2018 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2018, AALBORG, DENMARK

DETECTION OF CUT-POINTS FOR AUTOMATIC MUSIC REARRANGEMENT

Daniel Stoller
Queen Mary University of London
d.stoller@qmul.ac.uk

Vincent Akkermans
MXX Music
vincent@mxxmusic.com

Simon Dixon
Queen Mary University of London
s.e.dixon@qmul.ac.uk

ABSTRACT

Existing music recordings are often rearranged, for example to fit their duration and structure to video content. Often an expert is needed to find suitable cut points allowing for imperceptible transitions between different sections. In previous work, the search for these cuts is restricted to the beginnings of beats or measures and only timbre and loudness are taken into account, while melodic expectations and instrument continuity are neglected. We instead aim to learn these features by training neural networks on a dataset of over 300 popular Western songs to classify which note onsets are suitable entry or exit points for a cut. We investigate existing and novel architectures and different feature representations, and find that best performance is achieved using neural networks with two-dimensional convolutions applied to spectrogram input covering several seconds of audio with a high temporal resolution of 23 or 46 ms. Finally, we analyse our best model using saliency maps and find it attends to rhythmical structures and the presence of sounds at the onset position, suggesting instrument activity to be important for predicting cut quality.

1. INTRODUCTION

Rearranging an existing recording of a music piece has many applications, such as creating remixes or adapting background tracks to video content. Normally, the global structure of the music piece is changed, while leaving other aspects unaltered. This process can be modeled as selecting pairs of *entry* and *exit* cut positions so that switching from exit to entry cuts yields good-sounding transitions between different sections.

But good transition quality requires fulfilling the listener's expectations regarding instrumentation, rhythm and many other musical aspects, and so finding suitable cut-points is time-consuming even for professional audio engineers. Therefore, we propose an automatic algorithm, so cut-points can be matched quickly by hand or automatically according to their position in the bar and the musical segment, paving the way for automatic music editing that saves time and is easy to use.

In contrast to previous approaches that assume features such as timbre or loudness are sufficient in predicting transition quality, we train classifiers directly on annotated entry and exit cuts to learn the relevant features from the data, and find that other aspects such as note onsets are also important.

This work was partially funded by EPSRC grant EP/L01632X/1.

2. RELATED WORK

To find suitable cut points, methods originally developed for image retargeting such as *tiling*, *stitching* [1] and *seam-carving* [2] were transferred to the music domain. However, these combine very short excerpts, ignoring the higher-level structure of music such as melodic motifs and thus often creating annoying repetitions [3]. Therefore, music-specific methods have been proposed, such as a multi-scale approach [4], which was improved upon with a genetic algorithm [5] and then adapted to only consider beginnings of whole measures as cut points [6]. A good performance of these systems was only shown for dance music featuring a strong and regular pulse, but is unlikely to transfer across many music genres.

Similar work [3] is not only focused on dance music and offers extra functionality, such as providing infinite music streams. The author uses the musical content between two consecutive beats as atomic "building blocks" that can be concatenated to form new pieces, and compute a pair-wise timbral similarity to determine the best transitions between them. However, cuts between different counts in a measure can occur because the position of downbeats is ignored.

All of the above approaches employ a manually defined measure to find suitable transitions. Because the underlying musicological phenomenon is not well understood, this definition is difficult and leads to measures that lack robustness and do not accurately take into account all relevant aspects of transition quality [3]. Instead, we apply deep learning to discover the relevant factors from a hand-annotated dataset of over 300 popular Western songs to derive a more accurate measure of transition quality. We also gain insight into what makes a good cut by hyper-parameter optimisation and model visualisation, revealing the influence of different frequency ranges, temporal resolutions and temporal context, and the importance of instrument activity and the amount of acoustic energy as relevant factors.

3. APPROACH

In [3–5] music rearrangement is framed as an optimisation problem involving the search for a set of pairs of cut points, so that the transitions resulting from the cut pairs are minimally perceptible. The transition quality is estimated depending on the combination of entry and exit cut.

