DEEP EMBEDDINGS AND SECTION FUSION IMPROVE MUSIC SEGMENTATION

Justin Salamon **Oriol Nieto** **Nicholas J. Bryan**
Adobe Research, San Francisco, CA, USA
salamon@adobe.com

ABSTRACT

Music segmentation algorithms identify the structure of a music recording by automatically dividing it into sections and determining which sections repeat and when. Since the desired granularity of the sections may vary by application, *multi-level* segmentation produces several levels of segmentation ordered by granularity from one section (the whole song) up to N unique sections, and has proven to be a challenging MIR task. In this work we propose a multi-level segmentation method that leverages deep audio embeddings learned via other tasks. Our approach builds on an existing multi-level segmentation algorithm, replacing manually engineered features with deep embeddings learned through audio classification problems where data are abundant. Additionally, we propose a novel section fusion algorithm that leverages the multi-level segmentation to consolidate short segments at each level in a way that is consistent with the segmentations at lower levels. Through a series of experiments we show that replacing handcrafted features with deep embeddings can lead to significant improvements in multi-level music segmentation performance, and that section fusion further improves the results by cleaning up spurious short sections. We compare our approach to two strong baselines and show that it yields state-of-the-art results.

1. INTRODUCTION

Audio-based music structure analysis, also known as music segmentation, is one of the most widely studied and challenging tasks in Music Information Retrieval [1]. The goal of this task is to obtain a series of non-overlapping sections (segments) defined by a set of temporal boundaries, and to identify and label which sections are repetitions of each other. Automatic segmentation could enable efficient intra-track navigation [2], assisted music creation [3], and section-based music retrieval and recommendation [4], to name some applications.

A key challenge in music segmentation is to capture the different possible levels of temporal granularity with which a song can be segmented. From an application perspective, a *multi-level* segmentation [1] that produces multiple segmentations ranging from coarse (e.g., 1-3 unique sections and their repetitions) to granular (e.g., 8-12) would allow application designers and/or end users to choose the level(s) of segmentation that best fits their needs. To facilitate this, datasets such as SALAMI [5] and SPAM [6] have been manually annotated with multiple segmentation levels based on length (e.g., long-scale sections, short-term motives) or functional role (e.g., "sax solo," "outro"). Metrics to evaluate multi-level segmentation have also been proposed recently [7,8] and adopted by the community [9].

Most segmentation methods yield just one level. An early approach identified sharp differences in time series of audio features related to timbre and harmony by running a checkerboard kernel along the diagonal of a self-similarity matrix [10]. More sophisticated handcrafted features were later proposed, yielding superior boundary detection [11]. Currently, the best boundary detection is obtained with deep learning models, such as a deep convolutional neural network (CNN) [12] or deep metric learning which yields an effective feature space for boundary detection [13]. Multi-level approaches appeared more recently, and just a handful have been proposed to date. McFee and Ellis apply spectral clustering to a self-similarity matrix obtained via a simple combination of DSP features [14], an approach later enhanced by Tralie and McFee by adding harmonic embeddings from a convolutional-recurrent neural network and using Similarity Network Fusion (SNF) to combine features [9]. Supervised approaches include ordinal linear discriminant analysis [15] and a CNN that outputs two segmentation levels [16].

We propose an approach [2] that builds on the work of McFee and Ellis [14]. We summarize our contributions as follows: (1) we propose replacing or augmenting the handcrafted features with deep audio embeddings that can robustly capture various similarity and repetition cues; (2) we introduce a multi-level section fusion algorithm that leverages the different segmentation levels, consolidating short sections to produce a cleaner and more consistent segmentation across levels; (3) through a series of experiments and qualitative analysis, we demonstrate the effectiveness of each of our contributions. We compare our approach to strong baselines and show that it produces state-of-the-art results for multi-level music structure segmentation.

[1] Also known as hierarchical structure segmentation, but since the levels do not strictly form a hierarchy, we use multi-level segmentation.

[2] Code: github.com/justinsalamon/musicseg_deepemb

2. LAPLACIAN STRUCTURAL DECOMPOSITION

We start with an overview of Laplacian Structural Decomposition (LSD) [14], which forms the basis for our method.

A *recurrence matrix* captures the similarity between feature frames of a track, and can expose song structure [11]. It is a binary, squared, symmetrical matrix R such that $R_{ij} = 1$ if frames i and j are similar–for a specific metric, e.g., cosine distance–and $R_{ij} = 0$ otherwise. McFee and Ellis [14] treat the recurrence matrix as an unweighted, undirected graph, where each frame is a vertex and 1's in the recurrence matrix represent edges. Then they apply *spectral clustering* [17] (unrelated to audio spectrograms), yielding a per-frame cluster assignment. Sections are derived by grouping frames by their cluster assignment. Sections with the same cluster ID represent repetitions.

