrograms, that represent the intensity of each mel-frequency at each audio sample. A cepstrogram does represent the different sounds quite good, especially since the vector length at each time point is a lot longer than 12 (or a low factor of thereof). However, due to its enormous dimensions, using a cepstrogram as feature for a model can be quite computational intensive.

Partly for this reason, the *Mel Frequency Cepstral Coefficients* (MFCC) were created [51]. The MFCC discretize a mel-scaled spectrogram by first taking the Fourier Transform of an audio stream. Then, the powers of the spectrum is mapped onto the mel-scale, using triangular overlapping windows. The log of the power of each mel frequency is taken, followed by a discrete cosine transform over the list of mel log powers. Often 20 bins are used for the final vector length of the MFCC. A comparison between a mel-scaled spectrogram and the MFCC of that same spectrogram can be found in Figure 2.10.

Although its high information density, MFCCs are not 'the ultimate feature to describe all audio' [63]. Therefore, other timbre features still need to be considered, one example being the *Constant-Q Transform* (CQT) [9]. The CQT is very closely related, but is calculated in a slightly other way. However, due to the complex calculation, another way of calculating the CQT using the FFT in conjunction with a kernel was proposed [10, 6].

Rythm

Not only the tones or harmony of a piece of music can be used as input for music information retrieval models, rhythmic features can also add a lot of information. This is especially applicable to the detection of the musical structure. Similar segments often employ similar rhythmic features, while different segments, like verse and chorus, can differ a in their rhythmic. Rhythmic features therefore can be used to both label or cluster segments. or to detect boundaries between two segments (when a sudden change in a rhythmic feature is detected).

One of the most well known rhythmic features will be tempo. Tempo can be represented as the amount of beats per minute (BPM). Tempo represented on a time scale is called a *tempogram*; the feature vector at each time point represents the probabilities of a certain BPM at that time point.

Since detecting the tempo at each time point turned out to be quite difficult, cyclic tempograms were introduced [32]. A cyclic, or autocorrelation, tempogram detects tempi differing with a power of two, thereby reducing the amount of possible tempi at a certain time point, and thus increasing the probability value of the most probable tempo, since there will be more tempi differing with a power of

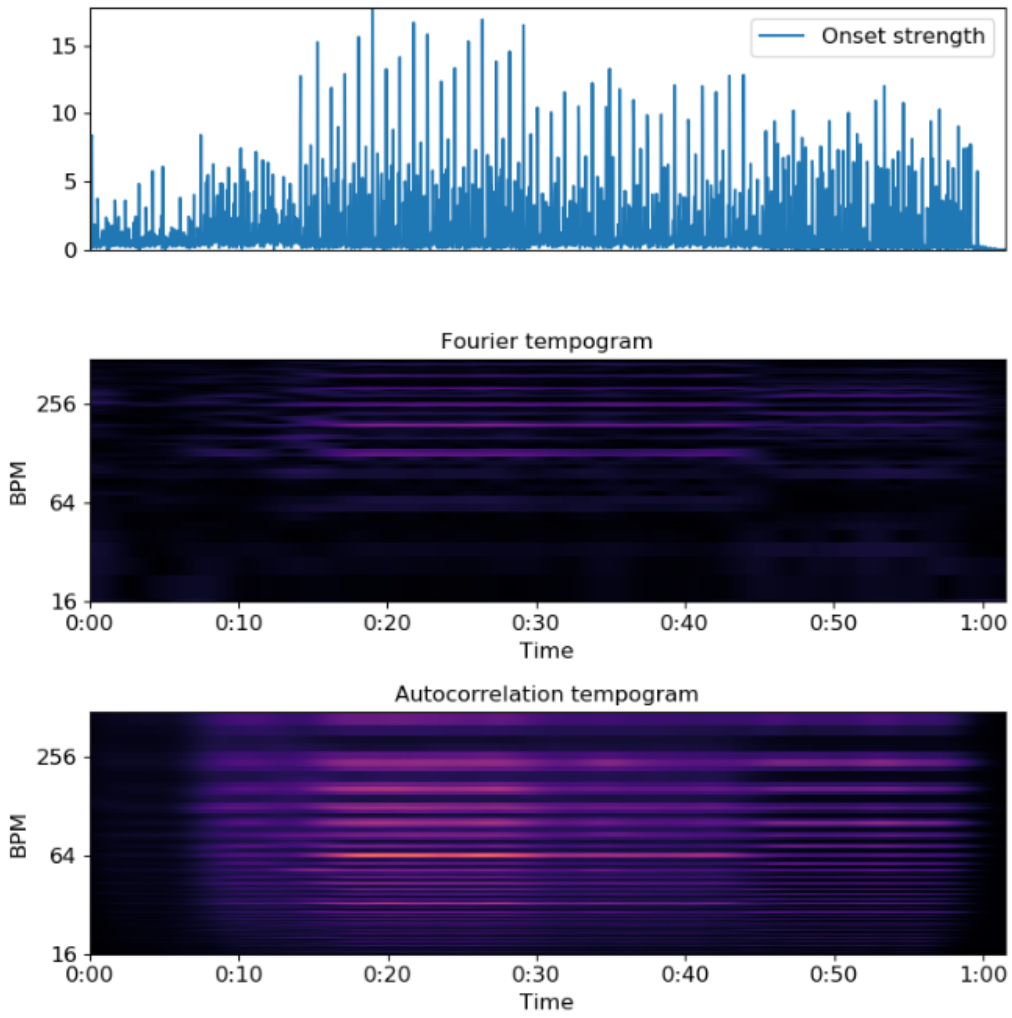


Figure 2.11: A comparison of a Fourier tempogram and autocorrelation (cyclic) tempogram made on the same onset strength envelope.

two around the true tempo.

Another way of computing a tempogram is by using a (short-time) Fourier transform on the onset strength envelope. A comparison between these tempograms can be found in Figure 2.11.

Chapter 3

Methodology

This chapter contains an extensive explanation of the dataset and the labels and acoustic features that were extracted from it. I then introduce the architectures of the models I have created.

3.1 The Data

As explained in the SALAMI Dataset section, I will be using the internet archives subset of the SALAMI dataset as data for the models I will create. Another benefit of using this dataset, apart from the amount of publicly available songs, is that the SALAMI dataset, and the internet archives subset in particular, have been part of the evaluation datasets for many MIREX Music Structural Segmentation contests. The results produced by the models I will create will therefore be quite comparable to the results reported in the many MIREX submissions to this specific MIREX contest.

The internet archives subset consists of two parts: A CSV file containing all metadata about each song and the actual annotations. Among this metadata is a web-link to the audio file of that song in mp3 format. Using these links and the SONG ID given to each song, each audio file was downloaded and saved as *[song-id].mp3*.

All annotations for a song are stored in a folder named by the ID of the song. Since one or two annotators have annotated each song, this folder may contain one or two text files. Each line of text in these text files stand for a segment; the first number denote the time point (in seconds) of the start of the segment. Then the labels describing the function of the segment follow. Because two levels of annotation were used, the collection of labels may contain one or more items.

The first item in the collection of labels always is a lowercase letter describing the low level function of the segment, often written in combination with a *'*. Segments with the same label are musically similar to each other, the *'* means that although the (musical) function between the segments is similar, they differ slightly in musical terms (e.g. key, mode).

Some collections of labels contain two more labels, the high-level annotation

(denoted by an uppercase letter) and the segment function (chorus, verse, etc.). All time points without high-level annotation and segment function implicitly have the most recent high-level annotation and segment function that occurred.

Apart from the text files containing all annotations of one annotator, the folder with annotations also includes a folder for each annotator. In this folder there are multiple text files each one containing one type of annotation, thus one text file contains the low-level annotations, another one contains the segment functions, etc.

The exact instructions given to each annotator on how to annotate a song, and how to format their annotations can be found in the annotator's guide (link).

Using the time points and labels of each segment boundary I will be able to evaluate the accuracy of each model to predict the start of each segment within a 0.5 second and 3 seconds tolerance, I will further explain the exact evaluation method in the Evaluation Method section. The full list of songs used can be found in Appendix C.

Label extraction

To not have to manually parse each text file I have made use of the formatted annotations in Json Annotated Music Specification (JAMS) format [38]. These formatted annotations are available in the Music Structure Analysis Framework (MSAF) [62].

For each song in the internet archives dataset I've chosen the first annotation. This reduced the amount of different annotators to under 5, thereby decreasing the amount of ambiguity between the annotations of different songs. Still, there were quite a lot of different labels present in the dataset (26), of which 10 labels have around or under 10 occurrence within the 4000 segments present in the first annotation of each song in the internet archives dataset.

By decreasing the amount of unique labels, I further decreased the amount of ambiguity between the annotations and presumably increased the accuracy of the models. I've based the grouping on occurrence of the unique label in the dataset, its musical function and the label of another annotator for the same segment. By also listening to a few segments of each of the labels that had a low occurrence in the data, I could also use my own judgement of the relation between a certain label and its associated sounds for the grouping of the labels.

One example are the segments that were labeled as *instrumental*. These segments turned out to be musically equal to *solo* in the context of the data. Since the data primarily consists of live recordings of (alternative) Western Popular Music, both meant a sole guitar (supported by drums) without any lyric. I therefore chose to group these labels together as *solo*. For the other label groupings a similar process was performed as well, while the other factors described were taken into account as well. The final grouping of each label can be found in Table 3.1.

Once each raw label was converted to its grouped label, a one-hot-encoding vector of each label was created. A one-hot-encoding vector of a beat is a vector with the length of the amount of labels containing only zeros except for the index

Label	Occurrence in data	Grouped Label	Occurrence in data
silence	446	**silence**	446
no_function	483	**no_function**	505
applause	12	no_function	
stage_sounds	6	no_function	
spoken	3	no_function	
crowd_sounds	1	no_function	
intro	243	**intro**	300
head	57	intro	
verse	718	**verse**	726
pre-verse	7	verse	
voice	1	verse	
interlude	189	**interlude**	249
transition	51	interlude	
break	9	interlude	
solo	514	**solo**	717
instrumental	160	solo	
theme	39	solo	
main theme	4	solo	
chorus	655	**chorus**	740
pre-chorus	61	chorus	
post-chorus	24	chorus	
bridge	106	**bridge**	107
build	1	bridge	
outro	132	**outro**	182
coda	48	outro	
fade-out	2	outro	

Table 3.1: Grouping of all 26 unique labels into 9 main labels.

of the true label for that beat. The indices of the grouped labels are 0 to 8 for *silence, no_function, intro, verse, interlude, solo, chorus, bridge* and *outro* in that specific order respectively.