In contrast, we aim to focus on finding entries and exits

978-1-5386-5477-4/18/$31.00

that produce good transitions regardless of which exit or entry they are paired with. Internal listening experiments have demonstrated that this can still produce good transitions, provided the cut candidate selection is more restrictive. As an exception to the above, we only allow transitions preserving the position in the metrical structure due to the listener's sensitivity to violations of metrical expectations.

We will focus on finding transitions between musical sections and therefore train classifiers to detect entries and exits near section boundaries using the local audio context as input. With suitable annotations, our approach could be generalized to not require structural segmentation information.

3.1. Dataset

Our dataset contains 311 popular Western songs in CD quality with a mostly modern pop style. The musical structure of each song is annotated with segments serving a musical function, such as verse and chorus. In addition, it contains automatically detected beat and onset positions that were manually corrected, as well as time signatures and downbeat positions.

We restrict our search for cut candidates to note onsets, which were manually annotated by an expert as entries or exits, or both, so that any combination of entry and exit produces a suitable transition. Note that previous work employed stricter restrictions, in which only beat positions [3–5] or even only beginnings of measures [6] were considered. This results in 28108 training examples, of which 19049 are entries and 19747 are exits.

For an automatic classification of the entries and exits annotated in the dataset, we have to generate negative examples from the songs in the dataset by carefully sampling onset positions. We found that over 99% of cut-points to are located between 8 beats before and 4 beats after their nearest section boundary. By relying on the structural segmentation, we thus simplified the task to only classifying onsets near section boundaries. Therefore, we sampled one negative example randomly from onsets in the 8+4 beat interval around each section boundary of each exit and entry cut, yielding a total of 38796 examples.

3.2. Preprocessing and feature extraction

To account for varying loudness levels between songs, we first perform loudness matching using *ReplayGain* [7] with a low reference level, so most songs are reduced in volume. In the few cases where amplification is required, we prevent signal distortions by applying *look ahead limiting*.

The feature sets introduced in Section 3.4 were extracted on an absolute and a beat-aligned timescale. Each feature coefficient was normalised to have zero mean and unit standard deviation across the dataset to aid model training.

3.3. Methods and context windows

Because we assume cuts can be judged independently, our models classify onsets individually based on audio excerpts with a length of t_{win} seconds, which are centered on the respective onset. To investigate the importance of long temporal dependencies, we use 20, 10, 4, 2, and 1 seconds for t_{win}.

We also consider different time resolutions to investigate the trade-off between feature dimensionality and the temporal precision required for classification. For the highest resolution, extraction is performed with frames of 46 ms duration and 50% overlap, resulting in a hop size (resolution) of 23 ms. Lower temporal resolutions $t_{\text{res}} \in \{46, 92, 184\}$ are obtained by repeated average pooling across time with a factor of two.

In addition to features aligned with respect to absolute time, we consider a *beat-aligned* feature covering a varying number of beats k_{win} with k_{res} frames per beat, since we hypothesise that the resulting invariance to musical tempo changes could help with cut detection. The beat-aligned features are computed by averaging the features of highest resolution based on absolute time to yield $k_{\text{res}} \in \{8, 4, 2\}$ non-overlapping frames per beat.

3.4. Feature sets

We used three different types of features. The *handcrafted feature set* is a compact, 36-dimensional representation designed to reduce overfitting risk and training time, and covers timbral, melodic, and metric aspects by combining three features with 12 feature coefficients each. Timbral quality is described by *Mel-frequency cepstral coefficients* (MFCCs), including the 0-th coefficient. *Chroma features* are extracted *MIR Toolbox* [8] to describe harmonic and melodic relationships, to complement the mostly pitch-invariant MFCCs. Finally, the *tempogram* [9] with a window of 4 seconds, describes rhythmic periodicities on multiple metrical levels.

