Music Structure Boundaries Estimation Using Multiple Self-Similarity Matrices as Input Depth of Convolutional Neural Networks

Alice Cohen-Hadria, Geoffroy Peeters

Audio Engineering Society

Conference Paper

Presented at the Conference on
Semantic Audio
2017 June 22 – 24, Erlangen, Germany

Music Structure Boundaries Estimation Using Multiple Self-Similarity Matrices as Input Depth of Convolutional Neural Networks

Alice Cohen-Hadria[1] and Geoffroy Peeters[1]

[1] *UMR STMS 9912 (IRCAM - CNRS - UPMC), 1 place Igor Stravinsky, 75004 Paris, France*

Correspondence should be addressed to Alice Cohen-Hadria (alice.cohenhadria@ircam.fr)

ABSTRACT

In this paper, we propose a new representation as input of a Convolutional Neural Network with the goal of estimating music structure boundaries. For this task, previous works used a network performing the late-fusion of a Mel-scaled log-magnitude spectrogram and a self-similarity-lag-matrix. We propose here to use the square-sub-matrices centered on the main diagonals of several self-similarity-matrices, each one representing a different audio descriptors. We propose to combine them using the depth of the input layer. We show that this representation improves the results over the use of the self-similarity-lag-matrix. We also show that using the depth of the input layer provide a convenient way for early fusion of audio representations.

1 Introduction

Music structure discovery (MSD) is a recent research field, which aims at estimating automatically the temporal structure of a music track by analyzing the characteristics of its audio signal over time. Such structure can be used for interactive browsing within a track [1], automatic summary generation [2], automatic DJ [3] or computational musicology.

In MSD, the temporal structure of a music track is represented as a succession of segments. Such segments can correspond to - homogeneous audio content (also named “states" [4]), - repeated audio content (also named “sequences" when they are non-homogeneous [4]) or - non-homogeneous non-repeated content (in this case they are only defined as the temporal segment between two novelty boundaries). In pop music, such segments can correspond to the verse, chorus or bridge parts of a song. In a MSD representation, each segment is characterized by its temporal boundaries and a label indicating its similarity with the other segments. In the present work, we only consider the problem of estimating the boundaries.

Defining what is music structure is still a research topic. Several proposals have been made so far to define it (see for example the works of [5], [6] and [7]), each leading to a different music structure annotation system. In this paper, we will rely on the definition underlying the annotations made for the SALAMI dataset.

1.1 Related works

Research related to the automatic estimation of music structure started in 1999 with the work of Foote

[8]. Until the accessibility of large annotated datasets, the methods used to estimate the music structure were mostly based on **unsupervised** learning algorithms: — clustering [9] or hidden Markov model [2] applied to audio signal features, — dynamic time warping [10], non-negative matrix factorization [11] or singular value decomposition [12] applied to a self-similarity matrix.

Recently, large datasets of music annotated in structure have appeared (such as RWC [13], INRIA [14], SALAMI [7]) allowing the use of **supervised** learning algorithms. The first supervised training approach [15] was an adaptation of Fisher's linear discriminant analysis used to improve the results of a previous clustering.

Convolutional neural network. Following what happened in other domains (such as text or image recognition), neural networks with many hidden layers, a.k.a. deep learning, have allowed to largely increase the recognition results in various music audio recognition tasks (onset, beat, downbeat or music structure boundaries estimation). Various types of units were proposed to apply a network to an audio signal representation. [16] or [17] first proposed to use Bi-directional Long-Short-Term-Memory (BLSTM) units. [18] or [19] later proposed to use the more tractable Convolutional units [20] with even better recognition results. In this work, we will rely on Convolutional units. In this case, the network is named a Convolutional Neural Network (CNN or ConvNet).