In the original approach, R is obtained by combining two recurrence matrices obtained from audio features: R^{loc} computed from mel-frequency cepstral coefficients (MFCC) to identify local similarity between consecutive frames, and R^{rep} computed from Constant-Q transform (CQT) features to capture repetition across the entire track. The goal is to detect sudden sharp changes in timbre with R^{loc}, while capturing long-term harmonic repetition with R^{rep}. These matrices are combined via a weighted sum controlled by a hyper-parameter $\mu \in [0, 1]$, which can be set manually or automatically [14]:

$$R = \mu R^{\text{rep}} + (1 - \mu) R^{\text{loc}} \tag{1}$$

The number of unique sections produced by the segmentation is equal to the number of clusters N used for spectral clustering. By clustering with increasing $N = 1...M$ we obtain a multi-level segmentation: the higher M is, the finer the resulting segmentation [18]. The clustering uses an eigenvalue decomposition, such that for a given M the data are projected onto the first M eigenvectors (ordered by their eigenvalues) of the symmetrical normalized Laplacian of R and then clustered. The key takeaway is that the same eigenvectors are reused for increasing M (each time adding one more), meaning cluster assignments at different levels are related. This property is essential for the multi-level section fusion algorithm we present later.

3. DEEP AUDIO EMBEDDINGS

We replace or augment the handcrafted features in LSD with deep audio embeddings learned via other tasks, making this a transfer learning approach. We propose to: (1) replace the MFCC features with deep embeddings learned via Few-Shot Learning (FSL) [19], and (2) augment the CQT features with deep embeddings learned via a state-of-the-art music auto-tagging model designed to capture music similarity across genre, mood, tempo, and era [20].

3.1 Few-shot Learning Embeddings

The purpose of the MFCC features used in the LSD method is to capture local (short-term) timbre similarity, with the goal of identifying sharp transitions as potential boundary locations. However, MFCC have been shown to be sensitive to noise [21], and so we hypothesize that an audio feature that captures short-term timbre similarity more robustly could lead to better boundary detection.

To this end, we employ the Few-Shot Sound Event Detection model recently proposed by Wang et al. [22]. Few-shot learning (FSL) is an area of machine learning which aims to train models that are able, once trained, to robustly recognize a new class given a handful of examples of the new class at inference time [19]. Wang et al. showed that Prototypical Networks [19], a metric-based approach originally proposed for FSL on images, can be successfully applied to the audio domain given the right adaptations. Importantly, Prototypical Networks do not require fine-tuning or retraining. Rather, they are used to embed audio such that perceptually similar sounds are also close in the embedding space, as shown by Wang et al. [22, 23]. As such, these embeddings, which are computed from a 0.5 second window, can be viewed as a general-purpose, short-term, timbre similarity feature. Wang et al. focused on the task of sound event detection (SED), training and evaluating the model on few-shot word recognition via an annotated speech corpus. We refer the reader to this study for further details about the model architecture and training [22].

As this model was trained on hundreds of thousands of audio samples, we hypothesize that the resulting audio embedding will be more robust compared to MFCC for capturing short-term timbre similarity. We replace MFCC with these embeddings, henceforth FSL, to compute R^{loc}. We use the model trained by Wang et. al, courtesy of the authors. Even though the model was trained on speech audio data, preliminary experiments indicated it captures timbre similarity for music too–a form of transfer learning.

3.2 Music Similarity Embeddings for Repetition

In LSD the repetition recurrence matrix R^{rep} is obtained using Constant-Q transform (CQT) features [24] computed from the audio signal after applying Harmonic-Percussive Source Separation (HPSS) [25] to enhance the harmonic components of the audio signal. Still, not all songs exhibit harmonic repetition, e.g., an EDM song may exhibit repetition of timbre (presence/absence of a beat, a high- or low-pass filter that is applied in specific sections, etc.). Many Western popular music songs use the same harmonic progression for both the verse and chorus, with only the instrumentation and lyrics indicating a section change.

