ZERO-MEAN CONVOLUTIONS FOR LEVEL-INVARIANT SINGING VOICE DETECTION

Jan Schlüter
Austrian Research Institute for Artificial Intelligence, Vienna
jan.schlueter@ofai.at

Bernhard Lehner
Institute of Computational Perception, Johannes Kepler University Linz, Austria
bernhard.lehner@jku.at

ABSTRACT

State-of-the-art singing voice detectors are based on classifiers trained on annotated examples. As recently shown, such detectors have an important weakness: Since singing voice is correlated with sound level in training data, classifiers learn to become sensitive to input magnitude, and give different predictions for the same signal at different sound levels. Starting from a Convolutional Neural Network (CNN) trained on logarithmic-magnitude mel spectrogram excerpts, we eliminate this dependency by forcing each first-layer convolutional filter to be *zero-mean* – that is, to have its coefficients sum to zero. In contrast to four other methods – data augmentation, instance normalization, spectral delta features, and per-channel energy normalization (PCEN) – that we evaluated on a large-scale public dataset, zero-mean convolutions achieve perfect sound level invariance without any impact on prediction accuracy or computational requirements. We assume that zero-mean convolutions would be useful for other machine listening tasks requiring robustness to level changes.

1. INTRODUCTION

Automatically annotating the presence of singing voice in a music recording is a challenging task, as singing voice covers a wide range of notes and expressions, is often accompanied by several other instruments, and may be confused with instruments capable of producing similar melody contours. Recent approaches try to capture this variability by training strong classifiers such as deep neural networks on annotated data [9, 12, 14, 20, 22]. While they achieve high accuracies on standard benchmark datasets, classifiers may exploit correlations between inputs and targets that are present in both the training and test data, but are not semantically meaningful (such a classifier is sometimes called a *horse* [24]) or unwanted (leading to *algorithmic bias* [6]). In [13], we demonstrated that three state-of-the-art singing voice detectors – both with hand-designed and learned features – exploit a dependency between singing voice and sound level present in common datasets.

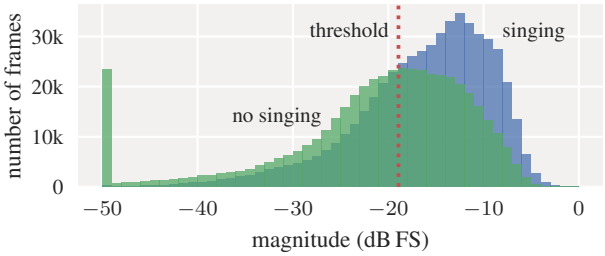

Figure 1: Spectrogram frames of the Jamendo training set containing singing voice tend to have larger magnitudes. A simple threshold allows distinguishing the classes with an accuracy of 61% (8.5 percent points above the baseline).

We can reveal this dependency in a simple experiment: We compute spectrograms for all files of the Jamendo dataset [18] and sum up the linear magnitudes for each frame. The distribution of magnitudes in the training set is clearly skewed towards larger values for frames containing singing voice (Figure 1). Choosing an optimal threshold, we can distinguish vocal from nonvocal frames at an accuracy of 61.1%. With the same threshold, we correctly classify 59.0% of the validation and 68.7% of the test set frames. This is a strong enough improvement over predicting the majority class (52.6%, 51.4% and 53.7%, respectively) that any classifier will pick up this cue. Note that for clarity of presentation, we omitted typical preprocessing steps such as mel scaling, logarithmic magnitude compression or bandwise standardization, but results hardly differ (0.3 percent points improved) with these steps included.

Of course this confound does not stem from inherent characteristics of singing voice, but from production habits in commercial music – if a track contains vocals, those are mixed to stand out. Thus, it affects many other Western-music datasets (we verified this for RWC [8, 16], MSD100 [17], and tracks containing vocals in MedleyDB [3]) that are commonly used for singing voice detection research.

In [13], we argue that to avoid this, datasets should include a sufficient number of instrumental tracks, which cannot feature vocals as the most prominent instrument. And indeed, for the enlarged dataset in [13], there is hardly any linear correlation between input magnitude and class (Figure 2). However, there is still a strong statistical dependency, with vocal frames exhibiting a different magnitude distribution from nonvocal frames, enabling a better-than-chance prediction of the class from the input magnitude.

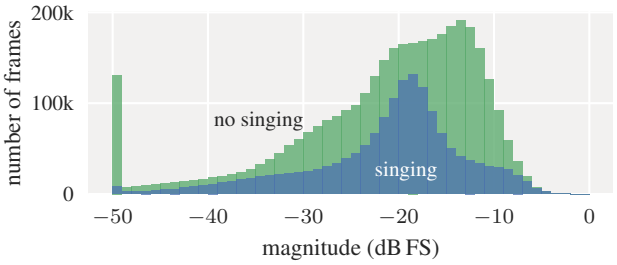

Figure 2: For a dataset including many purely-instrumental tracks, input magnitude and class are not linearly correlated, but still show a clear statistical dependency exploitable by a classifier.