Input representation. The input of such a ConvNet is a representation of the audio signal. In previous works, spectrogram, constant-Q-transform and Mel-scaled Log-magnitude Spectrograms (MLS) have been proposed. While these representations seems adequate to represent the short-term properties of an audio signal (hence short-term variations typical of onsets or beats), the boundaries defining music structure may need a wider observation context. For this reason, [19] have proposed to use as input the self-similarity-lag-matrix [21], i.e. a self-similarity matrix expressed in (lag, time) rather than (time, time).

Our proposal. In this work, we propose a new representation to be used as input to a ConvNet.

1. First, we propose to use as input the succession of square-sub-matrices centered on the main diagonal of a Self-Similarity-Matrix (SSM). In the case of homogeneous segments, this representation provides much sharper edges at the beginning and ending of segments than the lag-matrix. This representation was already used by Foote [22] to estimate music structure boundaries (by convolving it with a single predefined checker-board kernel) or by Kaiser et al. [23] (using several predefined kernels). We believe that the ConvNet can find more appropriate kernels for this task.

2. Second, we propose to use the depth of the input layer of a ConvNet to represent various viewpoints on the audio content. Indeed, when computing a SSM, the choice of the signal representation plays a crucial role. Using Mel-Frequency-Cepstral-Coefficient (MFCC) or chroma as audio signal representation will lead to two different SSMs highlighting possibly two different temporal structures. We therefore propose to combine several SSMs using the depth of the input layer. Such depth is usually used in computer vision to represent the R, G, B colors of an image.

1.2 Organization of the paper

The paper is organized as followed. In part 2, we explain the use of convolutional neural network for music structure boundary estimation, the architecture of the network, the peak-picking algorithm and the training process. In part 3, we present our new input representation and the way we use the input depth to represent it. In part 4, we then evaluate our proposal against previously published results. We finally discuss our results and present directions for future works.

2 Estimating musical boundaries using Convolutional neural networks

Convolutional neural network. A ConvNet is a specific type of feed-forward neural network in which units share their parameters ("parameter sharing") and are locally connected with each other ("local connectivity"). The connection weights can therefore be considered as a filter ("local connectivity") convolved ("parameter sharing") with the input representations. Such filter is often named a receptive field. At each layer, the output values of the previous layers are convolved with a set of such filters and passed through a non-linearity or activation function (such as a hyperbolic tangent denoted by tanh) to create the layer output values. To make the layer invariant by translations within the dimensions

of the input, a pooling operation can be applied after the non-linearity. A popular choice for this is the max-pooling, which only keeps the maximum value over a fixed area. This new representation is then used as input for the following layer.

ConvNet have been originally designed for computer vision. The input is then a two dimensional representation. In this case, the invariant feature detection performed by a layer is in fact similar to some function in the visual cortex. Ullrich et. al. [18] were the first to propose their use for music structure boundary estimation. Their system is the current state-of-the-art in terms of results.

Input representations. While ConvNet can be applied directly to a 1D signal (such as the audio waveform [24]), it is usual to transform the 1D signal in a 2D representation to allow to apply a convolution along the two dimensions. For this 2D representation, Ullrich et al. [18] have proposed the Mel-scaled Log-magnitude Spectrograms (**MLS**). An audio track is then represented as a succession of images taken from the MLS. Each image represent a 16 s duration segment by 80 Mel bands. Each image has an associate label ("boundary" or "non-boundary"), which indicates whether or not a boundary exists at the center frame of the segment. Later, Grill et al. [19] have proposed to add another 2D representation: the Self-Similarity-Lag-Matrix (**SSLM**), i.e. a self-similarity matrix expressed in (lag, time) rather than (time, time).

Network architecture. The network architecture used in [18] and [19] is the following: two convolutional layers followed by two fully connected layers, the last layer being made of a single neuron with a sigmoid activation, giving an output between 0 and 1. This output represents the probability that the center frame of the segment is a boundary. While [18] only apply this architecture to MLS, [19] apply it to both MLS and SSLM. Although the use of the sole SSLM does not improve results, its combination with the MLS proves to be the best. To combine the MLS and SSLM representations, two models are compared: 1) the average of the outputs of two independently trained ConvNets, 2) merging the two networks in a new convolutional layer. This second model performs the best and therefore will be the one used in the following.