We propose to use, in addition to CQT, deep audio embeddings that can capture other complementary music qualities that may be indicative of repetition, such as instrumentation, tempo, and mode. To achieve this, we leverage the deep music auto-tagger presented by Lee et al. [20]. In their work, the authors contrast classification and metric learning for training a deep music embedding that can be used for similarity-based music retrieval. Of the approaches compared, disentangled multi-task classification yielded an embedding that gave the best music retrieval results, in addition to producing state-of-the-art results for music auto-tagging. Here, *disentangled* means that the em-

bedding space is divided into sub-spaces that capture different dimensions of music similarity. The full embedding of size 256 is divided into four disjoint subspaces, each of size 64, where each subspace captures similarity along one musical dimension: genre, mood, tempo, and era. We refer the reader to Lee et al. [20, 26] for further details about the model architecture and optimization.

We hypothesize that this embedding, which is obtained from a 3-second context window and was trained on the Million Song Dataset [27], captures musical qualities that can be complementary to those captured by the CQT: genre is often a reasonable proxy for instrumentation; mood can be a proxy for tonality and dynamics; tempo is an important low-level quality in itself; and era, in addition to being related to genre, can be indicative of mixing and mastering effects. Combined, the full embedding, henceforce referred to as DEEPSIM, may surface repetitions along dimensions that are not captured by the CQT.

3.3 Fusing Similarity Matrices

In Section 2 we explained that the LSD method uses two matrices: R^{loc} (from MFCC) and R^{rep} (from CQT features). Now, we replace MFCC with FSL features for computing R^{loc}. For computing R^{rep} we do not replace the CQT features but rather combine them with the DEEPSIM embeddings, since they are potentially complementary. We do this via another weighted sum controlled by a hyperparameter $\gamma \in [0, 1]$, leading to the following equation:

$$R = \mu \left(\gamma R^{\text{DEEPSIM}} + (1-\gamma) R^{\text{CQT}}\right) + (1-\mu) R^{\text{FSL}} \quad (2)$$

All three matrices are normalized prior to being combined to ensure their values are in the same $[0, 1]$ range. This simplistic approach to feature fusion may not be optimal, and indeed more advanced fusion techniques such as Similarity Network Fusion [9] have been proposed. However, this approach has the advantage of allowing us to easily and clearly study the relative importance of the features we are proposing to use: μ controls the relative importance of local versus repetition similarity, while γ controls the relative importance of CQT versus DEEPSIM features for repetition similarity. We set our initial parameterization to $\mu = 0.5, \gamma = 0.5$, meaning we give equal weight to local similarity obtained via FSL features and repetition similarity, which is given by the simple average of the R^{CQT} and R^{DEEPSIM} matrices. Later on we will explore the impact of varying these parameters on two different datasets to gain insight about feature relevance for different data.

4. MULTI-LEVEL SECTION FUSION

In Section 2 we explained how segmentation is achieved by clustering each frame of the audio signal. This assigns each frame a cluster ID, and then consecutive frames with the same cluster ID are grouped to form sections. This process can sometimes result in a small number of consecutive frames having a different cluster ID to those around them, leading to very short sections. These often do not represent actual sections in the song, and even when they do, they may not be helpful to the end user or application.

The LSD method attempts to alleviate this issue via smoothing: it applies median filtering to R^{Rep} to enhance diagonals in the matrix before it is combined with R^{Loc}, and also applies median smoothing to the vectors obtained via spectral clustering. Even with this smoothing, we found that our approach (and LSD) can still produce spurious short sections. Smoothing more aggressively would deteriorate the temporal accuracy of the boundaries between sections, and is thus undesirable. What is more, it is unlikely for there to be a single "best" minimal section duration for all use cases. Rather, the appropriate lower bound on section duration will depend on the application.

We address this challenge in a way that allows the user to define their desired minimal section duration. Given the desired minimal duration in seconds, we now need to remove sections whose duration is below this value, henceforth "short sections", by fusing them with the previous section, the next section, or both. But, how do we determine which section(s) to fuse with? We propose an algorithm that leverages multi-level segmentation to solve this.

4.1 Multi-level Section Fusion Algorithm

Our method leverages two heuristics: (1) section IDs should mostly be consistent with overlapping sections at lower segmentation levels, since we are reusing the same eigenvectors for clustering (cf. Section 2); (2) section boundaries should be mostly consistent across segmentation levels, and thus a boundary that overlaps with boundaries at lower levels of the hierarchy is more likely to be a real boundary compared to one that does not.