3.2 Feature Selection

Feature Extraction

I used the LibROSA Python package for music and audio analysis [56] to extract the features from each audio file. I have used a hop length of 1024 together with a FFT window of 4096 for 75% overlap within each feature vector.

First the beats were extracted using the method introduced by Ellis [21]. Then the Chroma Energy Normalized Statistics (CENS) [60] chroma variant was ex-

Feature	Vector Length	Matrix Size
CQT	84	84×4
CENS	12	12×4
PCP	12	12×4
Tonnetz	6	6×4
MFCC	14	14×4
Tempogram	192	192×4

Table 3.2: Vector length and matrix size of each feature extracted.

tracted. As low level harmonic/chroma features I extracted the Pitch Class Profile (PCP) [45]. I saved the PCP as self-contained feature and used it to create the Tonnetz features [33]. I also created a Mel Spectrogram [76] and used it to create the Mel-Frequency Cepstral Coefficients (MFCC) with 14 coefficients [51], these represent the timbre of the song. As other timbre feature I extracted the Constant-Q Transform (CQT) [9], using the technique described by Schörkhuber and Klapuri [71]. Because often a segment boundary goes hand in hand with a change in tempo (e.g. a short speedup in tempo, or the next segment is in a lower or higher tempo), I extracted the tempogram of each song with a window length of 192 [32].

The resulting feature vectors have a length of 84, 12, 12, 6, 14 and 192 for the CQT, CENS, PCP, Tonnetz, MFCC and tempogram feature vectors respectively (Table 3.2).

Data Reduction

Each feature was first extracted based on the frames of a song. The amount of frames of a song is calculated by the sample rate (amount of audio samples per second) multiplied by the length of the song (in seconds) divided by the hop length. Each song is converted to Waveform (wav) audio format with a sample rate of 22050, this combined with a mean song length of 4 minutes or 240 seconds and a hop length of 1024 for each feature results in $(240 \cdot 22050)/1024 = 5168$ frames on average for each song.

To reduce the amount of feature vectors I have beat synced each feature. The means that for each beat in a song there is one vector for each feature. This vector is calculated as the average of all vectors of that feature within the beat. With an average beats per minute (BPM) of 120 for all songs (thus 2 'frames' per second), the average amount of vectors per song is now reduced to $240 \cdot 2 = 480$. With a total of 377 songs (after data cleaning) I gathered a total of about 180000 vectors per feature.

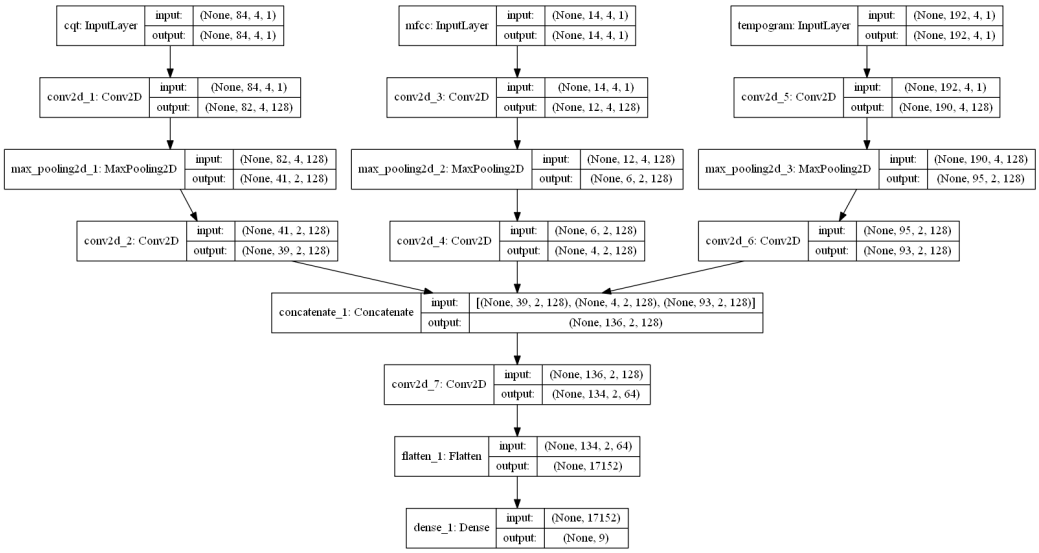


Figure 3.1: Proposed CNN architecture with CQT, MFCC and Tempogram as example inputs.

3.3 Proposed Architectures

3.3.1 Preserving Temporal Aspect of Music

To obtain faster learning speeds of the models, the data needs to be shuffled; if all beats of a song are kept in the initial order, the output labels will be sequences of the same label. Even when the batch size is increased, samples with the same output label are still being fed to the network in each learning iteration, thus lowering its capabilities to learn multiple output labels in one learning iteration.

One problem of shuffling the data however is that one beat alone will not be enough for a network to learn all patterns that can be associated with an output label, thus requiring the temporal aspect of the song to be intact. My proposed solution to this problem is to, instead of using a single vector per feature per beat, combine the vectors of two beats before the current beat, the vector of the current beat and the vector of the beat after the current beat into a matrix for each feature. The results is therefore a matrix, with a shape of $length\_feature \times 4$, per feature per beat.

In the next sections I will explain, per proposed model, how I use this matrix as input.

3.3.2 Convolutional Artificial Neural Network

Because of the great results with convolutional neural networks in the music structure analysis field, I propose an implementation of the segmentation approach using a convolutional neural network.

This CNN has multiple input layers, one matrix per beat for each feature used. Each input matrix will therefore act as the input image, as described in the CNN

in MIR section. After each input layer there is a two-dimensional convolutional layer with a 3×1 kernel, followed by a possible two-dimensional max- or average-pooling layer with a 2×2 pooling size. Then another two-dimensional convolutional layer follows with 128 neurons and 3×1 kernel.

The outputs of this last convolutional layer of each input is then concatenated to one big $n \times 4$ matrix. n represents the summed length of the first dimension of each output shape. The concatenated outputs are then fed into a last two-dimensional convolutional layer, with 64 neurons and 3×1 kernel.

The (two-dimensional) output of the final convolutional layer is flattened into a single vector, and used as input for a dense layer with 9 neurons (each neuron representing one output label) and *softmax* activation function. The softmax activation function normalizes the outputs of each neuron into a probability distribution, thus each output is scaled to be within the [0,1] interval and all scaled outputs sum to 1. The highest output is then selected as predicted label for the input, with its activation as confidence or probability that this label is the true label.

An example model, with CQT, MFCC and Tempogram as input features can be found in Figure 3.1.

3.3.3 Bi-Directional Long Short-Term Memory Artificial Neural Network

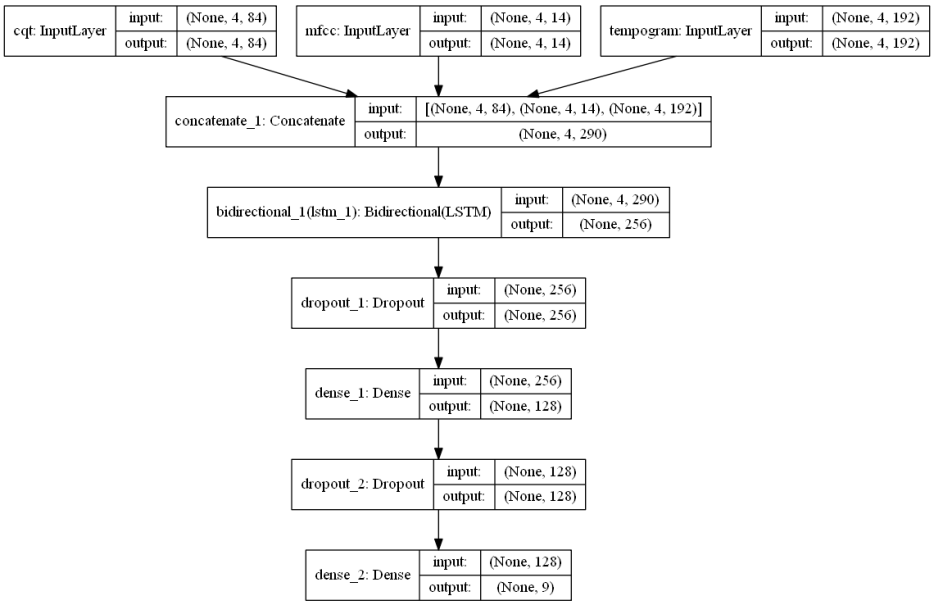
Alongside a convolutional neural network, I also propose a bi-directional long short-term memory neural network implementation of the segmentation by annotation approach. This model will show if its capabilities to cope with temporal data and its performance in the automatic audio segmentation field also apply to music structure detection. The model is made bi-directional because of the reported increased performance of bi-directional recurrent neural networks over normal recurrent neural networks [72].

I propose two different architectures using long short-term memory units, a model with one bi-directional LSTM layer (Figure 3.2a), and a model with a double bi-directional LSTM layer, optionally combined with a max- or average-pooling layer (Figure 3.2b). Both models first contain an input layer per feature used, similar to the proposed CNN. In contrast to the CNN, all input is immediately concatenated into one matrix (with shape $n \times 4$).