2.1 Training

As for fully-connected neural networks, ConvNet are trained using the back-propagation algorithm. In this, the parameters of the network (the weights or filters of each layer) are iteratively updated based on the gradient of the prediction error w.r.t. these parameters.

Loss function. The goal of the training is to minimize the prediction error also named loss function. In [18] and [19], this loss-function is defined by the binary-cross entropy between the ground truth we are trying to predict (y) and the prediction of it based on the output of the network ($\hat{y}$):

$$BinaryCrossEntropy(y,\hat{y}) = y\ln(\hat{y}) + (1-y)\ln(1-\hat{y})$$

For a binary classification, the ground truth y has to be in $\{0,1\}$. To match this binary classification, we will pass through the network an MLS or an SSLM excerpt, and considered that a segment is labeled 1 if the middle frame is a ground truth boundary, 0 if not.

Back propagation algorithm. For a given input, we compute successively the activations of all layers until the output $\hat{y}$. We then compute the loss function given this prediction $\hat{y}$ and the ground-truth output y. This is the forward pass. We then compute the gradient of the loss w.r.t. all the parameters of the network. The goal is to modify these parameters in the opposite direction of these gradients in order to decrease the loss to a good local minimum (hopefully to a global minimum). To compute these gradients, we use the chain rule: for a given layer, the gradient of the loss function is expressed recursively in function of the gradient of the layer above. This is the backward pass: we propagate the gradient of the loss backward in the network.

Many different algorithms have been proposed to update the gradient each allowing better convergence to a local minimum of the loss function. The *momentum* algorithm [25], used in [18] and [19], keeps track of the previous updates in order to optimize the convergence of highly non-convex loss functions. The more recent algorithm *AdaMax* [26], that we will use here, is based on the estimation of the first moment and the infinity norm of the past and current gradients.

In the simplest form (Batch Gradient Descent), every input of the training set is passed trough this forward/backward propagation. The average (over all inputs) gradients are computed and the parameters of the network updated accordingly. In this case, the convergence can be very slow. Passing all inputs of the training set to the network is named an epoch. In the

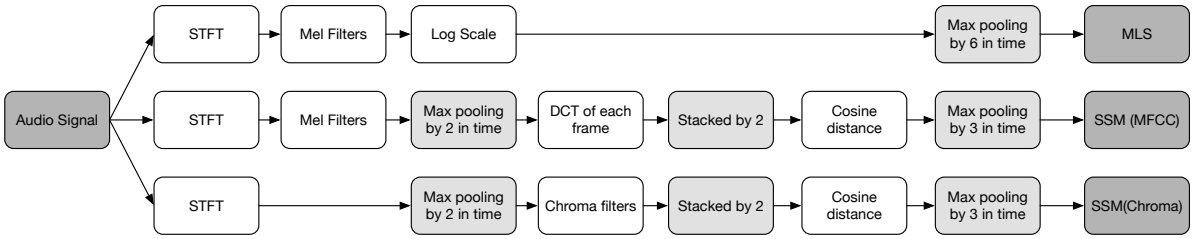

Fig. 1: Computation of the three input representations.

opposite, in Stochastic Gradient Descent (SGD), the update is performed after processing each input. In this case, the estimation of the gradient is very poor. In the middle, in mini-batch gradient descent, updates of the parameters are performed after processing a fixed number (named batch) of inputs. In the following, we will use a mini-batch of 128 inputs.

Stopping criteria (validation set). To stop the iterative update process, several strategies are possible. Apart from defining a fixed number of iterations, the most common is to use a separate dataset (named validation set) and periodically monitor the generalization performance of the network on it. When the error on the validation set stop decreasing, the training is stopped.

To further improve the results, one can use a bagging of several networks. In this, several networks are trained independently using the same input data but different initializations. The final output is then the averaging of the outputs of the different trained networks. In the following all results are presented using a bagging over 5 networks.