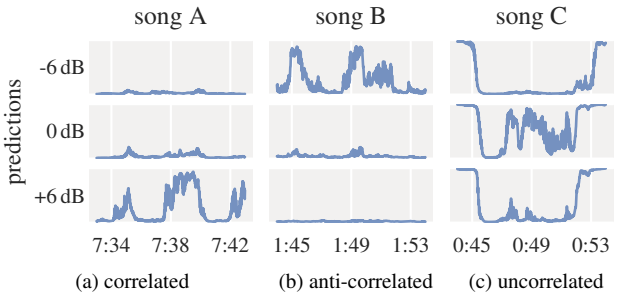

Figure 3: Presenting a state-of-the-art classifier with the same music excerpt at altered sound levels reveals a strong sound level dependency. (a) For some songs, increasing the level by 6 dB increases the classifier's output. (b) For some, this dependency is inverted. (c) For some, vocals are only detected at the original sound level (second row).

When training a state-of-the-art network on this dataset, it develops a complex sound level dependency: for some test files, predictions are correlated with input magnitude (Fig. 3a), for others, they behave conversely (Fig. 3b) or decrease for any deviation from the original level (Fig. 3c). If and which of these cases applies to a given input seems to depend on the content, not only the original sound level, and sometimes varies from model to model, but the effect appears reliably.

While a closer investigation of the underlying reasons would be highly interesting, for now we content ourselves with stating that this effect is unwanted. As changing the sound level of a music recording does not change the presence of singing voice, we would like a singing voice detector to be invariant to the scale of the input signal. In [13], we show how to achieve this for a system based on hand-designed features. In this work, we propose and evaluate different ways to achieve the same for a Convolutional Neural Network (CNN) trained on mel spectrograms, outperforming the hand-designed system.

The remaining paper is structured as follows: In the next section, we review related work on singing voice detection and level invariance. Section 3 explains the CNN-based baseline system as well as five methods to improve its robustness to level changes, and Section 4 evaluates these methods experimentally. Finally, Section 5 summarizes our findings and their implications.

2. RELATED WORK

From early approaches [2] to recent ones [9, 12, 14, 20, 22], singing voice detection has mostly been addressed with classifiers trained on audio features. Berenzweig et al. [2] based their system on an existing speech recognizer, combined with cepstral coefficients and classified with a simple Gaussian model. Leglaive et al. [12] trained a bidirectional Recurrent Neural Network (RNN) on preprocessed mel spectra, Lehner et al. [14] trained a unidirectional RNN on a set of hand-designed features. Schlüter et al. [22] define the current state of the art using a CNN on logarithmic-magnitude mel spectrograms trained with data augmentation; we will use their public implementation as a starting point. More recent work uses CNNs in attempts to lower annotation effort by learning from song-wise labels [20], or by deriving labels from pairing songs with instrumental versions [9]. The related tasks of auto-tagging (i.e., determining song-wise labels) and singing voice separation are also tackled with CNNs, but will not be considered here.

Apart from our work [13], to the best of our knowledge, invariance to the sound level has not been addressed in the context of singing voice detection, but at least Mauch et al. [15] and Sturm [24, Sec. III.B] recognized it as a possible confounding factor for music information retrieval systems. In speech recognition, early approaches based on Mel-Frequency Cepstral Coefficients (MFCCs) discard the 0^{th} coefficient [4, Eq. 1], effectively becoming invariant to the scale of the input signal. Modern CNN-based systems processing spectrograms or raw signals achieve robustness by using large networks and datasets (e.g., 38 million parameters and 7000 hours in [1]). For smaller CNNs, Wang et al. [26] recently proposed to process spectrograms with an automatic gain control of learnable parameters, termed per-channel energy normalization (PCEN). We will include this method in our experiments.

3. METHOD

In the following, we will describe the state-of-the-art method we used as a starting point, and five modifications aiming to reduce its sound level dependency (which was demonstrated in Figure 3).