To enable feature learning, we also employ two spectrogram representations as feature input. For our second feature set we use the Gammatone (GT) filterbank [10], because its logarithmic frequency scale reflects human perception more accurately than other representations such as FFTs. We use 75 filters with center frequencies from 60 Hz to 15.8 Khz.

The third feature set is motivated by the fact that fundamental frequencies of notes are hard to resolve with heavily overlapping, broad GT filters, although melodic and harmonic relationships may be needed for classification. Thus we compute the constant-Q transform with 12 bins per octave and 8 octaves spanning frequencies between 55 and 14.08 KHz.

3.5. Classification

For classification, we divide our problem into two binary decision problems: distinguishing entries from negative examples and exits, and distinguishing exits from negative examples and entries. Separate models for each problem were optimised independently during the experiments from section 4 to account for the potentially different decision making required for optimal performance. We minimise the cross entropy between the output and the ground truth probabilities.

We use neural networks in varying configurations with *rectified linear units (ReLU)* [11]. In initial experiments,

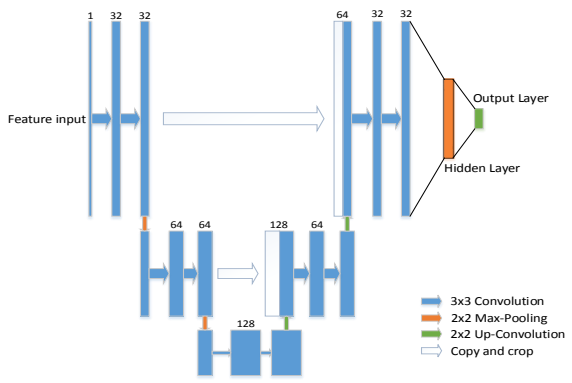

Fig. 1. The adapted U-Net architecture

we employed feed-forward neural networks of varying sizes, but performance was poor due to overfitting. Therefore, we turned to convolutional models.

Hypothetically, relevant features are local and stationary in time, meaning a local feature detector should be applied across the whole time dimension. After downsampling the resulting feature responses, we can repeat this process to form more complex and abstract features covering longer timespans. Thus we use a neural network with one-dimensional convolutions (1D-CNN). Together with *Max-pooling* without overlap by a factor of two and *Dropout* [12], this reduces the risk of overfitting. We use a two alternating convolution and pooling layers followed by one additional convolution and finally a fully connected layer with a softmax function.

We also employ a CNN with two-dimensional convolution (2D-CNN) that achieved great successes in image recognition tasks [13] to test whether relevant features for our task are not only stationary in time but also across the frequency range. The architecture is equivalent to the 1D-CNN, except that the convolutional and pooling filters are two-dimensional, operating across the time and frequency axis. We do not use the handcrafted feature set as input for this model as the filters should not be the same across all three features.

Max-pooling the features, especially along the time dimension, could be detrimental, as a certain temporal precision may be required for cut detection. We propose an extended architecture based on the *U-net* [14], a system for biomedical image segmentation. They avoid the problematic trade-off between localisation accuracy and the amount of context in traditional CNNs by augmenting them with an "upscaling" pathway, where the small, high-level feature maps are upscaled and combined with the lower-level feature maps to integrate detailed, local with abstract, global features.

We make two changes to the U-net model to adapt it to our problem, resulting in the model shown in Figure 1.

Firstly, the desired output in our case is a class label and not a stack of two-dimensional feature maps. Therefore we feed the activations from the final set of feature maps into a fully connected layer with Dropout, followed by softmax classification. Secondly, the convolutions in the model cause the final feature maps to be reduced to a centre piece of the input, because not enough context is available to compute the high-level features near the corners. Particularly local low-frequency information at the input "border" can however be important for our classification problem. As a solution, we apply zero-padding to the input so that the final feature map has the dimensions of the original input.