2.2 Using the network to estimate music structure boundaries

Given that a music track is represented as a succession of 16 s. segments over time and that each segment passed through the network gives an output value (the sigmoid activation) in $[0,1]$, the whole track can be represented as a temporal curve with values in $[0,1]$ representing the probability of a boundary.

Direct threshold. In this method, we simply apply a threshold on the curve to decide on the boundaries. This threshold has been trained (on the validation set).

Peak-picking. In order to take into account past and future context of the segment within the music track, [18] proposes a peak-picking approach. A peak is defined as a value which is the maximum in the temporal window $\pm$ 6s. A strength is associated to this peak defined as its value minus the average activation in the past 12 seconds and the forward 6 seconds. A threshold is then applied on the remaining peaks. This threshold has been trained (on the validation set).

3 Proposed method

Input representation. In this work, we propose a new representation to be used as input to a ConvNet. We propose to use the succession of square-sub-matrices centered on the main diagonal of a Self-Similarity-Matrix (SSM) as input. More precisely, if we denote by $SSM(i,i')\ \ \forall i,i' \in [0,I-1]$ the SSM representing the similarity between pairs of times i and i', we propose to use the set of square-sub-matrices $subSSM_i$ defined by $subSSM_i(j,j') = S(j+(i-F/2), j'+(i-F/2))$ with $j,j' \in [0,F-1]$, $i \in [F/2, I-F/2]$ and where F is the size of the sub-matrix (see Figure 2 for illustration).

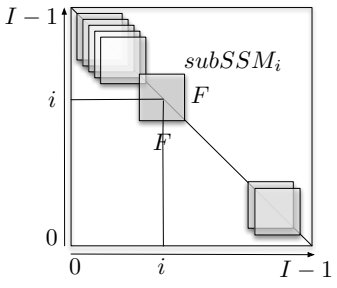

Fig. 2: $subSSM_i$ input representation for the ConvNet.

In the case of homogeneous segments, this representation provides much sharper edges at the beginning and ending of segments than the lag-matrix used by [19]. This representation was already used to estimate music structure boundaries: by Foote [22] (by convolving it

with a single predefined checker-board kernel) or by Kaiser et al. [23] (using several predefined kernels). We believe that the ConvNet can find more appropriate kernels for this task.

Using the depth of the input layer. When computing a SSM, the choice of the audio descriptors used to represent the audio signal content plays a crucial role. Using different audio descriptors will lead to different SSMs highlighting possibly different temporal structure and therefore different boundaries, such as the structure underlying timbre changes or the one underlying harmonic pattern repetitions. Here, we chose to use two classical audio descriptors, the MFCC (for Mel-frequency Cepstral Coefficients), which describes the timbre, and the chroma (also named Pitch-Class-Profile), which describes the harmonic content of the track. We therefore represent an audio track by two different SSMs.

To combine the information provided by the two SSMs late fusion of independent networks can be used (for example by averaging the output of independent networks). However, we believe that performing early fusion will help preserving the temporal relationship between the two representations of our audio signal. To achieve this early fusion, we propose to use the depth of the input layer of the convolutional layer (see Figure 3). In computer vision, this depth is used to represent the R, G, B colors of an image. Here, we combine the two SSMs (computed with the MFCC and the Chroma) to create an input tensor of depth 2.

The computation of the three input representations is detailed in Figure 1 and explained below.

Computation of the MLS. The MLS are computed as in [19]. A Short-Time-Fourier-Transform (STFT) is computed using a Hanning window of 46 ms. duration and 50% overlap. The magnitude spectrogram is then converted to 80 Mel bands and the resulting amplitudes logarithmically-scaled. They are then pooled over time by a factor 6 using a maximum operator. We denote it by $MLS(i, b \in [1, 80])$. The hop size (difference between successive time frames i) is 138 ms.