The algorithm works as follows: say we need to fuse a short section at level n. If the section is the first or last of the song, we merge it to the next or previous section respectively, as that is our only option. If the section is in the middle of the song *and* both the previous and next sections have the same ID, we merge all three together. These three simple scenarios are depicted in Figure 1a. Otherwise, we need to determine whether to merge the short section with the previous *or* the next section at level n. So, we look one level down in the hierarchy ($n-1$) and find the section that overlaps the most with our short section. If the ID of the overlapping section at level $n-1$ matches the ID of either the previous or next sections at level n, we merge the short section with the matching section (Figure 1b). If the overlapping ID at level $n-1$ does not match the ID of neither the previous nor the next section at level n, we go down to level $n-2$ and try again (Figure 1c), and so on until we find an overlapping section whose ID matches the ID of either the previous or next section at level n.

If we reach the bottom level and still do not find a match, we turn to our second multi-level cue: the boundaries (Figure 1d). Our short section at level n has two boundaries, let's call them "start" and "end." For each of the two we count how many boundaries they overlap with at all lower segmentation levels, where we consider boundaries at different levels to overlap if they are within one second of each other. Whichever of the two (start or end) overlaps with the most boundaries at lower levels is more

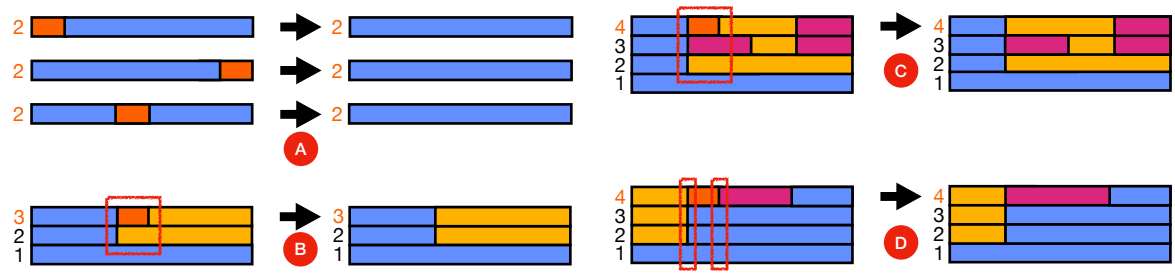

Figure 1: Multi-level section fusion algorithm: in all plots the short orange section needs to be fused. The number to the left of each segmentation represents its level, with the number in orange indicating the level being cleaned: (A) three simple fusion scenarios, (B) fusion via multi-level segmentation IDs: level $n-1$ gives us the answer, (C) same as (B) but this time level $n-2$ gives us the answer, (D) fusion via multi-level segmentation boundaries: the start boundary is consistent with boundaries at levels $n-1$ and $n-2$, whereas the end boundary is not consistent with any boundaries at lower levels.

likely to be a real boundary, so we keep that boundary and remove the other by fusing the short section to the adjacent section separated by the "losing" boundary.

We repeat the entire process until there are no short sections left at level n. We iterate over the sections in a double loop: the outer loop iterates over section IDs, from the highest to the lowest. The section ID corresponds to the eigenvector to which the section was clustered. By iterating in this way, we are more likely to keep sections with lower IDs, which in turn are more likely to appear at lower levels of the hierarchy. Within each section ID, our inner loop iterates over the sections from shortest to longest. We have found that this approach leads to more coherent section fusion across the entire hierarchy.

5. EXPERIMENTAL DESIGN

5.1 Datasets

We experiment with two datasets: Harmonix [28] and SALAMI [5], which are the largest datasets published to date with structural segmentation annotations. They are notably different in terms of audio content and annotations.

The Harmonix set contains 912 tracks of Western popular music that have been manually annotated with functional sections ("intro," "chorus," "solo," etc.). As such, the annotations are not multi-level (i.e., they are "flat") and represent a single segmentation level with one annotator per track. This dataset was compiled by the video game company Harmonix with the goal of incorporating music segmentation into some of its music games (e.g., Rock Band, Guitar Hero). This makes it highly relevant for evaluating algorithms that will be applied in real-world applications focusing on Western music.

The SALAMI set is comprised of 1,355 tracks spanning a wide range of musical genres including classical, jazz, non-Western music, live performances, etc. Each track is manually annotated with three levels: (1) functional segments representing sections such as "guitar solo," "verse," etc.; (2) larger-scale segments representing longer structural sequences, annotated with upper-case letters, e.g., A, B, C'; (3) small-scale segments capturing shorter time scales in the song that may include melody lines or motifs, annotated with lower-case letters, e.g., a, b', c. To be comparable to previous work [9], we evaluate against the large- (2) and small-scale (3) annotations, limiting our test set to tracks that have two or more annotations (884 tracks). The remainder (471) are used for hyper-parameter tuning.