3.1 Baseline

We base our work on the system of Schlüter et al. [22], in the variant they made available online [1] and described in [21, Sec. 9.8]. From monophonic input signals sampled at 22 kHz, it computes magnitude spectrograms (frame length 1024, hop size 315 samples), applies a mel filterbank (80 bands from 27.5 Hz to 8 kHz) and scales magnitudes as $\log(\max(10^{-7}, x))$. A CNN classifies 115-frame excerpts of these spectrograms into vocal/nonvocal. It starts with batch normalization [10] across the batch and time axis without learned scale and bias – this effectively standardizes each mel band over the training set as in [22], but can adapt to changes to the frontend during training,

[1] `https://github.com/f0k/ismir2015/tree/phd_extra`, accessed 2018-03-30

which we need for PCEN. This is followed by two convolutional layers of 64 and 32 3×3 filters, respectively, 3×3 max-pooling, 128 and 64 3×3 convolutions, 128 3×18 convolutions, 4×1 pooling, and three dense layers of 256, 64 and 1 units, respectively. Each convolutional and dense layer is followed by batch normalization and leaky rectification $\max(x/100, x)$ except for the final layer, which uses a sigmoid unit for binary classification.

During training, 50% dropout is applied before each fully-connected layer, and inputs are augmented with pitch shifting and time stretching up to $\pm30\%$, and random frequency band filters of up to ±10 dB, before mel scaling.

At test time, we turn the CNN into a fully-convolutional net, replacing dense layers by convolutions and adding dilations as described in [23]. This allows computing predictions over a full spectrogram without redundant computations that would occur when feeding overlapping 115-frame excerpts. All batch normalizations use statistics collected during training, not statistics from test examples.

3.2 Data Augmentation

A sure way to prevent classifiers from exploiting particular correlations in the training data is to remove these correlations from the data. Data augmentation attempts to remove or reduce correlations by varying the training examples along the confounding dimension. In our case, to reduce the dependency between input magnitude and target shown in Figures 1, 2, we scale input signals randomly by up to ±10 dB in addition to the existing augmentations.

3.3 Instance Normalization

As a more drastic measure, we replace the initial batch normalization with instance normalization [25], i.e., we separately standardize each 115-frame excerpt to zero mean and unit variance per mel band, both at training and at test time. This is in contrast to batch normalization, which uses batch-wise rather than excerpt-wise statistics during training, and fixed dataset-wise statistics [2] for testing.

Instance normalization trivially results in a representation that is fully invariant to scaling the input signal. However, it prevents using the CNN as a fully-convolutional net at test time, since every excerpt needs to be processed separately. In Section 4.4, we will see how this affects computation time.

3.4 Spectral Delta Features

Scaling the input signal results in a shift of the logarithmic-magnitude mel spectrogram. Delta features, i.e., the elementwise difference between a frame and its predecessor, are invariant to such an offset. They are commonly used as supporting features to include temporal information in frame-wise classification, but have also been used successfully as the only input for RNN-based musical onset detection (albeit in a rectified form, [5]) and might be sufficient for singing voice detection.

[2] For simplicity, an exponential moving average of batch-wise statistics collected during training, as suggested for validation in [10, Sec. 3.1]. Importantly, the normalization is independent of the input at test time.

3.5 PCEN

Proposed by Wang et al. [26], per-channel energy normalization (PCEN) processes a mel spectrogram of linear magnitudes (i.e., replacing the logarithmic scaling) as

$$Y_{t,f} = \left(\frac{X_{t,f}}{(\epsilon + M_{t,f})^{\alpha_f}} + \delta_f\right)^{r_f} - \delta_f^{r_f}, \qquad (1)$$

where $\boldsymbol{M}$ is an estimate of the local magnitude per time step and frequency band computed using a simple infinite impulse response (IIR) filter:

$$M_{t,f} = (1 - s_f)M_{t-1,f} + s_f X_{t,f} \qquad (2)$$

The division by $\boldsymbol{M}$ implements an automatic gain control, which is followed by root compression (for $0 < r_f < 1$). Wang et al. parameterize $\alpha_f := \exp(\hat{\alpha}_f)$, $\delta_f := \exp(\hat{\delta}_f)$, $r_f := \exp(\hat{r}_f)$ and learn $\hat{\boldsymbol{\alpha}}$, $\hat{\boldsymbol{\delta}}$, $\hat{\boldsymbol{r}}$ as part of a neural network. Learning the logarithms ensures that $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{r}$ remain positive. Instead of learning $\boldsymbol{s}$, Wang et al. replace $\boldsymbol{M}$ with a convex combination of precomputed IIR filters of different smoothing factors s and learn the combination weights.

We deviate from their approach in two respects:

1. We fix $\alpha_f := 1$, as any other choice will make $\boldsymbol{Y}$ dependent on the scale of $\boldsymbol{X}$.

2. We parameterize $s_f := \exp(\hat{s}_f)$ and learn $\hat{\boldsymbol{s}}$ directly as part of the neural network. [3] Wang et al. noted that option in [26, Sec. 3], but did not explore it.

The IIR filter must process the input sequentially, and thus is not a good fit for massively parallel computation devices such as Graphical Processing Units (GPUs). We will see how this affects computation time in Section 4.4.