Computation of $\boldsymbol{subSSM_i^{mfcc}}$. The MLS are computed as above but only pooled over time by a factor 2 using a maximum operator. A DCT is then performed over the resulting MLS (omitting the 0-th element) [27]. The resulting MFCC vectors are stacked by 2 to create a 160-dimensions vector. The SSM is then computed

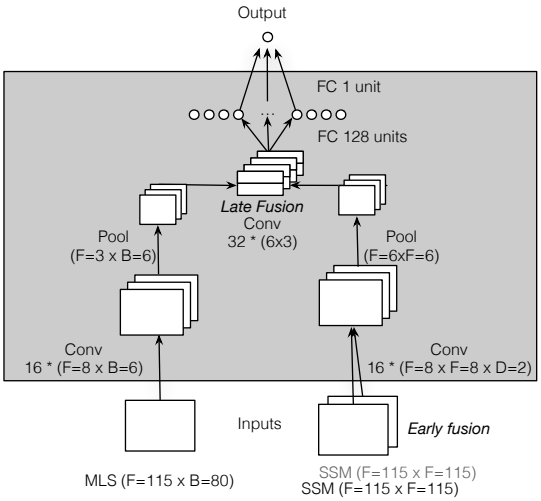

Fig. 3: Architecture for the proposed method for music structure bloundary estimation, using MLS, $subSSM^{mfcc}$ and $subSSM^{chroma}$.

from the MFCC using a cosine distance. To reduce the dimensionality, the SSM is pooled by a 3 factor in both time dimensions using a maximum operator. We denote it by $SSM^{mfcc}(i, i')$. We derive the set of square submatrices $subSSM_i^{mfcc}(j, j')$ from it. The hop size is also 138 ms.

Computation of $\boldsymbol{subSSM_i^{chroma}}$. We first compute a Short-Time-Fourier-Transform (STFT) using a Hanning window of 46 ms. duration with a 50% overlap. The STFT is then pooled over time by a factor 2 using a maximum operator. A set of 12 chroma filters [28] is then applied. The resulting chroma vectors are stacked by 2 to create 24-dimensions vectors. The SSM is then computed from the chroma using a cosine distance. To reduce the dimensionality, the SSM is pooled by a 3 factor in both time dimensions using a maximum operator. We denote it by $SSM^{chroma}(i, i')$. We derive the set of square sub-matrices $subSSM_i^{chroma}(j, j')$ from it. The hop size is also 138 ms.

In the case of SSM, as in [19], we used circular padding to add the necessary context for estimating boundaries at the beginning and ending of the track.

Network Architecture. The architecture of our network is close to one proposed by [19] but adapted to our proposals. It is illustrated in Figure 3. As in [19], the network operates a late-fusion of two sub-networks

(one representing MLS, the other *subSSM*) using a convolutional layer. This convolutional layer is followed by two fully-connected layers: one with 128 tanh hidden units and one with a sigmoid unit that constitutes our boundary probability.

The **left sub-network** operates over segments taken over the MLS of size $F = 115$ frames (representing a duration of 15.87 s.) and $B = 80$ frequency bands (F=115 $\times$ B=80). The convolutional layer contains 16 receptive fields of size (F=8 $\times$ B=6) followed by a tanh activation and a max-pooling of size (F=3 $\times$ B=6).

The **right sub-network** is different than the ones of [19]. It operates over 2-dimensional tensors representing the two square sub-matrices $subSSM_i^{mfcc}$ and $subSSM_i^{chroma}$. Its size is (F=115 $\times$ F=115 $\times$ D=2). The convolutional layer also contains 16 receptive fields but since the $subSSM_i$ are square, the receptive fields are also square. There size is (F=8 $\times$ F=8 $\times$ D=2). They are followed by a tanh activation and a max-pooling of size (F=6 $\times$ B=6).

We used the binary cross entropy as our loss function and we trained the network using the *AdaMax* algorithm ([26]) with mini batch of size 128. To avoid over-fitting, we applied dropout to both fully-connected layers with a 50% rate.