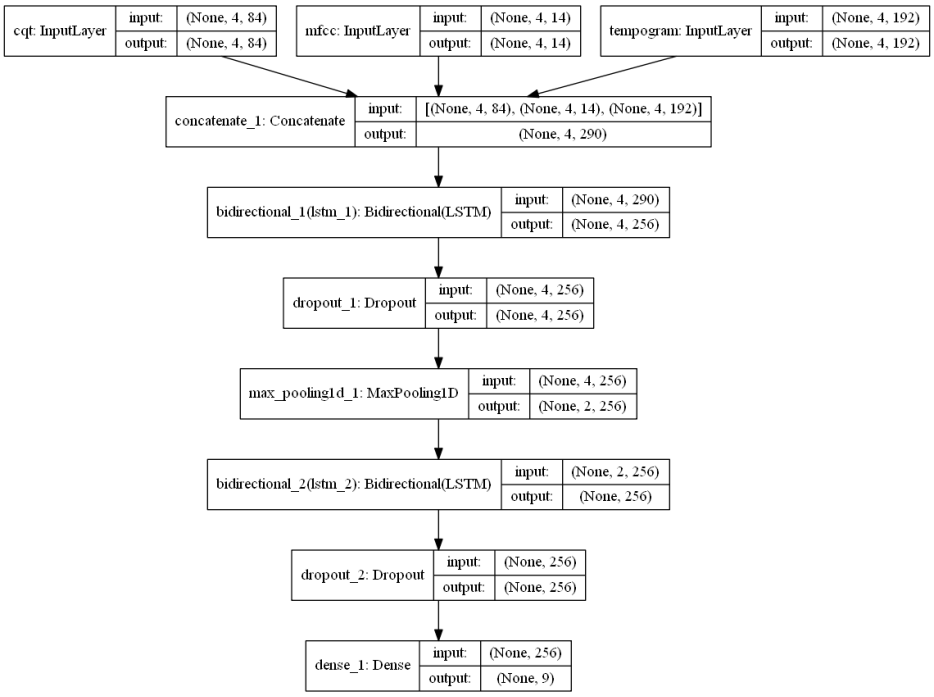
When a single LSTM layer model is build, this matrix is fed into bi-directional LSTM layer. However, since an (B-)LSTM layer has only one dimension, each column (or vector representing a beat) is separately evaluated. Because we want a single output from the full network, the B-LSTM layer is set to put one vector out after all 4 beats have been evaluated. This output is then passed into a dense layer, containing 128 neurons, whose output ultimately is put in a dense layer containing 9 neurons (each one representing one output label), and a *softmax* activation function (Figure 3.2a).

If a double LSTM layer model is built the first B-LSTM layer does not first

evaluate all 4 beats, but gives an output for each of the 4 beats. Each output is optionally passed through a one dimensional max- or average-pooling layer with a pool size of 2. Then another B-LSTM layer receives each output and evaluates all four, before outputting a single vector which is fed into the final dense layer with 9 neurons and softmax activation (Figure 3.2b).



(a) Single LSTM layer architecture.



(b) Double LSTM layer architecture.

Figure 3.2: Proposed LSTM architectures with CQT, MFCC and Tempogram as example inputs.

Chapter 4

Experimental Results

In this chapter I report the initial results from the first test. After that I propose changes to this first test, and report the results from this final test.

4.1 Evaluation Method

In order to find the best model on the dataset, I have created multiple models, each with different hyperparameters and combinations of features as input. I then tried to create a model for each possible combination of hyperparameters on each feature combination. Each model was created using the Keras [14] machine learning framework originally build for Python. This framework is build on top of the popular, well-known machine learning API *Tensorflow 2.0* [55].

4.1.1 The Hyperparameters

For each generally important parameter for a neural network I've created a hyperparameter. I will explain each hyperparameter below. A list of all hyperparameters and their values can be found in Table 4.1.

Hyperparameter	**Values**
Neurons	128, 256, 512
Activation function	*ReLu, Elu*
Pooling	*no_pooling, max, average*
Optimizer	*RMSprop, Adam*
Dropout	10%, 20%, 30%, 40%, 50%
Epochs	1, 10, 50, 100
Batch size	1, 5, 10, 50, 100

Table 4.1: Initial hyperparameters and their initial range of values.

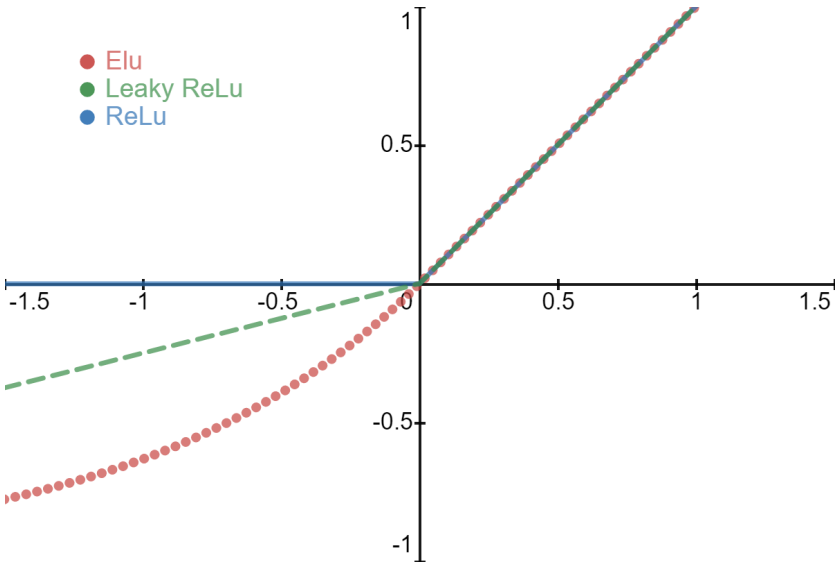


Figure 4.1: Comparison of output of Elu, ReLu and Leaky ReLu activation functions on varying input values. If $x > 0$, all activation functions have a linear output of $y = x$.

Neuron Count

The first hyperparameter is the amount of neurons that make up the first layer, the first convolutional layer for the CNN models and the first or only B-LSTM layer of the LSTM models. I've chosen 128, 256 and 512 as possible neuron counts.

Activation Function

The second hyperparameter is the activation function of each neuron, the chosen activation function is applied on each layer, except for the final dense layer, which has a *softmax* activation function (as explained in the Methodology chapter). I've included *Rectified Linear Unit* (ReLu) [1] and *Exponential Linear Unit* (Elu) as possible values. ReLu was included because it is seen as goto activation function, however because no negative activation is possible with this function, I've chosen to include Elu as activation function as well. Another alternative activation function close to ReLu allowing negative activation is Leaky ReLu, however this activation function was not available in Keras. A visualization of the different activation functions on varying input values can be found in Figure 4.1.

Pooling

The output feature maps of convolutional layers are sensitive to the location of features in the input. One way of solving this problem is to down sample the feature maps. An approach to down sampling is provided in the form of a pooling layer. Two common types of pooling layers are defined: max and average. Max pooling works by returning the maximum value present in the pooling size as

output, while average pooling works by calculating the average. The pooling layers defined in the models use local pooling with a pooling size of 2×2.

Not only is pooling being used in convolutional networks, Gimeno et al. [28] employ pooling between their bi-directional LSTM layers, that is why I have tested no pooling, max pooling and average pooling on both the CNN and LSTM models.

Optimizer and Dropout

Neural networks 'learn' by applying back propagation each learning iteration. During back propagation each weight is modified with regards to the output error. The aim is to reduce the value of the loss function. Because the models I created are multi-class classifiers with one true label, *categorical cross entropy* is used as loss function in the models. Future research has to prove whether using other loss functions may be better instead [36]. To reduce the loss, the gradient is calculated. An optimizer is a function that can efficiently use this gradient to update the weights of the models to decrease the loss as fast as possible. One of the newest, most popular and best performing optimizers of this moment is the *Adam* optimizer [40]. As comparison optimizer I've chosen the *RMSprop* optimizer.

Another way of boosting the learning rate (or speed of which the loss decreases) is to use dropout. Dropout, introduced by Hinton et al. [34], works by randomly 'dropping' a percentage of the neurons of certain layers. This means that all connections to these neurons are removed and that the other neurons are required to 'learn' the same representations that were first captured by the now removed neurons. Too low values for dropout are found to not have much impact, while too high dropout rates will reduce the overall performance of the models. 50% dropout rate is generally seen as a the threshold from which the performance of the models will decrease, but still very much used [79]. I've therefore defined the set of dropout rates to range between 10% and 50% with a step size of 10%. The dropout is applied on each layer except for the last dense layer.

Epochs and Batch Size

The final two parameters are used during the training phase of a model. These are the *epochs* and *batch size*. The amount of epochs defines how many times the training data is evaluated before the final model is returned. Increasing the amount of epochs means that the model has more time to learn all patterns into greater detail, however to high values will cause the model to overfit[1] and severely increases the time it takes to train the model. I've chosen to test models on 1, 10, 50 and 100 epochs (more epochs are being used in the field, this however would've made the training process all the much longer).

[1] If the model sees each sample many times it will become very good in classifying each sample, however new samples (less similar to the learned samples) will get a wrong classification more often.

Feature Combination
CQT, CENS, PCP, Tonnetz, MFCC, Tempogram
CQT, MFCC
CENS, MFCC
PCP, MFCC
Tonnetz, MFCC
CQT, MFCC, Tempogram
CENS, MFCC, Tempogram
PCP, MFCC, Tempogram
Tonnetz, MFCC, Tempogram
CQT, Tempogram
CENS, Tempogram
PCP, Tempogram
Tonnetz, Tempogram
MFCC, Tempogram

Table 4.2: Initial feature combinations.

The batch size determines how many samples are passed through the network before back propagation is applied. The back propagation then is applied with the gradient of the loss of all these samples. Larger batch size generally speeds up the learning process of a model since less updates per epoch are performed. However, if the batch size is too large, a model will not always be able to learn specific patterns for one specific output, thus lowering its precision. I've chosen to create models with a wide range of batch size: 1, 5, 10, 50 and 100.

Feature Combinations

Based on previous research and type of each feature I've created 14 different sets of feature combinations, these can be found in Table 4.2. I've chosen to include a combination of all features extracted, combined with each feature paired with the MFCC, based on its overall high success rate in other research.

I've also included one a few of the possible chroma feature combined with one timbre feature combined with the tempogram combinations. To evaluate the performance of the tempogram I've included the pairing with each other feature to the feature combinations as well. The combinations missing are combinations of different timbre features and combinations of different chroma features, since combining a feature of one type with a feature of the same type will most likely not improve performance.

Multi B-LSTM layer

The LSTM models had one extra hyperparameter: whether to use a single or double B-LSTM layer. If a double LSTM layer model was made, and pooling

was set to max or average, a pooling layer was put between the B-LSTM layers. Otherwise no pooling layer was used.