5.2 Metrics

The L-Measure (L-M) is the preferred metric for evaluating multi-level segmentations [7]. It treats music segmentation as a similarity ranking problem, capturing both boundary alignment (otherwise evaluated as a binary classification problem) and segment labeling (otherwise evaluated as a clustering problem). To compute L-M, we divide the reference annotation into time points (frames), and compare each point t against all other time points. If t is closer to point u than point v when considering all segmentation levels, we represent it as a triplet (t, u, v). We repeat the same process for the algorithm's estimate segmentation. We then define the L-Precision (L-P) as the fraction of estimate triplets that match reference triplets, the L-Recall (L-R) as the fraction of reference triplets that match estimate triplets, and L-M as their harmonic mean.

Our method and the baselines we compare it against produce much deeper segmentations than the two levels annotated in SALAMI or the single level annotated in Harmonix, meaning the L-Precision may not be sufficiently reliable [7, 9]. Conversely, the L-Recall, which captures how well the structure defined in the reference is retrieved in the estimation is, in this context, a more trustworthy metric, and so we focus on it in our study. Still, for completeness we report all three L metrics (L-P, L-R, L-M).

We also report the standard metrics used to evaluate flat music segmentation: Hit Rate for boundary retrieval at 0.5 and 3 second tolerance windows, $HR_{0.5}$ and HR_3, and the Pairwise Frame Clustering (PFC) [29] and Normalized Conditional Entropies (NCE) [30] for segment labeling. Since our application scenario assumes the preferred segmentation level will be preset by the application designer or set by the end user based on their needs, we simulate this scenario in our evaluation by computing these metrics for each track using the segmentation level that maximizes the metrics. For conciseness, we only report the aggregated harmonic mean for each of the flat metrics.

Deep Embs.	Sec. Fusion	L-P	L-R	L-M
No	No	36.70	65.77	46.82
No	Yes	38.07	66.68	47.92
Yes	No	40.91	**76.56**	53.01
Yes	Yes	**43.50**	76.47	**55.01**

Table 1: Ablation results on the Harmonix dataset.

5.3 Baselines

We compare our approach against two baselines: the original LSD method [14], and its improved variant that adds deep harmonic embeddings and uses Similarity Network Fusion (SNF) to combine the recurrence matrices [9]. While the latter baseline also leverages deep embeddings, they are only designed to capture harmony, unlike our multiple deep embeddings which capture a variety of musical properties. We use the LSD implementation from MSAF [6] and the SNF implementation released by the authors. [3] LSD and our method use beat-aligned features, for which we use the beat tracker by Korzeniowski et al. [31] implemented in the madmom package [32], as it has been been shown to yield better segmentation results [28] compared to the default beat tracker in Librosa [33]. SNF does not rely on beat tracking.

5.4 Ablations

To demonstrate the effectiveness of our contributions, we perform a systematic ablation compared to LSD: we fix $\mu = 0.5$ for LSD and $\mu = \gamma = 0.5$ for our approach, and then compare LSD, LSD + section fusion, our method (LSD + deep embeddings) without section fusion, and finally our full method (LSD + deep embeddings + section fusion). For section fusion we set the minimum section duration to 8 seconds (=4 bars at 120 bpm) as a reasonable lower bound. In a real world scenario this value could be chosen by an end user or preset by the application designer.

6. RESULTS

6.1 Ablations

In Table 1 we present the results of the ablations described above on the Harmonix dataset. We see that our deep embeddings and section fusion independently improve the baseline. Noteworthy is the dramatic increase in L-Recall due to our proposed deep embeddings. Combining the deep embeddings with section fusion improves L-Precision and thus the overall L-M. Though omitted from Table 1 for conciseness, we also confirmed that just replacing MFCC with FSL (without DEEPSIM) improves over the baseline, strengthening our hypothesis from Section 3.1.

6.2 Feature Importance

To understand the relative importance of the features used in our approach, we run a grid search over μ and γ, focusing on L-Recall (cf. Sec. 5.2). Remember that μ controls the ratio of local similarity (FSL embeddings) to repetition, while γ controls the relative contribution of the DEEPSIM embeddings versus the CQT features to repetition.

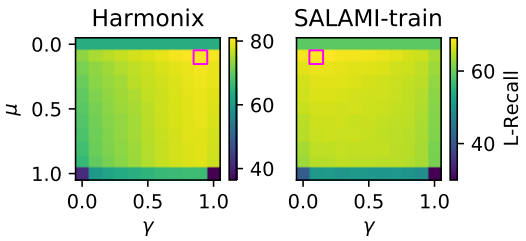

Figure 2: Grid search for μ and γ, red square is best.