To estimate the final boundaries, we will compare the two approaches mentioned above: direct threshold and peak-picking (as proposed by [18]). To measure the impact of the use of the depth of the input layer, we will also train models using only $subSSM_i^{mfcc}$ and using only $subSSM_i^{chroma}$.

4 Evaluation

4.1 Dataset

To evaluate our proposal and to be able to compare our results to current state-of-the-art algorithms[19], we used the SALAMI dataset [7] for training, validating and testing. SALAMI contains 1048 music tracks of various music genres (including popular, classical, jazz or world music) annotated in music structure at different temporal scales and by two different annotators. As [19] did, we will only used the annotation of annotator 1 at the largest temporal scale.

It should be noted that we didn't have access to all the audio tracks of the dataset but only to 732 of them. Also it should be mentioned that [19] used a second private dataset for training their system. Therefore, in order to be able to compare our system to the one proposed by [19], we re-implemented [19] and ran it on our dataset.

We split our 732 audio tracks into 400 tracks for training, 100 for validation and 232 for testing. The splitting has been to ensure that the same artist does not appear both in the training and testing set. The split into training, validation test, and testing is available at `https://github.com/aliceCohenHadria/SALAMI-split-for-AES-article`.

4.2 Performance measures

As in previous works, we evaluate our results using the **F-measure**.

Since the final step of the boundaries estimation algorithm involves the choice of a threshold (either in the "direct threshold" or in the "peak-picking" method), we also compute a performance measure which is independent of the choice of this threshold: the **Area Under the ROC Curve (AUC)**. For a given choice of a threshold, one can compute the True Positive (TP) rate and the False Positive (FP) rate. This can be done for all possible choices of the threshold. The ROC curve then represents the values of the TP rate as a function of the FP rate. The AUC is then simply the area under this curve. A value of the AUC close to 1 indicates a system, which does not depend on a specific choice of a threshold (a robust system). In our experiments, in all cases ("direct threshold" or in the "peak-picking" method), the AUC is computed using the network output value.

The F-measure and AUC are computed using two tolerance windows: $\pm$0.5 s. and $\pm$3 s.

4.3 Results

Results are presented in Table 1 in terms of F-measure, Precision, Recall and AUC at $\pm$0.5 s. and $\pm$3 s.

In this, we compare the results obtained using the late-fusion (using a convolutional layer) of a sub-network using MLS and a sub-network using ① $subSSM^{mfcc}$, ② $subSSM^{chroma}$ and ③ using the depth of the input layer to combine $subSSM^{mfcc}$ and $subSSM^{chroma}$. In all cases, we use the peak-picking proposed by [18] for the final boundary estimation. (3') is the same as ③ but using the direct threshold method instead of the peak-picking method. In order to be able to compare

Table 1: Results of boundary estimation in terms of F-meas., Precision, Recall and AUC at 0.5 s. and 3 s. tolerance.

Model	±0.5 s. tolerance				±3 s. tolerance			
	F-m. (std)	Prec.	Rec.	AUC	F-m. (std)	Prec.	Rec.	AUC
① MLS + $subSSM^{mfcc}$	0.273 (0.132)	0.279	0.30	**0.810**	0.551 (0.158)	0.563	0.602	**0.946**
② MLS + $subSSM^{chroma}$	0.270 (0.135)	0.43	0.215	0.800	0.540 (0.153)	0.604	0.555	0.922
③ MLS + Depth($subSSM^{mfcc}$,$subSSM^{chroma}$)	**0.291** (0.120)	0.470	0.225	0.792	**0.629** (0.164)	0.755	0.624	0.930
③' MLS + Depth($subSSM^{mfcc}$,$subSSM^{chroma}$)	0.211 (0.08)	0.128	0.699	0.792	0.618 (0.156)	0.502	0.878	0.930
④[19] re-implemented: MLS+SSLM(MFCC)	0.246 (0.112)	0.291	0.239	0.774	0.580 (0.150)	0.666	0.568	0.927
⑤[19] published: MLS+SSLM(MFCC)	0.523	0.646	0.484					