4.2 Initial Test

4.2.1 Initial Test Setup

Using the possible values of each hyperparameter and the feature combinations an attempt had been made to create a model on each possible combination trained on 80% of the total data (about 150000 samples) and evaluated on the remaining 20% (35000). An output was registered of being correct when the index of neuron with the highest activation in the output layer corresponded with the index of the 1 in the one-hot-encoded true label vector and wrong otherwise. The accuracy was then calculated as the ratio of correct classifications to wrong classifications.

One machine, containing an AMD Ryzen 7 3700x CPU and a NVIDIA GeForce GTX 1080 (8GB) GPU, was used to train and evaluate one CNN and one LSTM model simultaneously. Keras and the Tensorflow 2.0 backend were set to run on the GPU, while data flow was managed by the CPU.

4.2.2 Initial Test Results

Since both a CNN and LSTM model were trained simultaneously one epoch with a batch size of 1 roughly took $30 \sim 50$ seconds. A total of 50400 CNN models and 100800 LSTM models had to be trained. This means that about 6 million epochs of the data needed to be performed. Although only 20% of the models have a batch size of 1, the expected time per epoch still was way above 10 seconds. This means that training and evaluating all CNN models would take 1 year and training all LSTM models around 2 years[2].

Therefore the initial test setup was aborted after 1.5 weeks. In this time 145 LSTM and 292 CNN models were trained and evaluated. These models did not evaluate the impact of the amount of neurons, activation function and optimizer used nor the impact of the dropout. However, I was able to derive the impact of the amount of epochs, the batch size and feature combinations.

After preliminary data analysis, a number of trends were identified. The two trends that stood out the most were the amount of epochs and the batch size. A higher amount of epochs as well as a larger batch size always increased the performance of the model. The CQT turned out to be the best performing feature, with the Tempogram and MFCC being the two best features after that.

The best performing CNN model had an accuracy of 0.66 on the test data, the best performing LSTM an accuracy of 0.43. No real performance increase of using a double B-LSTM layer over one B-LSTM layer was noticed.

[2]Not counting the possible speedup when all CNN models have been evaluated. However, the very optimistic 10 seconds per epoch accounts for that.

Feature Combination
CQT, Tempogram
CQT, MFCC
CQT, MFCC, Tempogram

Table 4.3: Final feature combinations.

4.2.3 Adjustments

To also be able to evaluate the other hyperparameter values, I made some changes to the search space.

Feature Combinations

I first took a look at the feature combinations. The Tonnetz feature turned out to be too small of a feature to be usable with any pooling, and I therefore removed it from the features. Since no improvement was obtained using all features instead of only CQT, MFCC and Tempogram, the CENS and PCP features were removed from the feature combinations as well. This reduced the amount of feature combinations from 14 to 3. Thereby reducing the factor that determined the amount of models that had to be trained and evaluated the most. The final feature combinations can be found in Table 4.3.

Hyperparameters

After the feature combinations I've also reduced the amount of values of some hyperparameters. The most reduced ones are the epochs and batch size, these have been truncated to only contain 10, 50 and 100. Furthermore, I've reduced the dropout rates to 10%, 25% and 50%.

I've also trained the best performing CNN and LSTM on ReLu and Elu activation while using the Adam and RMSprop optimizer. This showed that using Elu did not improve the accuracy, while using the Adam optimizer heavily increased the accuracy with over 10%. I therefore decided to use only ReLu as activation function and Adam optimizer as the only optimizer. As an additional benefit, Adam is a faster optimizer and therefore decreased the time of each epoch with quite a large factor. Furthermore, I removed the double B-LSTM layer hyperparameter, and chose to train only single B-LSTM layer models. The final hyperparameters and their values can be found in Table 4.4.

Final Test

One final big factor that increased the duration of each epoch was the amount of data. To decrease this heavy impacting factor, I've decided to take only 10% of the original data as training and test data for the final test. The amount of samples used for training therefore decreased to 15000 and the amount of beats used for

Hyperparameter	**Values**
Neurons	1285,256,512
Activation function	*ReLu*
Pooling	*no_pooling, max, average*
Optimizer	*adam*
Dropout	10%, 25%, 50%
Epochs	10, 50, 100
Batch size	10, 50, 100

Table 4.4: Final hyperparameters and their final range of values.

testing decreased to 3500. Although overall accuracy of the models were expected to be lower on this subset that on the full dataset, models with better hyperparameters and features will perform better than other models on both the full dataset or a subset thereof. Reducing the dataset too much can result in the training data not containing enough variation. I, however, expect 15000 samples to be quite representative for the full dataset, especially since all samples are chosen at random and not on a song basis. Each sample in this context means the concatenation of the feature vectors of the two beats before the current beat together with the feature vector of the current and the next beat.

The search space was now reduced to just above 700 different models. By using the Adam optimizer, removing the batch size of 1 and reducing the size of the dataset, the average time per epoch also reduced to around 3 seconds. The expected time to train and evaluate these models therefore reduced to less than a week, which was a lot more within the time-scope of this thesis.

4.3 Final Results

After 5 days all models were trained and evaluated. The accuracy of each model was saved in a text file, named by the parameter values and features used to create that model separated by an underscore. All models were then grouped in a list sorted by their accuracy in descending order. I analysed each parameter by grouping all models by their value for that parameter. Then I counted the total amount of models made with that value for the parameter and amount of models in the top 10% of all models (around 700 models were created, top 10% therefore accounted for 70 models). With these counts I calculated the percentage of models with that value in the top 10% of models. This gives a distribution of best performing parameter value on the top 30% of all models. All distributions can be found in Table 4.5.

Using the percentages, a best performing value for each parameter can be derived. This does not immediately mean that the best performing models has these values for each parameter, however further analysis showed that the model with these parameter values and feature combination as input, indeed was among the

	Values	No. models created	No. models in top 10%	%. models in top 10%
Neuron Count	128	267	37	**0.139**
	256	250	18	0.072
	512	218	15	0.069
Dropout	10%	316	27	0.085
	25%	232	28	**0.121**
	50%	187	15	0.080
Pooling	none	242	0	0
	max	259	43	**0.166**
	average	234	27	0.115
Epochs	10	255	0	0
	50	244	20	0.082
	100	236	50	**0.212**
Batch size	10	249	2	0.008
	50	245	27	0.110
	100	241	41	**0.170**
Feature Combo	CQT + Tempogram	202	37	**0.183**
	CQT + MFCC	201	0	0
	CQT + MFCC + Tempogram	200	18	0.090

Table 4.5: Analysis of all CNN models created with different parameter values and feature combinations. Best performing value of each parameter had been put in bold.

top 3 best performing models.

4.3.1 Best performing CNN and LSTM models

Parameter values and input features combination

Following the analysis of the parameter values, I've defined the best performing CNN and LSTM models. These models both have a neuron count of 128 in the first layer, a dropout of 25%, max pooling, a batch size and epoch count of 100, ReLu activation function and Adam optimizer. As input features I've chosen the Constant-Q Transform in combination with the Tempogram. Using the best performing values for each parameter together with all features as input only improved the accuracy with $< 1\%$, while increasing evaluation time. The LSTM model has only one B-LSTM layer, since adding more layers did not improve its accuracy but instead added more time to the training and evaluation time of the model.

Dataset	**Measure**	**CNN**		**LSTM**	
Beat Classification Accuracy	Training data	0.9456		0.8265	
	Test data	0.80		0.77	
		0.5F	**3F**	**0.5F**	**3F**
All songs own ground truth	Untrimmed	0.6849	0.8545	0.4526	0.6141
	Trimmed	0.5627	0.7935	0.3003	0.5065
All songs SALAMI ground truth	Untrimmed	0.1451	0.3397	0.1474	0.3487
	Trimmed	0.0374	0.1960	0.0581	0.2296
SALAMI song-id 1200 own ground truth	Untrimmed	0.6087	0.9565	0.3846	0.6923
	Trimmed	0.5261	0.9474	0.2727	0.6364
SALAMI song-id 1200 SALAMI ground truth	Untrimmed	0.55	0.69	0.38	0.56
	Trimmed	0.48	0.64	0.29	0.50

Table 4.6: An overview of the final CNN and LSTM model evaluated on multiple datasets. The training- and testset accuracy is evaluated using Keras. The F-measures on SALAMI 1200 (custom and original ground truth) as well as the F-measures on the full SALAMI dataset (custom and original ground truth) are calculated using the MSAF. In all cases the same CNN or LSTM model were used, these are the ones trained on the custom ground truth (9 unique output labels).

Evaluation of best models on different datasets

I then trained each best model on the full dataset, with 80% of the dataset as training data and the remaining 20% as test data. The accuracy on the training and test data were 94.56% and 80% for the CNN model and 82.65% and 77% for the LSTM model respectively (Table 4.6 first row). These models were then saved in *.h5* file. This file format is used by Keras to models with their trained weights. This way inference can be easily applied on another machine, since the model only needs to be loaded instead of fully trained[3].

Since the end goal was to segment a song into *chorus*, *verse*, etc. instead of predicting a single beat, I've taken a few arbitrary songs from the dataset. Then I looked at the amount of segments and the amount of different labels. Based on this I've taken SALAMI song-id 1200 as my test song for the final models. To perform the test I've sequentially inferred the label of each beat of the song with each model. As expected there were some beats that had another label than the labels of a number of beats around it. This is due to the less than 100% accuracy of the models on the dataset (which is to be desired, since they would be overfitted otherwise).

Label Filtering Function

To still be able to use the start of a sequence of new labels as start of a segment I wrote a simple filtering function, based on the predictions on SALAMI 1200. This functions works by iterating over all the predicted labels of the beats. When a label different to the label of the beat before it is detected, the next 4 beats are scanned. If the label of the previous beat is detected in the labels of the next 4 beats, that label is taken as filtered label, otherwise the predicted label is taken as filtered label. This function thus requires each segment to be at least 5 beats in length, which is quite common in the ground truth I created based on the label grouping mentioned in The Data section in chapter 3. In the SALAMI dataset this is less common, but I will elaborate on this in the Discussion chapter.

I have also tried another filtering function. This function worked similar to the filtering function I used, however instead of checking if the label of the previous beat occurs in the labels of the next 4 beats, the most occurring label of the next 5 beats was taken as filtered label. The problem of this function was that if there were 3 falsely classified beats among the next 5 beats, this was taken as filtered label, thus causing a short, falsely classified segment to occur in the filtered predictions. This function also required segments of at least 6 beats, 1 beat (about 0.5 seconds) longer than the other filtering function, which can already be quite impactful.