4. EXPERIMENTAL SETUPS AND RESULTS

We used 60%, 20% and 20% of all samples for the training, validation and test set respectively. All samples from the same song were assigned to the same subset to ensure the model has to generalise to unknown songs. The models were developed using *Theano* [15] and an NVIDIA GeForce Titan X GPU.

Training is performed by stochastic gradient descent using *Nesterov-Momentum* [16] with a weight of 0.9, mini-batches of size 200, a learning rate of 0.01, and L2 regularisation. To prevent overfitting, we stop training in case the loss on the validation set does not decrease by at least 0.5% for 5 epochs.

Our dataset contains more negative than positive examples, as onsets annotated as only entries are also counted as negative examples for the task of detecting exits and vice versa. Therefore we use the *balanced classification rate* (BCR) as evaluation metric, which takes the average of the accuracies for positive and for negative samples, weighted by their occurrence [17].

The mean BCR on the test set for each configuration will be denoted in parentheses for brevity. Configurations will be analysed by keeping all parameters but one fixed and then performing a *paired Wilcoxon* test to measure if the performance changes when only varying one parameter.

4.1. Experiment 1 - 1D-CNN

Because hyperparameter optimisation is prohibitively slow, we constrain some hyperparameters: As we found 20, 2 and 1 seconds of audio context t_{win} and temporal resolutions t_{res} over 46 ms to perform worse on average in initial experiments, we explore 4 and 10 seconds for t_{win} and 23 and 46 ms for t_{res}. We use a regularisation strength of 0.01, along with a hidden layer with $N^1_{\text{hidden}} = 200$ neurons. Furthermore, we use a pooling shape $P^2_{\text{shape}} = (2, 1)$ for the second pooling layer to reduce model complexity. We also use the same filter shape C_{shape} for all three convolutional layers, but vary the number of filters C^i in each layer i.

The results show that convolving spectral filters across time is beneficial, as the 1D-CNN model performs better than the fully connected models with a mean BCR of 0.610 over all configurations. On average, we found no significant difference between using 10 or 4 seconds as context window size t_{win}. However, on average a temporal resolution of $t_{\text{res}} = 23$ ms achieved slightly higher performances (mean BCR of 0.617) compared to $t_{\text{res}} = 46$ ms (0.605), suggesting cuts could be distinguished by small timing differences. The GT features performed better (0.643) than the CQT features (0.600) by a large margin, and both superseded the handcrafted combination (0.587) perhaps due to their higher di-

Parameter	List of settings
Feature set	Handcrafted, GT, CQT
t_{win}	$10, 4$
t_{res}	$23, 46$
(C^1, C^2, C^3)	$(32, 64, 128)$; $(64, 128, 256)$
C_{shape}	$(2, F)$; $(4, F)$
P^1_{shape}	$(1, 1)$; $(2, 1)$

Table 1. Parameters varied in experiment 1. F is the number of feature dimensions for each time point in the input feature. Filter shapes are denoted as tuples of their size in the time and feature dimension, in that order.

mensionality. On the other hand, the number and the shape of filters did not make any significant difference.

4.2. Experiment 2 - 2D-CNN

Three different filter shapes are tested - a small filter, a "tall" filter that captures more context along the frequency dimension, and a "wide" filter that captures more context along the time dimension, to investigate the extent of locality of the relevant features along the two dimensions, similar to [18]. We introduce a fixed pooling factor of 2 in the frequency dimension in both pooling layers, while retaining the option between not pooling along the time dimension and pooling with a factor of two as from experiment 1 in section 4.1 to keep the hyper-parameter space small.

Because the GT features performed best in experiment 1, we also include the beat-aligned GT features in this experiment to test whether introducing tempo invariance brings further improvement. The average BCR in this experiment was 0.656 and thus even better than in experiment 1 from section 4.1. Perhaps 1D filters detecting the same pattern at different frequencies can be replaced by one 2D filter, reducing the number of parameters without loss of expressivity.

For absolute-time features, varying the amount of context t_{win} between 10 and 4 seconds as well as the temporal resolution t_{res} between 23 and 46 ms led to small differences.

