# A LOW-LATENCY, REAL-TIME-CAPABLE SINGING VOICE DETECTION METHOD WITH LSTM RECURRENT NEURAL NETWORKS

*Bernhard Lehner, Gerhard Widmer, Sebastian Böck*

Department of Computational Perception
Johannes Kepler University of Linz, Austria

## ABSTRACT

Singing voice detection aims at identifying the regions in a music recording where at least one person sings. This is a challenging problem that cannot be solved without analysing the temporal evolution of the signal. Current state-of-the-art methods combine timbral with temporal characteristics, by summarising various feature values over time, e.g. by computing their variance. This leads to more contextual information, but also to increased latency, which is problematic if our goal is on-line, real-time singing voice detection.

To overcome this problem and reduce the necessity to include context in the features themselves, we introduce a method that uses Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN). In experiments on several data sets, the resulting singing voice detector outperforms the state-of-the-art baselines in terms of accuracy, while at the same time drastically reducing latency and increasing the time resolution of the detector.

***Index Terms***— singing voice detection, music information retrieval, recurrent neural nets

## 1. INTRODUCTION

At a first glance, the problem of singing voice detection (SVD) in polyphonic music recordings seems to be analogous to voice activity detection (VAD). However, there are several reasons why they are quite different. Often, VAD algorithms utilise features relating to energy, zero crossing rate, or periodicity, but the influence of musical accompaniment renders those features less useful. In [1] it was shown that music is the most challenging disturbance for several state-of-the-art VAD algorithms. On the other hand, in [2] it was shown that the presence of singing voice did not produce a high false positive rate for their VAD system, which indicates that a VAD system is not necessarily suited for SVD.

In the field of Music Information Retrieval (MIR), SVD has attracted increasing attention lately, usually as a preprocessing step to improve, e.g. Artist Recognition or Singing Voice Separation. Often, audio frames of approx. 200 ms length are used, mainly for two reasons: 1) annotating a finer grained ground truth is a tedious task for humans; and 2) sometimes a relatively long observation window is necessary in order to yield an appropriate frequency resolution in the spectrum for further computations. Therefore, SVD becomes more challenging with decreasing frame lengths. A higher resolution could open up new possibilities, and be useful for tasks like keyword spotting in music, automatic karaoke text alignment, and various other real-time synchronisation tasks.

In order to achieve a higher resolution in time, we take the method from Lehner et al. [3] as a starting point. By moving some complexity from the feature extraction to the classification stage, we manage to increase resolution by a factor of 10, to 50 predictions per second. The time that needs to pass before the features can be fed to a classifier, which one might call *latency-by-design*, is reduced from 1500 ms to 140 ms (see Section 3.1.1 below), compared to this baseline.

## 2. RELATED WORK

Our starting point and baseline is the method by Lehner et al. [3], which gave promising results on several publicly available corpora. It is designed to be light-weight, and suited to be combined with standard classifiers like Random Forests. In order to allow such classifiers to also consider temporal characteristics of the signal, feature values are summarised over time, by calculating sums or variances over several frames. The units of audio to be classified are 200 ms frames, resulting in five classifications per second. The latency-by-design, caused mainly by the Vocal Variance (VV) feature [3], is rather high. The VV is simply the variance of the first five MFCCs, calculated over 11 consecutive values around the current frame. Therefore, for the computation of the VV for the current frame, six MFCCs need to be available in the forward direction.

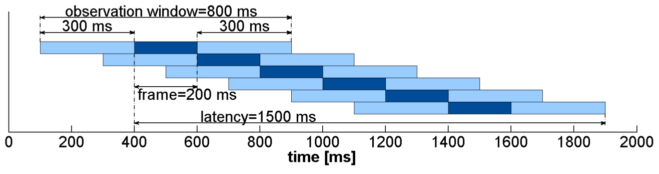

**Fig. 1**. Latency in the method by Lehner et al. [3].

MFCCs are computed over a rather long observation window of 800 ms, placed symmetrically around the current classification frame (see Fig. 1).This windowing scheme results in a latency-by-design of 1500 ms (1200 ms for six frames, and the additional 300 ms in the forward direction). To alleviate this problem, we will move some complexity from the feature extraction to the classification stage by using a context-aware classifier, as described in Section 3.

All in all, the feature vector used in [3] comprises 116 attributes, computed every 200 ms: 17 Fluctogram variance features, 17 Spectral Flatness means and 17 Spectral Contraction variances, 60 MFCC features (30 MFCCs plus corresponding deltas), and 5 Vocal Variance features. For postprocessing, simple median filters are used to first smooth the continuous output of the classifier (order=4), and then to realise majority voting (order=5). A detailed overview and comparison to other methods is given in [3], and would be beyond the scope of this paper.