The ground truth, predicted and filtered output of the CNN model on the first and last 15 beats of SALAMI 1200 can be found in Table 4.7, the same output of the LSTM model on the first and last 15 beats of SALAMI 1200 can be found in Table 4.8.

Final Evaluation on SALAMI and own Ground Truth

Using the filtering function I described earlier, I was able to create a *JSON Annotated Music Specification* (JAMS) file [38] of each song from the dataset. I also created a JAMS file for each ground truth I created. I then used the Music Structure Analysis Framework (MSAF) [62] to evaluate the filtered segments of the CNN and LSTM models on both the SALAMI and my own ground truths.

The MSAF returns quite an extensive evaluation report. To enhance overview, I have only included the (mean) F-measures from the reports. Four different F-measures are returned with each report. An F-measure of an evaluation of the segment boundary location with a 0.5 second and 3 seconds time tolerance to the ground truth, both trimmed and untrimmed. MSAF automatically adds a segment boundary at the start and end of a song, if the untrimmed F-measure is calculated, these segment boundaries are included in the calculation. If a trimmed F-measure is calculated, these segment boundaries are excluded from the calculation, often

[3] It can also be used to further train the network on other or more data on another machine and than re-saved as *.h5* file, however this is less common in practice.

lowering the F-measure, since only the segment boundaries produced by the models are now taken into account.

All F-measures can be found in Table 4.6, the full ground truth, predicted and filtered labels of SALAMI 1200 of the CNN here (link), and the same data of the LSTM here (link).

Beat	Beat Start Time	True Label	Predicted Label	Filtered Label
0	1.253877551020408	no_function	no_function	no_function
1	1.764716553287982	no_function	no_function	no_function
2	2.321995464852609	no_function	no_function	no_function
3	2.832834467120181	no_function	no_function	no_function
4	3.390113378684807	no_function	no_function	no_function
5	3.900952380952381	intro	intro	intro
6	4.458231292517007	intro	intro	intro
7	4.96907029478458	intro	intro	intro
8	5.479909297052155	intro	intro	intro
9	6.03718820861678	intro	intro	intro
10	6.548027210884354	intro	intro	intro
11	7.058866213151927	intro	intro	intro
12	7.569705215419501	intro	verse	intro
13	8.080544217687075	intro	intro	intro
14	8.591383219954649	intro	solo	intro
15	9.102222222222222	intro	intro	intro
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
453	238.3760544217687	solo	solo	solo
454	238.8868934240363	solo	solo	solo
455	239.3977324263039	solo	no_function	solo
456	239.9085714285714	solo	solo	solo
457	240.419410430839	solo	solo	solo
458	240.9302494331066	solo	solo	solo
459	241.4410884353742	solo	solo	solo
460	241.9519274376417	solo	solo	solo
461	242.4627664399093	solo	solo	solo
462	242.9271655328798	solo	chorus	solo
463	243.3915646258504	solo	chorus	solo
464	243.8559637188209	solo	solo	solo
465	244.3668027210885	solo	solo	solo
466	244.8776417233560	solo	solo	solo
467	245.3884807256236	solo	solo	solo
468	245.8528798185941	solo	no_function	solo

Table 4.7: First and last 50 beats of SALAMI 1200 true, predicted and filtered labels produced by best CNN model.

Beat	Beat Start Time	True Label	Predicted Label	Filtered Label
0	1.25387755102041	no_function	no_function	no_function
1	1.764716553287982	no_function	no_function	no_function
2	2.321995464852608	no_function	no_function	no_function
3	2.832834467120181	no_function	no_function	no_function
4	3.390113378684807	no_function	no_function	no_function
5	3.900952380952381	intro	no_function	no_function
6	4.458231292517007	intro	intro	intro
7	4.96907029478458	intro	intro	intro
8	5.479909297052155	intro	intro	intro
9	6.03718820861678	intro	intro	intro
10	6.548027210884354	intro	intro	intro
11	7.058866213151927	intro	intro	intro
12	7.569705215419501	intro	intro	intro
13	8.080544217687075	intro	intro	intro
14	8.591383219954649	intro	intro	intro
15	9.102222222222222	intro	intro	intro
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
453	238.3760544217687	solo	solo	solo
454	238.8868934240363	solo	solo	solo
455	239.3977324263039	solo	solo	solo
456	239.9085714285714	solo	solo	solo
457	240.419410430839	solo	solo	solo
458	240.9302494331066	solo	solo	solo
459	241.4410884353742	solo	solo	solo
460	241.9519274376417	solo	chorus	solo
461	242.4627664399093	solo	solo	solo
462	242.9271655328798	solo	solo	solo
463	243.3915646258504	solo	solo	solo
464	243.8559637188209	solo	solo	solo
465	244.3668027210885	solo	solo	solo
466	244.8776417233560	solo	verse	solo
467	245.3884807256236	solo	solo	solo
468	245.8528798185941	solo	chorus	solo

Table 4.8: First and last 15 beats of SALAMI 1200 true, predicted and filtered labels produced by best LSTM model.

Chapter 5

Discussion

The focus of this chapter is to interpret the results reported in the previous chapter. After this interpretation a discussion of the results is held. The different architectures are compared to each other as well will the SbA approach be compared to the DSA approach to MSA. These comparisons aim to provide an answer to the research questions.

5.1 Results Evaluation

Beat Classification Accuracy

Looking at the absolute performance of the best CNN and LSTM model on classifying a beat, one may conclude that the convolutional model performs better than the long short-term memory model. However, looking at the accuracy of both models on the training and test data, the CNN model drops relatively more in accuracy (15.4% for the CNN, 6.8% for the LSTM). This may be an indication of the CNN model being overfit on the training data, therefore reducing its capabilities of classifying beats that the model has not 'seen' before. The possible overfit is unlikely to be a result of too little data since the LSTM did not seem to overfit on the data. The relatively small drop in accuracy for the LSTM model going from the training data to the test data may also indicate that the LSTM managed to capture the overall patterns that define the label of a certain beat, although less accurate for this particular set of beats.

5.1.1 Segment Boundary Detection Accuracy

When looking at the results of the JAMS files with the filtered labels evaluated by the MSAF, we need to take them with a grain of salt. There are a few reasons for this.

Own ground truth

Although the models have been trained and evaluated on the ground truth I created, MSAF only evaluates the accuracy of locating the segment boundary locations of each model. The accuracy is therefore not only subject to the pure performance of a model to classify a beat, but also to the performance of the filtering function. Since I primarily focused on the first, the latter received way less to none effort. As described in chapter 4, the filtering function was based on the predictions made on SALAMI 1200, and the function was therefore optimized to generate filtered labels, and therefore segment boundaries that would make sense, for this specific song. We see that this indeed is the case, since the trimmed and untrimmed F-measures, with 3 seconds tolerance, on SALAMI 1200 (own ground truth) far exceeds the trimmed and untrimmed 3 second F-measure on the full dataset (own ground truth).

It is, however, interesting that the trimmed and untrimmed 0.5 second F-measure on SALAMI 1200 (own ground truth) is lower than the same F-measure on the full dataset (own ground truth) for both the CNN and LSTM model. We see that both the trimmed and untrimmed F-measures severely drop going from a 3 second tolerance to a 0.5 second tolerance (25% to 45%). This applies to both the CNN and LSTM model. Considering that this drop in F-measure is less present on the full dataset, this may indicate that the filtering function is not that bad after all, or that the other songs generally have stronger segment boundaries therefore enabling the models to detect the exact position more precisely. Further research is needed to show this.

SALAMI ground truth

I have also evaluated the JAMS files produced by the models on the official SALAMI ground truth. When looking at the F-measures of the models on SALAMI 1200, we see that the CNN model outperforms the LSTM model. However, if we look at the F-measures of the CNN model and LSTM model on the full dataset (SALAMI ground truth), we see that the differences are gone. If any, they are in the advantage of the LSTM.

5.1.2 Own ground truth vs SALAMI ground truth

As we can see, all F-measures on both SALAMI 1200 and the full dataset drop massively when taking the (official) SALAMI ground truth. This has a few reasons.

Unique Label Count

The most important reason is the amount of unique labels occurring in the SALAMI ground truth and in my own ground truth. As discussed in the section 3.1 and showed in Table 3.1, I've reduced that amount of unique labels from 26 to 9. I

have done this because of the low amount of occurrences of some of the original labels in the dataset (10 labels had around or less that 10 occurrences in the original segments).

If all 26 unique labels were kept, it may have occurred that some labels were only present in the test data, therefore removing the possibility of the models to learn the patterns corresponding to those labels. It also would've almost tripled the amount of neurons in the output layer of both models. Generally, adding more outputs to a model lowers its accuracy, especially when each output has very similar input (*transition* and *interlude*, *pre-verse* and *verse*). In addition, it would've also almost tripled the size of the ground truth output matrix and increased the amount of weights, greatly slowing down the training and evaluation speed of the models.

As a result of this decision, way less segment boundaries occur in my own ground truth than that they occur in the SALAMI ground truth, thus obviously lowering the accuracy of the model. This is supported by the Normalized Entropy Scores Precision (*So*) values of the CNN and LSTM being 0.479 and 0.474 respectively, whilst the Normalized Entropy Scores Recall (*Su*) values are 0.602 and 0.614 respectively [54][1]. As a side effect, more labels also meant that some segments were less than 4 or 5 beats, and would therefore not be detected by my filtering function. It is thus of relatively high priority to, in follow-up research, come up with a better filtering function.

Annotator Inconsistency

The amount of unique labels, and the low occurrence of some of them, may be due to the different annotators that have created the SALAMI ground truth. As mentioned in the introduction, the experience of music differs per person and is i.a. subject to emotion. This means that some annotator may classify one segment as *instrumental*, while another annotator classifies the same segment as *solo*, thereby increasing the amount of unique labels, without there being any musical difference. A consequence of this is that some annotator, judging *instrumental* and *solo* as being different segment, may place more segment boundaries in a piece of music compared to an annotator who judges *instrumental* and *solo* as equal.