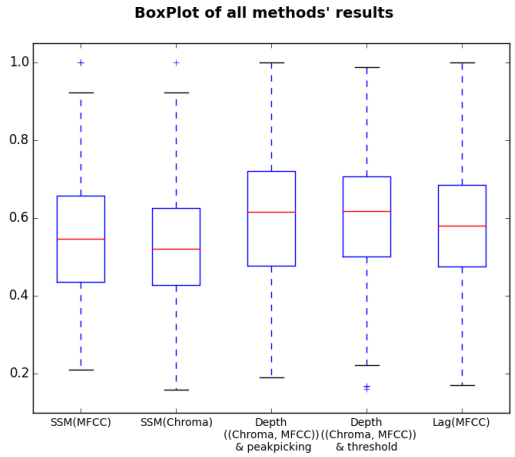

Fig. 4: Boxplot of the F-measure at ± 3 s. of all the models

our results to the ones of [19] (since we do not have the same dataset and since [19] did not publish their results at ±3 s.) we re-implemented their most successful model (the fusion of the MLS and SSLM through a convolutional layer) and apply it (training and testing) to our dataset. Results are presented in row ④. We also indicate the results published in [19] in row ⑤.

In Figure 4, we provide the corresponding boxplots[1] representation of the F-measure at ± 3s. From left to right: ①, ②, ③, ③' and ④.

4.4 Discussion

We first see that the size of the dataset plays a crucial role: our re-implementation of [19] (row ④) achieves much lower results (0.246 < 0.523) than the results published by the author (row ⑤).

We see that using the self-similarity-matrix expressed in time ① rather than in Lag (row ④) provides a small improvement at ± 0.5 s. in terms of F-measure (0.273 > 0.246) and AUC (0.810 > 0.774). At ± 3 s., this is only the case for the AUC (0.946 > 0.927) but not for the F-measure. We see that using the depth of the input layer to combine the two SSMs (row ③) allows to increase the F-measure at ± 0.5 s. and ± 3 s. This is due to a large increase of the Precision. It seems that using the two SSMs simultaneously allows reducing false detection. However, the AUC remains better using only one SSM of MFCC (row ①).

We see that replacing the peak-picking algorithm (row ③) by a direct threshold on the network output (row ③') decrease the results: a threshold on the output leads to a F-measure of 0.618 (resp.0.211), compared to 0.629 with the pick peaking (resp. 0.29). However, according to the boxplot representation (Figure 4), the variance is reduced.

Finally, considering the F-measure and our version of the SALAMI dataset, we see that our proposed model (row ③) performed the best in all cases and exceeds our re-implementation of the state-of-the-art system of [19].

5 Conclusion

In this paper we have proposed a new representation to be used as input to a Convolutional Neural Network in the goal of estimating music structure boundaries. We have proposed to use the square-sub-matrices centered on the main diagonal of a self-similarity matrix. In the case of homogeneous segments, this representation provides much sharper edges at the beginning and ending

[1] Each boxplot show the minimum, first quartile, median, third quartile, and maximum for one model through all the tracks in the test set.

of segments than the previously used Lag-Matrix. In order to represent various viewpoints on the content of a music track, we proposed to use square-sub-matrices extracted using two different audio features representing timbre (MFCCs) and harmony (chromas). We have proposed to combine them in a early fusion way using the depth of the input layers of a ConvNet. This fusion allows preserving the temporal relationship between the two representations. During a large experiment using the SALAMI dataset, we showed that the proposed representation allows improving music structure boundary estimation over our re-implementation of the state-of-the-art approach of Grill et al.

Acknowledgement. This research has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no 688122 (ABC_Dj project) and from the French National Research Agency under contract ANR-16-CE23-0017-01 (WASABI project).

References

[1] Boutard, G., Goldszmidt, S., and Peeters, G., "Browsing inside a Music Track, the Experimentation Case Study," in *Proc. of LSAS*, 2006.

[2] Peeters, G., Laburthe, A., and Rodet, X., "Toward Automatic Music Audio Summary Generation from Signal Analysis," in *Proc. of ISMIR*, 2002.