Another state-of-the-art baseline that we re-implemented for comparison is the method by Weninger et al. [4]. As features, they extract a 46 attribute vector with the open-source toolkit openSMILE [5]: 13 MFCCs along with their first and second order derivatives (delta and double delta), short-time energy, zero- and mean-crossing rate, voicing probability, F-zero, harmonics-to-noise ratio (HNR), and predominant pitch.

In our implementation of this reference method, we do not extract the features in a beat-wise manner, but with a fixed observation window of 40 ms and 50% overlap, which gives the same time resolution as in our method to be described in this paper. To focus on the performance of their features, we also skip the combined NMF-based source separation. This is justified because the benefit regarding singing voice detection – according to their results – is rather small, with an increase of approx. 1 percentage point (ppt) in accuracy. For other tasks like gender recognition, their preprocessing appears more helpful, with approx. 3-4 ppt better accuracy reported. As classifier, Weninger et al. [4] use *Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNNs)*, which have access to the complete past and future context. Therefore, their method is not online-capable.

## 3. METHOD

Recurrent neural networks (RNNs) are used for a wide range of tasks beyond simple frame classification. Especially if used with Long Short Term Memory (LSTM) [6] units in a bidirectional topology, they have exhibited state-of-the-art performance in a lot of tasks where the temporal context of a signal is important to classify individual frames, e.g. in phoneme classification [7] or beat tracking [8]. Temporal context is sometimes also necessary for humans to make a *vocal-nonvocal* decision [1], therefore using LSTM-RNNs seems to be an appropriate choice for SVD. Furthermore, they are capable of learning the amount of context needed for classifying the current frame, which is an advantage over fixed-size context approaches with e.g. graphical models.

In our method we use a uni-directional RNN with one hidden layer and 55 LSTM units to classify frames into *vocal* and *non-vocal*. The hidden units are not only connected to the input units (or in the case of consecutive hidden layers, to the units of the preceding hidden layer), but each unit also has a connection to itself, i.e. the previous time step. Through these recurrent connections the RNN has access to past information, which enables it to model temporal context.

Compared to standard *tanh* units as generally used in NNs, LSTM units have a unique memory cell that enables them to store information for an arbitrary time span. Read, write and delete operations on this memory cell are handled through gates, which act similarly to normal units. RNNs equipped with LSTM units do not suffer from the vanishing gradient problem and expand the temporal context handled by RNNs from a few time steps to as much as hundreds of steps. By exploiting the ability of LSTM-RNNs to model a wide temporal context, we will be able to solve one of the major drawbacks of [3] – the latency-by-design of 1500 ms – by removing the need to compute long-term functionals like mean or variance of features, and by computing the basic underlying MFCC features from shorter audio frames of 100 ms length.

Since we use a uni-directional topology, our method is online capable, i.e. it can classify a signal almost instantaneously. The maximum latency introduced by our method is 140 ms (see Fig. 2).

### 3.1. Feature Set

Before the feature extraction, the songs are unified, i.e. downsampled to 22 kHz and converted to mono. No amplitude normalisation is done, since this would destroy the online-capability. The units of audio for classification are frames of 20 ms, resulting in 50 classifications per second of audio. The actual window over which all features are calculated is 100 ms, and is always placed symmetrically around the current frame. Therefore, we need an additional 40 ms in both directions of the current frame.

#### *3.1.1. MFCCs and deltas*

We use the VOICEBOX toolbox [9] to extract 30-dimensional MFCCs including the 0th coefficient. Their first order derivatives (deltas) are also computed, which results in a 60-dimensional feature vector. In VOICEBOX [9], the deltas are computed as the slope of 9 consecutive coefficients, placed symmetrically around the current frame. Therefore, five MFCCs (current frame+four frames afterwards) need to be available before the deltas for the current frame can be computed.

[1]examples at `http://www.cp.jku.at/misc/eusipco2015/`

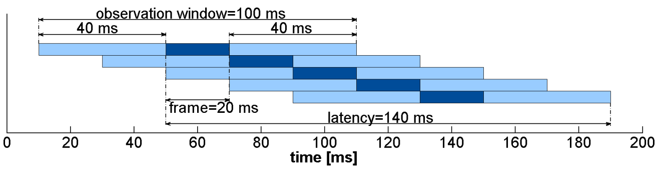

**Fig. 2**. Latency in our new method.

According to the windowing scheme explained in 3.1 above, this results in a latency-by-design of 140 ms (100 ms for five frames, and the additional 40 ms in the forward direction).