Recurring Segments

Another reason that my models may under-segment compared to the SALAMI ground truth, is that a human annotator would place a boundary between two repeated choruses. My models would classify most of the beats in these repeated choruses as *chorus*, and the filtering function will therefore see this as one long *chorus* segment.

Also models implementing the Distance-based Approach, without using a repetition-based technique, will face the same problem. This is therefore an im-

[1] If the *So* value is lower than the *Su* value, the model is undersegmenting the data.

portant factor to take into future research; how often a repetition of similarly labeled segments occur in music and which methods for MSA can detect this.

5.2 Model Comparison

As is the main research question of this thesis, I want to research the feasibility of machine learning for the segmentation by annotation approach to music structure analysis.

5.2.1 CNN versus LSTM

To find an answer to this question I have first explored which deep learning architecture, implementing the SbA approach, produces the best results. I have proposed two deep learning architectures, one using convolutional layers, one using long short-term memory units. I have tried multiple hyperparameters to produce, for each architecture, a best model.

As the results showed, the convolutional architecture to produced the best results overall, however the long short-term memory architecture showed some interesting patterns that could mean that with future research, this architecture can become equal to, or even surpass, the CNN architecture in terms of performance. For this thesis however, I will take the convolutional architecture as best implementation of the segmentation by annotation approach to music structure analysis.

5.2.2 SbA versus DSA

To place the feasibility of machine learning within all approaches of music structure analysis, I have done research to the the other general approach to MSA, the distance-based segmentation by annotation approach respectively and to the current state-of-the-art of music structure analysis. Kuiper [43] describes an successful attempt at improving the current best implementation to the DSA approach to MSA, using feature fusion. His best performing model produces an 0.5 second F-measure of 0.327. Compared to Grill and Schlüter [30], who report a 0.5 second F-measure of 0.508 on a similar but slightly different subset of the SALAMI dataset, this is quite a lot lower. Kuiper [43] does however show that the DSA approach has still room for improvement. The smaller and slightly different subset of the SALAMI dataset used can cause the gap to seem bigger than it may be.

Advantages and Disadvantages

One big advantage of the segmentation by annotation approach, and therefore the models I introduced, is that both annotation and segmentation is performed within one model. The DSA approach first detects the segment boundaries with one model, and then requires another model to label the data. In previous work,

a clustering method was used to group similar segments. Each cluster was then given a capital letter based on the order of occurrence of the segments in the song.

This, however, is still behind the SbA approach, since in the SbA approach the actual segment function is assigned to each segment. Future research has to prove whether each capital letter, assigned via a certain method, always correspond to a segment function. Otherwise, a more complex model is required to assign a function to each segment. If a machine learning model is used, one could argue that using one machine learning model to both annotate and segment a song is more efficient and effective.

An advantage of using DSA over SbA is that a DSA approach model has a higher chance to be applicable on multiple music genres. Although further research has to prove this, the DSA approach detects changes or repeating patterns in music, and uses a quite versatile distance metric for clustering. Since these musical properties are not per definition genre specific, a DSA approach model will be more capable of finding the segment boundaries, while a new SbA model must be created for each specific genre. If the segments need to be labeled, using SbA may then be more effective again, since a specific labeling model then needs to be created anyways.

Conclusion

As explained earlier, due to i.a. the different labels and therefore ground truths used, comparing my results to the current reported results is quite difficult. However the results of the convolutional architecture on both the SALAMI 1200 song with the SALAMI ground truth and full dataset with own ground truth, show that a convolutional architecture implementation of the SbA approach has similar performance as the best DSA approach implementation and current state-of-the-art for MSA. Especially seen the early development stage of my proposed architecture.

For me, this indicates that there is enough reason to put more effort into exploring the segmentation by annotation approach to music structure analysis and an implementation of that approach using machine learning in particular. I will discuss these efforts and the advantages as well as the disadvantages of this approach in the next chapter.

Chapter 6

Future Research

From the results I reported and the discussion thereof, I will propose future research that can be performed to firstly improve the architectures I introduced, and secondly advance the current state-of-the-art of music structure analysis, with the insights gained in this research. I will also discuss the consequences of these insights gained and the proposed future research in the context of artificial intelligence.

6.1 Model improvements

As mentioned in chapter 5, the CNN model performed better than the LSTM model. I also mentioned that despite this preliminary difference between the models, both prove to be quite powerful already. In this next section I will propose some future research that can be performed to improve both architectures.

6.1.1 Hyperparameters and Layout

One of the most important aspects of a neural network are its hyperparameters and layout. Although there are many possible values for the hyperparameters I listed, near infinite amount of possible layouts are possible; more or less hidden layers can be added, the amount of neurons per layer can be differed or a bias can be added to each layer.

For this thesis I have tried to create an as simple layout for each architecture, without letting the layout in itself be the bottleneck for each architecture. To achieve this I have tried to create a layout per architecture that kind of resembles a funnel structure. I then let the values of each hyperparameter define the actual performance of each architecture. One small sign that may indicate that the layout was of decent size is the better performance of models with less neurons in the first layer.
Although more time is always beneficial when trying to find an optimal combination of hyperparameter values I, in my opinion, think that a sufficient amount of values were tested. This research has given, among two working models, at least a

glimpse of the true performance of an optimal layout for each architecture as well as enough reason to find such a layout.

As future research I propose more sophisticated modifications to both proposed architectures, like adding biases, trying more activation functions or testing different optimizers. It may also be beneficial to create more relevant layouts for each architecture; this may include adding or removing hidden layers, bagging models, trying more neuron counts, or even changing some neurons into other types of neurons.

6.1.2 Input Features

Another interesting part of future research will be to find better input features, or, at least, modify the existing features. I suggest this future research based on the final input used by both final models: the CQT and Tempogram. Considering the fact that these features were the top two features regarding their feature vector length, may indicate that both models are benefitted more by more data describing a beat than for example compact features that may express more information per scalar.

One way to test this could be by using a Mel Spectrogram as input (as done in for example [29, 70]). Not only extending the feature vector length can improve the performance, also extending the time context can improve results. Grill and Schluter [29] show that using a longer time context for their self-similarity lag matrix improves performance. Using features with shorted feature vector lengths may then again be used in combination with a longer time context. This will especially be of great benefit for a convolutional as well as a long short-term memory network because of their great performance on big images and longer sequence context respectively. A combination of a long and short time context features combined can also improve results [30].

Another addition to the time context of the features, could be to include the location of the beat that has to be classified in the song. This has to be some measure in proportion to the total length of a song, since in some songs the, for example, intro may be very long (*Xanadu* by Rush). This also applies in general to relatively long songs (*2112* by Rush). It, however, has to be seen how much one scalar adds to a quite big input size, but it may function as a bias for the input.

6.1.3 Architecture Specific Research

Although the long short-term memory architecture performed worse than the convolutional architecture, I expect that with more effort more aimed at LSTM specific properties may greatly increase its performance. The input features may be used in a similar way for both the CNN and LSTM models, they both still work in a very different way. It may therefore be of more benefit for a LSTM architecture to use only the features of the previous and current beat while trying to classify the current beat (similar to how a LSTM is used in word prediction), or to use smaller features while increasing the time context.

In a similar way the convolutional architecture may make leaps in performance if more specific CNN properties are considered. This may include the kernel or pooling size of the (hidden) convolutional and pooling layers respectively. This also ties in with the size of the features size that will partly determine these sizes.

It will also be interesting to test whether using features with a similar length will have any impact. Using features with equal feature vector length will enable the CNN to make use its 3-dimensional layers, originally meant for, for example, RGB- or CMYK-images.

6.1.4 Changing the Output

One big difference of the proposed architectures are the amount of outputs. This is in line with the SbA approach to MSA they implement, however it makes for difficult comparison to other approaches to MSA as does it limit the applicability of these architectures on other music genres with more, less or other segment functions. Although the latter was never the aim of the architectures when creating them, having the possibility of applying these architectures onto other music genres without having to modify them is quite useful. One way to solve this is to create a specific, modified, version of my proposed architectures for each music genre.

The problem of creating a model for multiple music genres at once possibly is a reason why previous work focused on only finding the segment boundaries in piece of music [31]. As discussed earlier, once the segment boundaries are found, another model can then be trained, for each music genre specifically or for all music genres in general, to classify each segment. The architectures I introduced and the insights gained from creating them, can then be used to either improve the boundary detection models, or to create a classification model for many music genres, or the Western Popular Music genre in particular.

It would've been interesting to extend the amount of outputs for the architectures I propose to the amount of unique labels occurring in the original data, however I have explained the reason for having a lower output in subsection 5.1.2. I will propose solutions to the underlying reasons that cause this 'problem' in section 6.2.1.

6.1.5 Real-time MSA

One further interesting application of segmentation by annotation models is real time. In the context of this thesis this means that the architectures will run whilst music is playing, and show the predicted segment function of the segment that is currently being played. This may be interesting for, for example, radio DJs to mix songs at the right moment. To accomplish this, the architectures must be made completely independent of the next beats. This means that a new filtering function needs to be created (something I already recommend as relatively high

priority follow-up research), and all features need to be constructed from past beats.

6.2 Future Research MSA

Although this thesis focused primarily on the segmentation by annotation approach to music structure analysis, the insights gained may be used beyond this approach. Insights about the inputs, the models and the outputs are obtained. In the next subsections I will explain these insights and discuss their consequences for music structure analysis in general.

6.2.1 Data Improvements

One of the most important parts of any model driven research is the data. Although in some cases lots of good quality data is provided, one often has to do with data they can find publicly available. Especially within the field of music analysis, this is a quite common problem. This is primarily due to copyright, which prevents someone of listing lots of songs for free. To not enter any grey areas regarding copyright, I explicitly used copyright free, annotated data.

Audio Quality

One major problem of these copyright free audio files were that these primarily consisted of recordings of live concerts. This meant that the recording quality was not that great overall; audio was not normalized over one song (let alone all songs), crowd sounds were present throughout the duration of the songs, etc.

It has to be seen if this kind of data actually improves models by giving it a bigger challenge, or worsens a model by requiring these models to also make a distinction between actual music and crowd sounds.

One way of solving this problem is already in development at for example Spotify, a music streaming service. They provide an API which can be used to obtain processed audio data. This audio data can not be used to listen to music, but does still contain the auditive features of the song. Creating annotations based on this data means that all music from Spotify may be used, which are millions of songs, together with more metadata that is provided by Spotify (such as the artist, genre or time signature).