We present the results in Figure 2 for Harmonix and SALAMI, with the L-Recall maxima marked by a square. First, we note that results always worsen when $\mu = 0$ or 1, illustrating the importance of combining both local similarity and repetition matrices. While less pronounced, the same is true of γ, showing that both datasets benefit from combining DEEPSIM and CQT features for repetition.

For Harmonix, performance is maximized when $\mu = 0.1, \gamma = 0.9$: most weight goes to local repetition via FSL, with most of the remainder going to DEEPSIM features for repetition. On the other hand, for SALAMI performance is maximized when $\mu = \gamma = 0.1$, i.e., most of the weight goes to FSL with the remainder going to CQT features.

The difference in optimal parameter values per dataset warrants discussion. A small μ in both cases highlights the importance of local similarity information (FSL), regardless of music genre. On the other hand, we see the DEEPSIM embeddings are preferred when segmenting Western popular music (Harmonix), while CQT features are favored for SALAMI which is more diverse. One possible explanation is that DEEPSIM was trained on a subset of MSD that leans more heavily toward Western popular music compared to SALAMI [34]. Still, it is always beneficial to use a combination of both features.

6.3 Multi-level and Flat Results

We compare our approach against two strong baselines representing the state-of-the-art in multi-level music structure segmentation [9, 14]. We compute the baselines using the same setup reported by their authors. LSD sets μ automatically per-track in a data-driven fashion. For our approach, Deep Embeddings with section Fusion (DEF), we report results for three parameter configurations: (1) $\mu = \gamma = 0.5$, (2) optimal values obtained via grid search on SALAMI-train (μ^S, γ^S), (3) optimal values obtained via grid search on Harmonix (μ^H, γ^H) [4].

The full multi-level segmentation results are presented in Table 2. For SALAMI, we see that the parameter values obtained by optimizing DEF over SALAMI-train generalize well to the test set, beating all other methods in terms of multi-level segmentation (L-P, L-R, L-M) and setting a new state of the art. Turning to Harmonix, we see that DEF outperforms the baseline for all three parameter configurations, setting a new state of the art for this dataset too. Most

[3] https://github.com/ctralie/GraphDitty

[4] These may be artificially inflated due to the lack of train/test splits for the Harmonix dataset.

Method	SALAMI			Harmonix		
	L-P	L-R	L-M	L-P	L-R	L-M
LSD [7]	41.89	63.60	49.77	39.05	69.33	49.63
SNF [9]	43.08	66.82	51.65	36.38	67.47	47.01
$\text{DEF}_{0.5,\,0.5}$	42.43	64.51	50.38	**43.50**	76.47	55.01
$\text{DEF}_{\mu^S,\,\gamma^S}$	**43.46**	**67.30**	**52.02**	42.63	75.43	54.06
$\text{DEF}_{\mu^H,\,\gamma^H}$	41.64	66.06	50.38	43.23	**81.03**	**56.04**

Table 2: Multi-level segmentation results.

Method	SALAMI				Harmonix			
	$\text{HR}_{0.5}$	HR_3	PFC	NCE	$\text{HR}_{0.5}$	HR_3	PFC	NCE
LSD [7]	31.99	47.46	56.13	59.00	40.69	56.50	61.21	57.43
SNF [9]	29.17	45.59	56.73	59.98	26.57	51.42	57.38	54.86
$\text{DEF}_{0.5,\,0.5}$	**33.78**	55.65	59.44	62.16	**45.74**	68.84	70.11	66.48
$\text{DEF}_{\mu^S,\,\gamma^S}$	32.07	53.91	**59.99**	**62.39**	43.18	67.34	69.34	65.44
$\text{DEF}_{\mu^H,\,\gamma^H}$	31.79	**56.35**	58.56	61.48	41.61	**71.17**	**71.49**	**67.59**

Table 3: Flat segmentation results.

sections in the "lowercase" level of SALAMI are shorter than 8 s which, given our section fusion, may explain why the improvement is moderate compared to Harmonix. Furthermore, SALAMI contains various tracks with limited inter-annotator agreement [6], making it harder for an algorithm to match the reference annotations.

Finally, we report the flat results in Table 3 (SALAMI is evaluated against the "uppercase" level). Note that these results mimic the behavior of a user choosing their desired segmentation level, as described in Section 5.2. Similar to the multi-label results, vanilla DEF also outperforms the baselines on all flat metrics in both datasets, including $\text{HR}_{0.5}$ which is the strictest metric for boundary retrieval.