In order to distinguish highly harmonic instruments from singing voice, [3] suggest an additional set of temporal characteristic descriptors of the signal, which is explained in the next section.

### 3.1.2. *Fluctogram*

To detect sub-semitone fluctuations of partials without the necessity of pitch estimation, in [3] the *Fluctogram* was introduced. The basic idea behind this feature is to use the *cross correlation* to compare each band-divided spectrum of a time frame $X_t$ to the subsequent one $X_{t+1}$, and the index of the maximum correlation when $X_{t+1}$ is shifted $\pm n$ bins, is calculated. This way, only sub-semitone, pitch-continuous fluctuations are targeted and detected, as can be seen in Fig. 3.

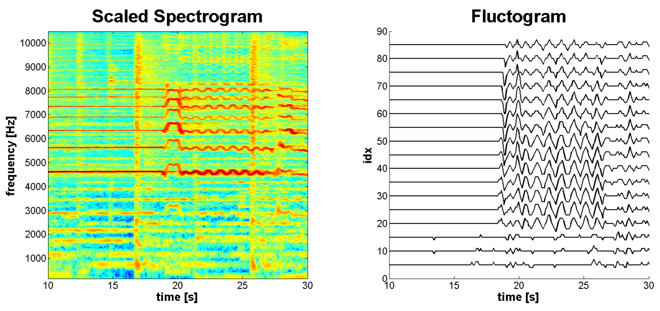

**Fig. 3**. A Spectrogram and the corresponding Fluctogram.

In [3], each frame of audio is characterised by calculating the variance over a window of 40 successive Fluctogram values, centered on the current frame, separately for each band. For the LSTM-RNN, we consider it sufficient to just use the plain Fluctogram values, which results in 17 values per audio frame. This is also true for the reliability indicators (see below) that we use along with the Fluctogram, where we only use the interim values, instead of summarising them with means or variances.

### 3.1.3. *Spectral Flatness and Spectral Contraction*

The Fluctogram is error-prone under certain circumstances, therefore additional information that relates to the reliability of its values in the individual bands is necessary. The Fluctogram is most reliable when the signal is not noise-like, which is characterised for each frequency band by the *Spectral Flatness (SF)* measure [10], yielding another 17 feature values.

The trajectory of the partial that dominates the result of the cross-correlation, can sometimes cross the frequency band boundary. To account for that, the *Spectral Contraction (SC)* [3] relates the energy of the spectrum in the center to the total amount of energy. It is also computed for each frequency band, yielding another 17 feature values. Conveniently, the results of both reliability indicators are in the range $[0 : 1]$, and input level invariant.

### 3.1.4. *Final Featureset and Normalisation*

As input for the LSTM-RNNs we use 30 MFCCs and their first derivatives (delta coefficients), 17 Fluctogram shift indices, 17 Spectral Contraction, and 17 Spectral Flatness values. All of the resulting 111 attributes can be calculated from the same spectrogram with an observation window of 100 ms, centered around a 20 ms frame. According to Graves [11], it is recommended to apply normalisation (mean=0, std=1) to the features. This is done by solely using the training set to compute mean and standard deviation. The test set is always left unseen, and normalised according to the training set.

## 3.2. LSTM-RNN Training and Testing

The LSTM-RNNs have one input layer matching the size of the feature vector (111), one hidden layer with 55 LSTM units, and one output layer with two units. The weights are randomly initialised from a zero-mean Gaussian distribution ($\sigma$=0.05). A similar distribution ($\sigma$=0.3) is used to add noise to the input activations for improved generalisation.

Each song from the training set is presented in correct order on a frame-by-frame basis, and the weights are updated with a steepest descent optimiser (rate=$10^{-5}$, momentum=0.9). Furthermore, we apply the following strategy to improve the generalisation of the proposed method: We randomly split the training set into 80%-20% subsets, and use these as training and validation sets, where the validation part of the data is used for early stopping after no improvement over 20 epochs. We repeat this $N$ times. This will result in several models, trained with different input, which are then combined by averaging their continuous outputs. In order to get meaningful results for comparison, we do this also for the baseline method of Weninger et al. [4].

## 4. RESULTS

In this section, we present the results on three different corpora, two of which are publicly available. We train, according to our previously determined strategy, several LSTM-RNNs and compare the results to the previously introduced methods in section 2. The topology of the networks and all additional parameters remain unchanged throughout all experiments, hence solely the training and test data is different.