For the beat-aligned GT features, including $k_{\text{win}} = 12$ beats as context leads to slightly better performance (0.663) than only including 6 beats (0.650). While the performance using $k_{\text{res}} = 8$ frames (0.663) or $k_{\text{res}} = 4$ frames (0.660) per beat did not significantly differ, in both cases it was better than only using 2 frames (0.648).

We will compare the GT features based on absolute and musical time despite their different parameters, by comparing the network configurations that led to the best performance for each feature. Applying the pair-wise Wilcoxon test shows that on average, GT features aligned to beats (0.678) slightly outperform those based on absolute time (0.670). However, this effect is small, suggesting that the model builds time-invariant features itself, or that cut quality depends less on rhythmical aspects than expected.

The CQT feature performs slightly worse (0.653) than the GT feature (0.658), but this effect is much weaker than in the previous experiment in section 4.1.

Parameter	List of settings
Feature set	GT, Beat-aligned GT, CQT
t_{win}	$10, 4$
t_{res}	$23, 46$
k_{win}	$12, 6$
k_{res}	$8, 4, 2$
(C^1, C^2, C^3)	$(32, 64, 128)$; $(64, 128, 256)$
C_{shape}	$(3, 3)$; $(3, 9)$; $(9, 3)$
P^1_{shape}	$(1, 2)$; $(2, 2)$

Table 2. Parameters varied in experiment 2

Varying the number of filters does not change performance, while the smallest filter shape $(3, 3)$ performed slightly better than the wide and the tall filter. Finally, a pooling with shape $(1, 2)$ performs worse (0.65) than the shape $(2, 2)$ (0.66), possibly due to the reduction in parameters.

4.3. Experiment 3 - Adapted U-Net

Because the large size of the adapted U-net leads to long training times, we use the configuration shown in Figure 1 and only vary the context window $t_{\text{win}} \in \{10, 4\}$ and temporal resolution $t_{\text{res}} \in \{23, 46\}$ of the GT and CQT features. Furthermore, we use the beat-aligned GT features covering $k_{\text{win}} \in \{12, 6\}$ beats with $k_{\text{res}} \in \{8, 4, 2\}$ per frame.

Despite the theoretical appeal of the adapted U-Net, it did not provide better accuracies than the 2D-CNN model from Section 4.2 with an average BCR of 0.637. Potential benefits of this model in the form of more easily achieved high temporal precision might be counteracted by the higher model capacity which increases the risk of overfitting to our relatively small dataset. Perhaps most importantly, as we did not perform any hyper-parameter optimisation, other configurations could have outperformed the 2D-CNN, but we were unable to test this due to time constraints.

4.4. Best performing model

The best performances were obtained by 2D-CNNs from experiment 2 in section 4.2. Over a ten-fold cross-validation, we achieve a mean BCR of 0.654 with a standard deviation of 0.011 on the exit task, and a mean BCR of 0.692 with a standard deviation of 0.030 on the entry task.

5. MODEL ANALYSIS

In this section, we will investigate what our best-performing model learnt about the cut detection problem. Since it takes beat-aligned GT features as input, whose 75 frequency bins have centre frequencies ranging from 60 Hz to 15.8 KHz, as input, we can determine the performance after removing the lower or higher ends of the frequency range. We find that keeping larger frequency ranges generally lead to better performance, and that the middle frequencies are slightly more important for cut detection than high or low frequencies.