6.4 Qualitative Analysis

To gain further insight into how our approach compares to the LSD baseline, we examine the multi-level segmentations they produce for a particular track in the Harmonix dataset: track 199, "The Number of the Beast" by Iron Maiden (we encourage the reader to listen to this track to better follow this section). The multi-level segmentations are shown in Figure 3 (we use vanilla $\text{DEF}_{0.5,\,0.5}$) with the reference boundaries overlaid as vertical magenta lines and the reference section labels printed at the top of each plot.

It is apparent that the baseline method produces many more noisy segments (i.e., too short, not pertinent) compared to our approach. Particularly relevant is the "breakdown" segment, where there are multiple changes in terms of instrumentation: the drums stop suddenly, the rhythm and bass guitars change riffs drastically, and all this occurs between two different guitar solos ("solo" and "solo2"). The baseline method detects these short changes starting at level 3, without being able to detect the whole "breakdown" as a single whole section. This also prevents it from recognizing other important sections at levels 3 and 4 such as "inst," which is where the drums kick in along with a loud and long scream, or the "verse" and "chorus" sections, since the new unique sections introduced at these levels are "cannibalized" by the changes in the "breakdown."

In contrast, our method detects the breakdown as a segment starting from level 6, labeling it similarly to the "bridge," which makes musical sense given that both parts are instrumental and quite different to all others parts. The

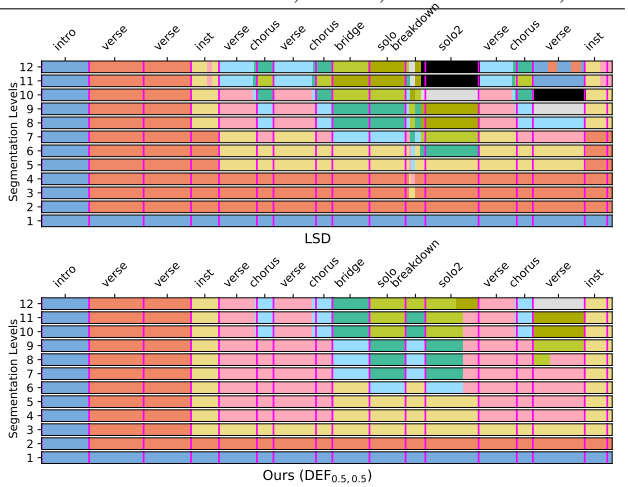

Figure 3: Segmentation of Harmonix track 199: LSD (top) and $\text{DEF}_{0.5,\,0.5}$ (bottom). Ground truth in vertical lines.

absence of noisy short segments in our approach can be attributed, in all likelihood, to our proposed section fusion algorithm. Our method successfully captures the drum entrance in level 3, identifying three highly differentiated long segments: spoken word intro (blue), music with minimal drums (orange), and music with full drums (yellow). Successfully capturing these key changes in timbre can be attributed to our introduction of the proposed deep embeddings. Overall, it is apparent in this example that our method obtains notably cleaner sections that better align to the reference annotations thanks to both the deep embeddings and the multi-level section fusion algorithm.

7. CONCLUSION

In this work we introduced a multi-level segmentation method that leverages deep audio embeddings learned via other tasks. Building on an existing multi-level segmentation algorithm based on spectral clustering, we replaced MFCC features with deep embeddings trained via Few-Shot Learning for computing local timbre similarity. We also augmented the CQT features used to identify section repetition with deep embeddings from a state-of-the-art music auto-tagging model that captures similarity along different music dimensions. Next, we introduced a novel section fusion algorithm that leverages the multi-level segmentation to consolidate short segments. Through a series of experiments we showed that our two key contributions–replacing the handcrafted features with our proposed deep embeddings and applying multi-level section fusion–lead to significant improvements in multi-level music segmentation, outperforming two strong baselines and yielding state-of-the-art results. Finally, we complemented our quantitative results with a qualitative analysis to gain further insight into how our proposed enhancements improve segmentation performance. Future work includes evaluating a broader range of deep embeddings with our segmentation approach such as OpenL3 [35] and VGGish [36], exploring advanced feature fusion approaches such as SNF [9], and investigating automated strategies for determining the optimal minimum section duration for section fusion.

8. ACKNOWLEDGMENTS

We would like to thank the authors of our baselines, Brian McFee, Daniel P.W. Ellis, and Christopher Tralie for making their code publicly available and reproducible.

9. REFERENCES

[1] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 246–263, 2020.

[2] M. Schedl, "Intelligent user interfaces for social music discovery and exploration of large-scale music repositories," in *Proceedings of the 2017 ACM Workshop on Theory-Informed User Modeling for Tailoring and Personalizing Interfaces*, ser. HUMANIZE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 7–11. [Online]. Available: https://doi.org/10.1145/3039677.3039678

[3] H. Jhamtani and T. Berg-Kirkpatrick, "Modeling self-repetition in music generation using generative adversarial networks," 2019.