### 4.1. In-House data set

To find an optimal setting of classifier parameters, we first perform a set of experiments with an in-house data set of 149 annotated songs by 149 different artists. Although all songs are from the same genre (rock), we consider this data set suitable for parametrising the classifier: the songs contain a lot of guitar soli, which have characteristics similar to vocals. We expect any classifier that achieves a low false positive rate for singing voice on this set, to also have a low false positive rate in general. Approximately 52% of the frames are annotated as vocal, and the amount of pure singing, i.e. without instrumental accompaniment, is negligible.

This data set is split into a 75 song training set, and a 74 song test set, approx. 5 h each. Only the 75 song train set was used to optimise parameters, and the test set is always left unseen. For this, the train set is further split 5-fold into 60 train- and 15 validation songs, respectively. Therefore, 5 LSTM-RNNs are trained. The previously unseen 74 song test set is normalised according to the train set, and the outputs of all models are averaged.

To assess the impact of adaptive context learning, we additionally train a second set of RNNs, which use *tanh* units instead of *lstm* units. We changed the hidden layer to comprise 142 *tanh* units in order to have the same amount of weights. The learning rate was reduced to $10^{-6}$ in order to obtain network convergence. The remaining parameters (incl. the train- and test setup) of the RNNs are equal to those of the LSTM-RNNs. The results on the 74 song test set are presented in Table 1.

Clearly, the LSTM-RNN with adaptive context learning (column NEW) outperforms the RNN with *tanh* units (col. RNNtanh), with a difference in accuracy of almost 6 ppt (89.1% vs. 83.2%). Compared to the method of Lehner et al. [3] (col. LEH), the accuracy increased considerably by more than 5 ppt, at the same time giving an increased time resolution, decreased latency and size of feature vector, and without any postprocessing involved. Compared to the method of Weninger et al. [4], the accuracy is only slightly better (89.1% vs. 87.8%). Considering the advantage of the bi-directional BLSTM-RNNs (access to complete future context) over our LSTM-RNN (online-capable), it appears that the set of features utilised in our method is more adequate. On the other hand, the feature vector of Weninger et al. [4] is much smaller (46 vs. 111 attributes).

### 4.2. Experiments on Common Benchmark data sets

In these experiments we use two publicly available corpora along with singing voice annotations: **Jamendo Corpus:** 93 copyright-free songs from the Jamendo music sharing website [12], collected and annotated by Ramona et al. [13]. **RWC Music Database: Popular Music (RWC-MDB-P-2001):** 100 songs released by Goto et al. [14], with annotations provided by Mauch et al. [15].

| | LEH | WEN | RNNtanh | NEW |
|---|---|---|---|---|
| acc [%] | 83.62 | 87.81 | 83.20 | 89.06 |
| recall | .848 | .884 | .873 | .905 |
| precision | .818 | .881 | .816 | .887 |
| f-measure | .833 | .883 | .843 | .895 |

**Table 1**. Results on internal data set. LEH: the method from Lehner et al. [3]. WEN: the method from Weninger et al. [4]. RNNtanh: proposed method w/o adaptive context learning. NEW: the proposed method with adaptive context learning. Recall, precision, and f-measure relate to our class of interest, *vocals*.

| | LEH | WEN | NEW |
|---|---|---|---|
| acc [%] | 88.17 | 86.20 | 89.42 |
| recall | .862 | .875 | .906 |
| precision | .880 | .869 | .898 |
| f-measure | .871 | .872 | .902 |

**Table 2**. Results on Jamendo Corpus.

#### 4.2.1. *Jamendo Corpus*

For this data set, the authors [13] suggest a specific split, with 61 training, 16 validation, and 16 test songs. We present the results on the Jamendo Corpus in Table 2. Similar to the results on the internal data set, our method gives promising results. Compared to the method of Weninger et al. [4], the accuracy is approx. 3 ppt higher (89.4% vs. 86.2%). Compared to the results of Lehner et al. [3], the accuracy is approx. 1 ppt higher.