[3] Davies, M. E., Hamel, P., Yoshii, K., and Goto, M., "Automashupper: An Automatic Multi-Song Mashup System," in *Proc. of ISMIR*, 2013.

[4] Peeters, G., "Deriving Musical Structures from Signal Analysis for Music Audio Summary Generation: "Sequence" and "State" Approach," in *CMMR*, 2004.

[5] Peeters, G. and Deruty, E., "Is music structure annotation multi-dimensional? A proposal for robust local music annotation," in *Proc. of LSAS*, 2009.

[6] Bimbot, F., Deruty, E., Sargent, G., and Vincent, E., "Semiotic structure labeling of music pieces: Concepts, methods and annotation conventions," in *Proc. of ISMIR*, 2012.

[7] Smith, J. B. L., Burgoyne, J. A., Fujinaga, I., De Roure, D., and Downie, J. S., "Design and Creation of a Large-Scale Database of Structural Annotations." in *ISMIR*, 2011.

[8] Foote, J., "Visualizing Music and Audio Using Self-similarity," in *Proc. of ACM Multimedia Conference*, 1999.

[9] Logan, B. and Chu, S., "Music summarization using key phrases," in *Proc. of ICASSP*, 2000.

[10] Müller, M., Grosche, P., and Jiang, N., "A segment-based fitness measure for capturing repetitive structures of music recordings," in *In Proc. of ISMIR*, 2011.

[11] Kaiser, F. and Sikora, T., "Music Structure Discovery in Popular Music using Non-negative Matrix Factorization," in *Proc. of ISMIR*, 2010.

[12] Cooper, M. and Foote, J., "Summarizing popular music via structural similarity analysis," in *Proc. of WASPAA*, 2003.

[13] Goto, M. et al., "Development of the RWC music database," in *Proc. of ICA*, 2004.

[14] Bimbot, F., Deruty, E., Gabriel, S., and Vincent, E., "Semiotic Structure Labeling Of Music Pieces: Concepts, Methods And Annotation Conventions," in *Proc. of ISMIR*, 2012.

[15] McFee, B. and Ellis, D., "Learning to segment songs with ordinal linear discriminant analysis," in *Proc. of ICASSP*, 2014.

[16] Böck, S. and Schedl, M., "Enhanced beat tracking with context-aware neural networks," in *Proc. of DAFx*, 2011.

[17] Böck, S., Arzt, A., Krebs, F., and Schedl, M., "Online real-time onset detection with recurrent neural networks," in *Proc. of DAFx*, 2012.

[18] Ullrich, K., Schlüter, J., and Grill, T., "Boundary Detection in Music Structure Analysis using Convolutional Neural Networks," in *Proc. of ISMIR*, 2014.

[19] Grill, T. and Schlüter, J., "Music Boundary Detection Using Neural Networks on Combined Features and Two-Level Annotations," in *Proc. of ISMIR*, 2015.

[20] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-Based Learning Applied to Document Recognition," in *Intelligent Signal Processing*, IEEE Press, 2001.

[21] Bartsch, M. A. and Wakefield, G. H., "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proc. of IEEE WASPAA*, 2001.

[22] Foote, J., "Automatic audio segmentation using a measure of audio novelty," in *Proc. of ICME*, 2000.

[23] Kaiser, F. and Peeters, G., "Multiple Hypotheses At Multiple Scales For Audio Novelty Computation Within Music," in *Proc. of ICASSP*, 2013.

[24] Sainath, T. N., "Towards End-to-End Speech Recognition Using Deep Neural Networks," in *Proc. of ICML*, 2015.

[25] Qian, N., "On the momentum term in gradient descent learning algorithms," *Neural networks*, 1999.

[26] Kingma, D. and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] Mermelstein, P., "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, 1976.

[28] Fujishima, T., "Realtime chord recognition of musical sound: a system using Common LISP music," in *Proc. of ICMC*, 1999.