Label Quality

This thesis and Kuiper [43] shows the inconsistency between multiple annotations made by different annotators on the same piece of music, and therefore the inherent subjectivity of music structure annotation. This means that if only the annotations of one annotator are taken as ground truth, the models will be overfitted or tuned to the annotation level of that specific annotator.

Using the annotations of all annotators while creating or adjusting a model may therefore benefit overall accuracy, but may also decrease the accuracy of a model on the annotations of one annotator. This means that a new evaluation method needs to be created to evaluate a model that sort of acts as a new annotator, who may be on an annotation level between the other annotators.

One way to solve this would be to use the hierarchical structure of music to our advantage. Grill and Schlüter [30] show that using multiple levels of annotations, and therefore using multiple outputs per beat or time step, increased the accuracy of one of these outputs. This further shows that music is inherently hierarchical, and therefore using another level of this hierarchical property will improve the results of models working on another hierarchical level.

Instead of producing the labels for one specific annotation level, a hierarchical tree can be constructed, each branch denoting the boundary of a segment on a certain hierarchical level. This will further complicate model evaluation, since each branch does not have to be on the same hierarchical level [77, 2].

Another approach to this problem would be to remove the inconsistency between annotators by jointly determining the annotation of a piece of music, this may be one or multiple annotation levels. Once a true ground truth is established, evaluating new or already existing models will be a lot easier as well will be comparing the performance of multiple models. Although this method has a lot of benefits, it has also quite some downsides. Firstly, it will be quite challenging to jointly create such a consistent labeling. Secondly, subjectivity is a very important part of music experience and therefore removing this part will not always be of benefit in the long term.

6.2.2 Interpretable, Comprehensible and Opaque

The comparison between the SbA and DSA approach to music structure analysis can not only be seen as a comparison between two approaches to music structure analysis, but also the comparison between a machine learning and more symbolic approach to music structure analysis, especially when taking the current state-of-the-art in consideration. From this comparison more differences, advantages and disadvantages of each approach can be derived.

An important advantage of a more symbolic approach is its interpretability. This means that each step in the process of music structure detection can be understood, explained and reproduced. In contrast to a convolutional neural network which is at best comprehensible. Comprehensible means that the results of each step can understood, but an underlying technique can not be derived. A way of doing this could be to look at the output of each convolutional layer to see which patterns are extracted in each layer. Reproducing the output, without using the weights learned by the network, will be very difficult.

A LSTM model is even worse in this aspect by being opaque. This implies that one has no understanding of the model except for the learned weights and its inputs and outputs. Comprehending what the role of neuron is, is near impossible, creating a symbolic method that imitates the behavior even less possible.

A consequence of not being able to explain the underlying method of a machine learning model can limit the amount of insights gained from a model that is capable of for example determining the hierarchical structure of all annotations of a piece of music. A more explainable model could therefore be more preferred, even if that means lower absolute accuracy. This is not only a problem within music structure analysis or music information retrieval but in the general field of Artificial Intelligence [19], and is therefore an important factor to take into account when brute forcing machine learning on certain problems within this field.

More hybrid models, combining both machine learning and interpretable symbolic aspects, can therefore be the key to good MSA models, explainable AI and true world apperception of intelligent systems.

Bibliography

[1] Abien Fred Agarap. “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375* (2018).

[2] Abdulqader Almars et al. “Evaluation Methods of Hierarchical Models: 14th International Conference, ADMA 2018, Nanjing, China, November 16–18, 2018, Proceedings”. In: Nov. 2018, pp. 455–464. ISBN: 978-3-030-05089-4. DOI: 10.1007/978-3-030-05090-0_39.

[3] Roberto Basili, Alfredo Serafini, and Armando Stellato. “Classification of musical genre: a machine learning approach.” In: *Proceedings of 5th International Society for Music Information Retrieval Conference, ISMIR*. 2004.

[4] Juan Pablo Bello et al. “A tutorial on onset detection in music signals”. In: *IEEE Transactions on speech and audio processing* 13.5 (2005), pp. 1035–1047.

[5] Bruce Benward and Marilyn Nadine Saker. *Music in theory and practice*. Vol. 7. McGraw-Hill, 1997.

[6] Benjamin Blankertz. “The constant Q transform”. In: (2001). URL: http://doc.ml.tu-berlin.de/bbci/material/publications/Bla_constQ.pdf.

[7] Sebastian Böck et al. “Online real-time onset detection with recurrent neural networks”. In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. 2012.

[8] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[9] Judith Brown. “Calculation of a constant Q spectral transform”. In: *The Journal of the Acoustical Society of America* 89.1 (1991), pp. 425–434.

[10] Judith Brown and Miller Puckette. “An efficient algorithm for the calculation of a constant Q transform”. In: *The Journal of the Acoustical Society of America* 92.5 (1992), pp. 2698–2701.

[11] Peter Brown et al. “Class-based n-gram models of natural language”. In: *Computational linguistics* 18.4 (1992), pp. 467–479.

[12] Taras Butko and Climent Nadeu. "Audio segmentation of broadcast news in the Albayzin-2010 evaluation: overview, results, and discussion". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2011.1 (2011), pp. 1–10.

[13] Wei Chai. "Automated analysis of musical structure". PhD thesis. Massachusetts Institute of Technology, 2005.

[14] François Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[15] GW Cooper and LB Meyer. *The rhythmic Structure of Music. Chicago: Univ*. of Chicago press, 1960.

[16] John Covach. "Form in rock music". In: *Engaging music: Essays in music analysis* (2005), pp. 65–76.

[17] Arnie Cox. "Tripartite Subjectivity in Music Listening". In: *Indiana Theory Review* 30.1 (2012), pp. 1–43. ISSN: 02718022. URL: `http://www.jstor.org/stable/24045414`.

[18] Ebru Dogan, Mustafa Sert, and Adnan Yazici. "Content-based classification and segmentation of mixed-type audio by using MPEG-7 features". In: *2009 First International Conference on Advances in Multimedia*. IEEE. 2009, pp. 152–157.

[19] Derek Doran, Sarah Schulz, and Tarek Besold. "What does explainable AI really mean? A new conceptualization of perspectives". In: *arXiv preprint arXiv:1710.00794* (2017).

[20] Samuel Dupond. "A thorough review on the current advance of neural network structures". In: *Annual Reviews in Control* 14 (2019), pp. 200–230.

[21] Daniel Ellis. "Beat tracking by dynamic programming". In: *Journal of New Music Research* 36.1 (2007), pp. 51–60.

[22] Jeffrey Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[23] Florian Eyben et al. "Universal onset detection with bidirectional long-short term memory neural networks". In: *Proceedings of 11th International Society for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*. 2010, pp. 589–594.

[24] Jonathan Foote. "Automatic audio segmentation using a measure of audio novelty". In: *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*. Vol. 1. IEEE. 2000, pp. 452–455.

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

[26] Stanley Gelfand. *Essentials of audiology*. s 562. Thieme New York, 1997, p. 87.

[27] Sahar Ghannay et al. "Word embedding evaluation and combination". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016, pp. 300–305.

[28] Pablo Gimeno et al. "Multiclass audio segmentation based on recurrent neural networks for broadcast domain data". In: *EURASIP Journal on Audio, Speech and Music Processing* (2020). URL: https://doi.org/10.1186/s13636-020-00172-6.

[29] Thomas Grill and Jan Schluter. "Music boundary detection using neural networks on spectrograms and self-similarity lag matrices". In: *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE. 2015, pp. 1296–1300.

[30] Thomas Grill and Jan Schlüter. "Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations." In: *Proceedings of 16th International Society for Music Information Retrieval Conference, ISMIR*. 2015, pp. 531–537.

[31] Thomas Grill and Jan Schlüter. "Structural segmentation with convolutional neural networks MIREX submission". In: *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)* (2015), p. 3.

[32] Peter Grosche, Meinard Müller, and Frank Kurth. "Cyclic tempogram—a mid-level tempo representation for musicsignals". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, pp. 5522–5525.

[33] Christopher Harte, Mark Sandler, and Martin Gasser. "Detecting harmonic change in musical audio". In: *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. 2006, pp. 21–26.

[34] Geoffrey Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[35] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[36] Yen-Chang Hsu et al. "Multi-class classification without multi-class labels". In: *arXiv preprint arXiv:1901.00544* (2019).

[37] David Hubel and Torsten Wiesel. "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[38] Eric Humphrey et al. "JAMS: A JSON Annotated Music Specification for Reproducible MIR Research." In: *Proceedings of 15th International Society for Music Information Retrieval Conference, ISMIR*. 2014, pp. 591–596.

[39] Michael Jordan. “Serial order: A parallel distributed processing approach”. In: *Advances in psychology*. Vol. 121. Elsevier, 1997, pp. 471–495.

[40] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[41] Marko Kos et al. “On-line speech/music segmentation for broadcast news domain”. In: *2009 16th International Conference on Systems, Signals and Image Processing*. IEEE. 2009, pp. 1–4.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[43] Jesper Kuiper. *Music Structure Analysis: An Exploration of and Improvements on the Distance-based Segmentation and Annotation Approach*. Bachelor’s Thesis, University Utrecht. June 2020.

[44] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[45] Kyogu Lee. “Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile.” In: *ICMC*. 2006.

[46] Fred Lerdahl and Ray Jackendoff. “An overview of hierarchical structure in music”. In: *Music Perception: An Interdisciplinary Journal* 1.2 (1983), pp. 229–252.

[47] Tom Li, Antoni Chan, and Andy Chun. “Automatic musical pattern feature extraction using convolutional neural network”. In: *Genre* 10 (2010), p. 1.

[48] Xiangang Li and Xihong Wu. “Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 4520–4524.

[49] Grace Lindsay. “Convolutional neural networks as a model of the visual system: past, present, and future”. In: *Journal of Cognitive Neuroscience* (2020), pp. 1–15.

[50] Hung-Yi Lo, Ju-Chiang Wang, and Hsin-Min Wang. “Homogeneous segmentation and classifier ensemble for audio tag annotation and retrieval”. In: *2010 IEEE International Conference on Multimedia and Expo*. IEEE. 2010, pp. 304–309.