#### 4.2.2. *RWC Music Database*

For this data set, in [15, 16], and [3], a 5-fold cross validation is utilised. In order to allow for a fair comparison, we used the exact same split as in [3]. To make sure that no future context was used, we treated every data split as a separate data set and normalised always only according to the songs that were used for training. Thus, we ended up with five different representations of the same songs that only differ in the normalisation scheme. We present the results on the RWC Music Database in Table 3. Compared to the method of Weninger et al. [4], the accuracy is approx. 2 ppt higher (92.3% vs. 90.0%). Compared to the results of Lehner et al. [3], the accuracy is almost 5 ppt higher (92.3% vs. 87.5%), and the biggest improvement appears to be regarding the precision.

| | LEH | WEN | NEW |
|---|---|---|---|
| acc [%] | 87.51 | 90.02 | 92.29 |
| recall | .926 | .915 | .934 |
| precision | .875 | .919 | .938 |
| f-measure | .900 | .917 | .936 |

**Table 3**. Results on RWC Music Database.

## 5. CONCLUSION

We suggest a method for singing voice detection that uses very light-weight features, along with LSTM-RNN. Due to the reduced latency-by-design of 140 ms, as well as the increased time resolution of one prediction every 20 ms, new possibilities open up, especially for real-time scenarios.

Consistently, our method outperformed two state-of-the-art baselines on three different data sets, two of which are publicly available. No adaptation of parameters was done after determining a suitable strategy with just the internal data set. The baseline from Lehner et al. [3] has the disadvantage of a latency-by-design of 1500 ms, and the baseline of Weninger et al. [4] is an offline method due to the need for BLSTM-RNNs to access future context.

## 6. ACKNOWLEDGMENTS

This research is supported by the Austrian Science Fund (FWF) under grants TRP307-N23 and Z159, and by the European Union Seventh Framework Programme FP7 / 2007-2013 through the GiantSteps project (grant agreement no. 610591).

## REFERENCES

[1] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, “Real-life Voice Activity Detection with LSTM Recurrent Neural Networks and an Application to Hollywood Movies,” in *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2013*. IEEE, 2013, pp. 483–487.

[2] R. Sonnleitner, B. Niedermayer, G. Widmer, and J. Schlüter, “A Simple And Effective Spectral Feature For Speech Detection In Mixed Audio Signals,” in *Proc. of the 15th Int. Conf. on Digital Audio Effects (DAFx’12)*, 2012.

[3] B. Lehner, G. Widmer, and R. Sonnleitner, “On the reduction of false positives in singing voice detection,” in *Proc. of the 2014 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2014*. IEEE, 2014, pp. 7530–7534.

[4] F. Weninger, M. Wöllmer, and B. Schuller, “Automatic assessment of singer traits in popular music: Gender, age, height and race,” in *Proc. of the 12th Int. Conf. on Music Information Retrieval (ISMIR 2011)*, 2011.

[5] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: the munich versatile and fast open-source audio feature extractor,” in *Proc. of the Int. Conf. on Multimedia*. ACM, 2010, pp. 1459–1462.

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *IEEE Trans. on Neural Networks*, vol. 18, pp. 602–610, 2005.

[8] S. Böck and M. Schedl, “Enhanced Beat Tracking with Context-Aware Neural Networks,” in *Proc. of the 14th Int. Conf. on Digital Audio Effects (DAFx-11)*, Paris, France, September 2011, pp. 135–139.

[9] M. Brookes, “Voicebox: Speech Processing Toolbox for Matlab,” Website, 1999.

[10] A. Gray Jr. and J. Markel, “A spectral-flatness measure for studying the autocorrelation method of linear prediction of speech analysis,” *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 22, no. 3, pp. 207–217, Jun 1974.

[11] A. Graves, *Supervised sequence labelling with recurrent neural networks*, vol. 385, Springer, 2012.

[12] P. Grard, L. Kratz, and S. Zimmer, “Jamendo, open your ears,” Website, 2005, Available online at `http://www.jamendo.com`; visited on June 1st, 2015.

[13] M. Ramona, G. Richard, and B. David, “Vocal detection in music with support vector machines,” in *Proc. of the 2008 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2008*. IEEE, 2008, pp. 1885–1888.

[14] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Popular, classical, and jazz music databases,” in *Proc. of the 3rd Int. Conf. on Music Information Retrieval (ISMIR 2002)*, 2002, vol. 2, pp. 287–288.

[15] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto, “Timbre and Melody Features for the Recognition of Vocal Activity and Instrumental Solos in Polyphonic Music,” in *Proc. of the 12th Int. Conf. on Music Information Retrieval (ISMIR 2011)*, 2011, pp. 233–238.

[16] B. Lehner, R. Sonnleitner, and G. Widmer, “Towards Light-weight, Real-time-capable Singing Voice Detection,” in *Proc. of the 14th Int. Conf. on Music Information Retrieval (ISMIR 2013)*, 2013.