[4] A. Bozzon, G. Prandi, G. Valenzise, and M. Tagliasacchi, "A music recommendation system based on semantic audio segments similarity," in *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications*, ser. EuroIMSA '08. USA: ACTA Press, 2008, p. 182–187.

[5] J. B. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and Creation of a Large-Scale Database of Structural Annotations," in *Proc. of the 12th International Society of Music Information Retrieval*, Miami, FL, USA, 2011, pp. 555–560.

[6] O. Nieto and J. P. Bello, "Systematic Exploration of Computational Music Structure Research," in *Proc. of the 17th International Society for Music Information Retrieval Conference*, New York City, NY, USA, 2016, pp. 547–553.

[7] B. McFee, O. Nieto, M. M. Farbood, and J. P. Bello, "Evaluating hierarchical structure in music annotations," *Frontiers in Psychology*, vol. 8, no. 1337, 2017.

[8] B. McFee and K. M. Kinnaird, "Improving Structure Evaluation Through Automatic Hierarchy Expansion," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 152–158.

[9] C. J. Tralie and B. McFee, "Enhanced Hierarchical Music Structure Annotations via Feature Level Similarity Fusion," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 201–205, 2019.

[10] J. Foote, "Automatic Audio Segmentation Using a Measure Of Audio Novelty," in *Proc. of the IEEE International Conference of Multimedia and Expo*, New York City, NY, USA, 2000, pp. 452–455.

[11] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity," *IEEE Transactions on Multimedia, Special Issue on Music Data Mining*, vol. 16, no. 5, pp. 1229 – 1240, 2014.

[12] K. Ullrich, J. Schlüter, and T. Grill, "Boundary Detection in Music Structure Analysis Using Convolutional Neural Networks," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 417–422.

[13] M. C. McCallum, "Unsupervised Learning of Deep Features for Music Segmentation," in *Proc. of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019.

[14] B. McFee and D. P. W. Ellis, "Analyzing Song Structure with Spectral Clustering," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 405–410.

[15] ——, "Learning to Segment Songs With Ordinal Linear Discriminant Analysis," in *Proc. of the 39th IEEE International Conference on Acoustics Speech and Signal Processing*, Florence, Italy, 2014, pp. 5197–5201.

[16] T. Grill and J. Schlüter, "Music Boundary Detection Using Neural Networks on Combined Features and Two-level Annotations," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Málaga, Spain, 2015.

[17] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, aug 2007.

[18] H. Grohganz, M. Clausen, N. Jiang, and M. Müller, "Converting Path Structures into Block Structures using Eigenvalue Decomposition of Self-Similarity Matrices," in *Proc. of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[19] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017.

[20] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Metric learning vs classification for disentangled music representation learning," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[21] C. V. Cotton and D. P. W. Ellis, "Spectral vs. spectro-temporal features for acoustic event detection," in *IEEE Worksh. on Apps. of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, Oct. 2011, pp. 69–72.

[22] Y. Wang, J. Salamon, N. J. Bryan, and J. P. Bello, "Few-shot sound event detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 81–85.

[23] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, "Few-shot drum transcription in polyphonic music," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2020.

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

[25] D. Fitzgerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, vol. 13, 2010.

[26] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6–10.

[27] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *12th Int. Soc. for Music Info. Retrieval Conf.*, Miami, USA, Oct. 2011, pp. 591–596.

[28] O. Nieto, M. McCallum, M. E. Davies, A. Robertson, A. Stark, and E. Egozy, "The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 565–572.

[29] M. Levy and M. Sandler, "Structural Segmentation of Musical Audio by Constrained Clustering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 318–326, feb 2008.

[30] H. Lukashevich, "Towards Quantitative Measures of Evaluating Song Segmentation," in *Proc. of the 10th International Society of Music Information Retrieval*, Philadelphia, PA, USA, 2008, pp. 375–380.

[31] F. Korzeniowski, B. Sebastian, and G. Widmer, "Probabilistic Extraction of Beat positions from a Beat Activation Function," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 513–518.

[32] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[33] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proc. of the 14th Python in Science Conference*, Austin, TX, USA, 2015, pp. 18–25.

[34] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2392–2396.

[35] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen and learn more: Design choices for deep audio embeddings," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 3852–3856.

[36] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, Mar. 2017, pp. 131–135.