[51] Beth Logan et al. “Mel frequency cepstral coefficients for music modeling.” In: *Proceedings of International Society for Music Information Retrieval, ISMIR*. Vol. 270. 2000, pp. 1–11.

[52] Marc Moreno Lopez and Jugal Kalita. “Deep Learning applied to NLP”. In: *arXiv preprint arXiv:1703.03091* (2017).

[53] Lie Lu, Stan Li, and Hong-Jiang Zhang. “Content-based audio segmentation using support vector machines”. In: *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.* IEEE. 2001, pp. 749–752.

[54] Hanna Lukashevich. "Towards Quantitative Measures of Evaluating Song Segmentation." In: *Proceedings of 9th International Society for Music Information Retrieval Conference, ISMIR*. 2008, pp. 375–380.

[55] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. `https://www.tensorflow.org`. 2015.

[56] Brian McFee et al. "LibROSA: Audio and music signal analysis in Python". In: *Proceedings of the 14th Python in science conference*. Vol. 8. 2015.

[57] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[58] Ananya Misra. "Speech/nonspeech segmentation in web videos". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[59] Meinard Müller. *Fundamentals of Music Processing*. Erlangen, Germany: Springer International Publishing Switzerland, 2015.

[60] Meinard Müller, Frank Kurth, and Michael Clausen. "Chroma-based statistical audio features for audio matching". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE. 2005, pp. 275–278.

[61] Oriol Nieto and Juan Pablo Bello. "Music segment similarity using 2d-fourier magnitude coefficients". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 664–668.

[62] Oriol Nieto and Juan Pablo Bello. "Systematic exploration of computational music structure research." In: *Proceedings of 17th International Society for Music Information Retrieval Conference, ISMIR*. 2016, pp. 547–553.

[63] Francois Pachet and Jean-Julien Aucouturier. "Improving timbre similarity: How high is the sky". In: *Journal of negative results in speech and audio sciences* 1.1 (2004), pp. 1–13.

[64] Yorgos Patsis and Werner Verhelst. "A speech/music/silence/garbage/classifier for searching and indexing broadcast news material". In: *2008 19th International Workshop on Database and Expert Systems Applications*. IEEE. 2008, pp. 585–589.

[65] Jouni Paulus and Anssi Klapuri. "Acoustic features for music piece structure analysis". In: *Proc. of 11th International Conference on Digital Audio Effects*. 2008, pp. 309–312.

[66] Gaël Richard, Mathieu Ramona, and Slim Essid. "Combined supervised and unsupervised approaches for automatic segmentation of radiophonic audio streams". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. Vol. 2. IEEE. 2007, pp. II–461.

[67] AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, 1987.

[68] Hasim Sak, Andrew Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: (2014).

[69] Jan Schlüter and Sebastian Böck. "Improved musical onset detection with convolutional neural networks". In: *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2014, pp. 6979–6983.

[70] Jan Schlüter and Sebastian Böck. "Musical onset detection with convolutional neural networks". In: *6th international workshop on machine learning and music (MML), Prague, Czech Republic*. 2013.

[71] Christian Schörkhuber and Anssi Klapuri. "Constant-Q transform toolbox for music processing". In: *7th Sound and Music Computing Conference, Barcelona, Spain*. 2010, pp. 3–64.

[72] Mike Schuster and Kuldip Paliwal. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[73] Joan Serra et al. "Unsupervised detection of music boundaries by time series structure features". In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[74] Joan Serrà et al. "Measuring the evolution of contemporary western popular music". In: *Scientific reports* 2 (2012), p. 521.

[75] Jordan Smith et al. "Design and creation of a large-scale database of structural annotations". In: *International Society of Music Information Retrieval, ISMIR* (2011), pp. 555–560.

[76] Stanley Smith Stevens, John Volkmann, and Edwin Newman. "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[77] Aixin Sun and Ee-Peng Lim. "Hierarchical text classification and evaluation". In: *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE. 2001, pp. 521–528.

[78] Theodoros Theodorou, Iosif Mporas, and Nikos Fakotakis. "An overview of automatic audio segmentation". In: *International Journal of Information Technology and Computer Science (IJITCS)* 6.11 (2014), p. 1.

[79] Karen Ullrich, Jan Schlüter, and Thomas Grill. "Boundary Detection in Music Structure Analysis using Convolutional Neural Networks." In: *Proceedings of 15th International Society for Music Information Retrieval Conference, ISMIR*. 2014, pp. 417–422.

[80] Paul Werbos. "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4 (1988), pp. 339–356.

[81] Saadia Zahid et al. “Optimized Audio Classification and Segmentation Algorithm by Using Ensemble Methods”. In: *Mathematical Problems in Engineering* (2015). URL: https://doi.org/10.1155/2015/209814.

[82] Junfang Zhang et al. “Audio segmentation system for sport games”. In: *2010 International Conference on Electrical and Control Engineering*. IEEE. 2010, pp. 505–508.

[83] Ruohua Zhou and Josh Reiss. “Music Onset Detection”. In: *Machine Audition: Principles, Algorithms and Systems*. IGI Global, 2011, pp. 297–316.

Appendices

Appendix A

Definitions

Definition 1.1.1 (Musical Structure)**.** The *Musical Structure* of a piece of music is a series of *segments* (1.1.2) that describes the high-level structure of a piece of music.

Definition 1.1.2 (Segment)**.** A *Segment* is one consecutive piece of audio in a piece of music that has one *segment function* (1.1.3), with a start- and an end-boundary.

Definition 1.1.3 (Segment Function)**.** Following the definitions as they can be found in Benward and Saker [5], Part B: *The Structural Elements of Music*, a *Segment Function* is the function of a *segment*, this is either:

- Chorus
- Verse
- Bridge
- Interlude / Transition
- Intro
- Outro
- Solo
- (Silence)
- (Background Noise)

Definition 1.3.1 (Segmentation by Annotation)**. Segmentation by Annotation** in the context of music structure analysis means first dividing a song into many small pieces (e.g. pieces of 10ms or every beat). Thereafter a *segment function* is assigned to each small piece using some kind of classification method (e.g. statistics, support vector machine, deep learning). Sequences of similarly annotated small pieces are then combined into *segments*, resulting in the musical structure of the song.

Definition 1.3.2 (Distance-based Segmentation and Annotation)**. Distance-based Segmentation and Annotation** in the context of music structure analysis means first finding segment boundaries using some kind of distance metric. Subsequently

these segments are grouped into groups that are similar to each other in their structural role in the piece of music, using some kind of grouping method (e.g. nearest neighbors, support vector machines), resulting in each segment getting a capital letter denoting their function. Then, each capital letter is converted into a segment function, this can be either done using some kind of statistics, pattern matching or machine learning method.

Appendix B

Research Questions

Research Question 1 (Main RQ)**.** What is the feasibility of a machine learning implementation of the Segmentation by Annotation approach to Music Structure Analysis in Western Popular Music?

Research Question 2 (Sub-RQ 1)**.** Which deep learning architecture, implementing the Segmentation by Annotation approach to Music Structure Analysis, yields the best relative results?

Research Question 3 (Sub-RQ 2)**.** How does the performance of the best performing deep learning architecture, implementing the Segmentation by Annotation approach to Music Structure Analysis, compare to the performance of implementations of the Distance-based Segmentation and Annotation approach, and a state-of-the-art implementation in particular?

Appendix C

Dataset

Below is a listing of all songs used in these thesis. The songs were extracted from the *Internet Archives* subset of the SALAMI database [75]. The IDs correspond to the *SONG_ID* indexer of the SALAMI database. These are 277 tracks in total.

```
956,  958,  960,  962,  964,  968,  970,  972,  974,  976,
978,  980,  982,  984,  986,  988,  990,  992,  994,  996,
998,  1000, 1004, 1006, 1008, 1012, 1014, 1016, 1018, 1020,
1022, 1024, 1026, 1028, 1032, 1034, 1036, 1038, 1042, 1044,
1046, 1048, 1050, 1054, 1056, 1058, 1060, 1062, 1064, 1066,
1068, 1070, 1072, 1074, 1076, 1078, 1080, 1082, 1084, 1086,
1088, 1090, 1092, 1094, 1096, 1098, 1100, 1102, 1104, 1106,
1108, 1110, 1112, 1114, 1116, 1118, 1120, 1122, 1124, 1128,
1130, 1132, 1134, 1136, 1138, 1142, 1144, 1146, 1148, 1150,
1152, 1154, 1156, 1158, 1160, 1162, 1164, 1166, 1168, 1170,
1172, 1174, 1176, 1180, 1182, 1184, 1186, 1188, 1190, 1192,
1194, 1196, 1198, 1200, 1202, 1204, 1206, 1208, 1210, 1212,
1214, 1216, 1218, 1220, 1222, 1224, 1226, 1228, 1230, 1232,
1234, 1236, 1238, 1240, 1242, 1244, 1246, 1248, 1250, 1254,
1256, 1258, 1260, 1262, 1264, 1266, 1268, 1270, 1272, 1274,
1276, 1278, 1280, 1282, 1284, 1286, 1288, 1290, 1292, 1294,
1296, 1298, 1300, 1302, 1304, 1306, 1308, 1310, 1312, 1314,
1316, 1318, 1322, 1324, 1326, 1328, 1330, 1332, 1334, 1336,
1338, 1340, 1342, 1346, 1348, 1350, 1352, 1354, 1356, 1358,
1360, 1362, 1364, 1366, 1368, 1370, 1372, 1374, 1376, 1378,
1380, 1382, 1384, 1386, 1387, 1388, 1390, 1391, 1392, 1394,
1395, 1396, 1399, 1400, 1402, 1403, 1404, 1406, 1407, 1408,
1412, 1414, 1415, 1416, 1418, 1419, 1420, 1422, 1423, 1424,
1427, 1428, 1431, 1432, 1434, 1435, 1436, 1438, 1439, 1442,
1443, 1444, 1446, 1447, 1448, 1450, 1451, 1452, 1454, 1455,
1456, 1458, 1459, 1460, 1462, 1464, 1467, 1468, 1470, 1472,
1474, 1475, 1476, 1478, 1479, 1482, 1483, 1484, 1487, 1488,
1490, 1491, 1492, 1494, 1495, 1496, 1498
```