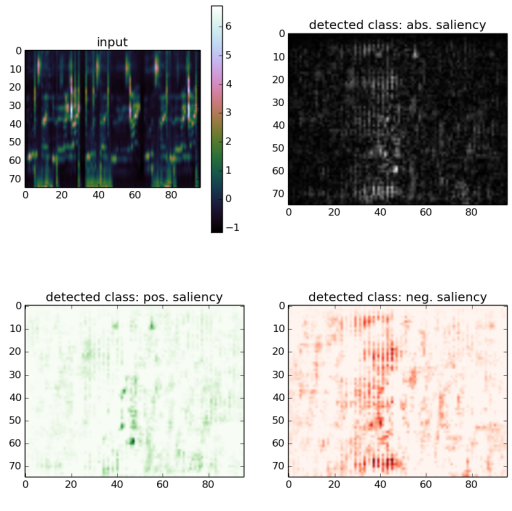

Fig. 2. Onset correctly identified as not being an entry.

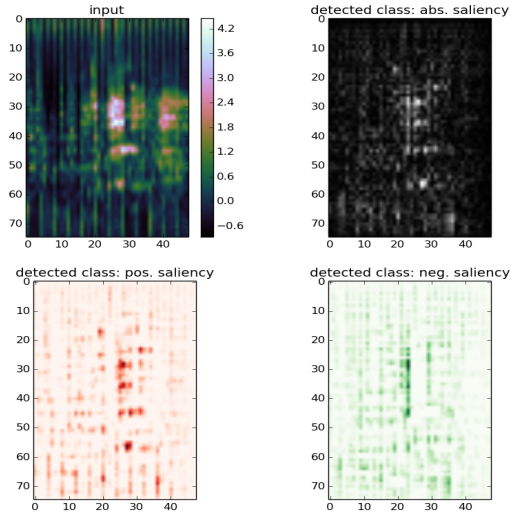

Fig. 3. Onset incorrectly identified as exit.

To better understand what features our model uses for cut-detection, we can select examples from the dataset and compute the gradient of the network's output with respect to the input. Visualising the gradient in the form of a *saliency map* reveals the influence of each part of the input on the classification outcome. To compute the saliency maps, we apply *guided backpropagation* [19] using the implementation by Schlueter [20]. For a given example $\mathbf{x}$, we take the network's class prediction $f(\mathbf{x})$ and compute $\frac{\partial f}{\partial x_i}$ for all input dimensions x_i. We generate a positive, negative, and absolute saliency map visualising the values $\max\{\frac{\partial f}{\partial x_i}, 0\}$, $-\min\{\frac{\partial f}{\partial x_i}, 0\}$ and $|\frac{\partial f}{\partial x_i}|$, respectively. We will show and interpret the results for a selection of examples below, in which the beat-aligned GT features along with the best-performing model is used, and provide conclusions from observing a larger number of examples.

In the true negative example shown in Figure 2, the music vocals present at the onset position at time frame 45 and frequency bin 60 are used by the model to determine the negative class, indicated by the high positive saliency at this point. The negative saliency map shows that with more acoustic presence before the onset, it could have misidentified as an entry. In general, the model for entries learnt that the amount of acoustical activity tends to be low before and high after an entry cut, whereas the exit model learnt the opposite – a rule also mentioned and used by the annotator.

However, Figure 3 shows this rule is followed too strictly: Although within a gap of the vocals, exiting here would interrupt an ongoing lyrical line mid-sentence. This suggests the exit model has an insufficient understanding of the musical phrasing as well as lyrical content that is required to deal with these more complicated cases. Judging by the regular vertical "stripes" in the positive and negative saliency map, rhythmical structure appears to be captured by the model.

6. ANALYSIS OF INCORRECT PREDICTIONS

To investigate error sources of the model, we let the annotator evaluate its predictions. We focused on analysing false positive errors, as transition quality is more important than having many possible transitions. We hypothesised that many false positive predictions might be true positives, as annotators tend to annotate only a portion of all suitable entries and exits, because only a limited number are needed to provide enough transitions for music rearrangement.

For the study, we generated predictions from the best model in section 4.4 for five randomly selected songs from the test set. From all false positive predictions, a random selection of 65 onsets was given to the annotator to decide whether and why the annotation is still considered correct or not, with "yes", "no", and "maybe" as possible answers.

Overall, our hypothesis is supported by the fact that 22 (33.8%) onsets were accepted as true positive predictions, and the annotator was undecided in 5 cases (7.7%). These inaccuracies in annotation imposes a limit on the achievable performance of any classifier trained on this dataset. Analysis of the other false positives however reveals that many are not suitable since they disrupt vocal phrases, indicating the model has not learnt to detect vocals reliably.

7. CONCLUSIONS

In this paper, we applied neural networks to the problem of finding cut points in music for the purpose of music rearrangement. While convolutional filters improve performance, adding an upscaling pathway to integrate information at different levels of abstraction did not increase accuracy further.

We found a GT spectrogram of 4 to 10 seconds with 23 to 46 ms of resolution to perform best, which outperforms the CQT and demonstrates the importance of high temporal resolution. Visualisations of our models highlight the relevance of instrument activity as a factor. Furthermore, the model learns

that acoustic energy tends to be low before and high after an entry point, and the opposite at exit points.

Because transition quality is inherently subjective and hard to define, we found a large number of missing positive labels in the dataset, and suggest measuring the inter-annotator agreement in future work.

As previous approaches [3, 5] focus on transition quality as an interaction between entry and exit, they could be integrated into our approach to provide a way of choosing the best transitions from the detected entries and exits.

8. REFERENCES

[1] J. R. Parker and B. Behm, "Creating audio textures by example: tiling and stitching," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. IEEE, 2004, vol. 4, pp. iv–317.

[2] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," in *ACM Transactions on graphics (TOG)*, 2007, vol. 26, p. 10.

[3] S. Wenner, "Music retargeting and synthesis," M.S. thesis, Swiss Federal Institute of Technology Zurich, 2012.

[4] S. Wenger and M. Magnor, "Constrained example-based audio synthesis," in *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2011, p. 6.

[5] S. Wenger and M. Magnor, "A genetic algorithm for audio retargeting," in *Proceedings of ACM Multimedia (ACMMM)*, 2012, pp. 705–708.

[6] J. Tauscher, S. Wenger, and M. Magnor, "Audio resynthesis on the dancefloor: A music structural approach," in *Proceedings of Vision, Modeling and Visualization (VMV)*, 2013, p. 8.

[7] David Robinson, "Replay gain - a proposed standard," `http://wiki.hydrogenaud.io/index.php?title=ReplayGain_1.0_specification`, 2001, Accessed: 2017-04-21.

[8] O. Lartillot, P. Toiviainen, and T. Eerola, "A matlab toolbox for music information retrieval," in *Data Analysis, Machine Learning and Applications*, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and Reinhold Decker, Eds., Studies in Classification, Data Analysis, and Knowledge Organization, pp. 261–268. Springer Berlin Heidelberg, 2008.

[9] P. Grosche and M. Müller, "Extracting predominant local pulse information from music recordings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1688–1701, 2011.

[10] RD Patterson, Ian Nimmo-Smith, John Holdsworth, and Peter Rice, "An efficient auditory filterbank based on the gammatone function," in *a meeting of the IOC Speech Group on Auditory Modelling at RSRE*, 1987, vol. 2.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[12] G. E. Hinton, N. Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[15] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[16] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, 1983, vol. 27, pp. 372–376.

[17] Gael De Lannoy, Damien François, Jean Delbeke, and Michel Verleysen, "Weighted svms and feature relevance assessment in supervised heart beat classification," in *International Joint Conference on Biomedical Engineering Systems and Technologies*. Springer, 2010, pp. 212–223, balanced classification rate.

[18] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," *arXiv preprint arXiv:1703.06697*, 2017.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *arXiv:1412.6806, also appeared at ICLR 2015 Workshop Track*, 2015.

[20] J. Schlüter, "Saliency maps and guided backpropagation," `https://github.com/Lasagne/Recipes/blob/master/examples/Saliency%20Maps%20and%20Guided%20Backpropagation.ipynb`, 2015, Accessed: 2017-04-21.