3.6 Zero-Mean Convolution

Spectral delta features are just one of many ways to compute differences in the spectrogram that are invariant to adding a constant to the input. For example, we could just as well compute differences between neighbouring frequencies. More generally, any cross-correlation with a zero-mean filter $\boldsymbol{W}$ will remove a global offset c from $\boldsymbol{X}$:

$$\begin{aligned}((\boldsymbol{X} + c) * \boldsymbol{W})_{t,f} &= \sum_{i,j} (X_{t+i,f+j} + c)\, W_{i,j} \\ &= \sum_{i,j} X_{t+i,f+j} W_{i,j} + c \sum_{i,j} W_{i,j} = (\boldsymbol{X} * \boldsymbol{W})_{t,f}\end{aligned}$$

The last step uses our assumption of a zero-mean filter, $\sum_{i,j} W_{i,j} = 0$. The first convolutional layer of our CNN already computes 64 separate cross-correlations of the input with learnable filters $\boldsymbol{W}^{(k)}$, where k indexes the 64 filters. We enforce these to be zero-mean by parameterizing $W^{(k)}_{i,j} := \hat{W}^{(k)}_{i,j} - \frac{1}{MN}\sum_{i,j} \hat{W}^{(k)}_{i,j}$ and learning $\hat{\boldsymbol{W}}^{(k)}$, where $M = N = 3$ is the filter size.

[3] We could also use a sigmoid function to ensure $0 < s_f < 1$, but in practice, the bound $s_f < 1$ was not at a risk to be broken during learning.

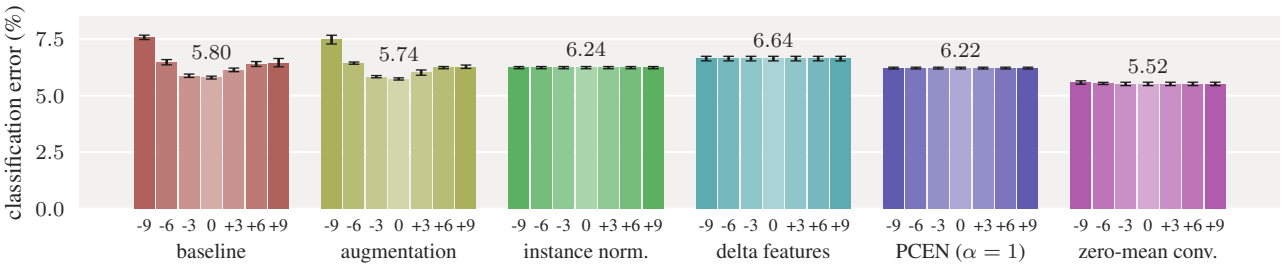

Figure 4: Classification error on our test set for each method with modified input gain between -9 dB to +9 dB. Error bars indicate the standard deviation over five networks. To facilitate comparison, the result at 0 dB is printed at the top.

4. EXPERIMENTS

To compare the five methods and the baseline, we trained and tested each of them on a large public singing voice detection dataset, comparing the quality of their predictions, robustness to level changes, and computational demands.

4.1 Dataset

For our previous work [13], we curated a dataset combining data from Jamendo [18], RWC [8, 16], MSD100 [17], a music video game, YouTube and several instrumental albums. Compared to existing corpora, it is larger and more diverse, both in terms of music genres and by including purely instrumental music pieces – it can be insightful to test a singing voice detection system on music that does not feature vocals as the predominant instrument (for example, Figures 3a,b show excerpts of two instrumental pieces).

In total, the dataset contains almost 80 h of music, split up (without artist overlaps) into 20 h for training, 17.5 h for validation, and 42 h for testing. For a more detailed listing, we refer the reader to [13, Table I].

4.2 Training

Networks are trained to minimize cross-entropy loss on mini-batches of 32 excerpts with ADAM [11]. Weights are initialized following Saxe et al. [19], PCEN parameters $\hat{\delta}_f$ and $\hat{r}_f$ to zeros, $\hat{s}_f$ to $\log(0.025)$, when used. Compared to the public implementation of the baseline system, we use an adaptive learning rate schedule to cope with the larger dataset. We start at a learning rate of 0.001 and drop it to a tenth whenever the training loss [4] did not reach a new minimum for 10 consecutive mini-epochs of 1000 updates each. At each drop, we reset the network weights to the previous minimum. On the third drop, we stop training.

4.3 Evaluation

After training, we compute framewise predictions (network outputs between 0.0 and 1.0) for all validation and test recordings at their original sound level as well as all test recordings at gains of -9 dB, -6 dB, -3 dB, +3 dB, +6 dB, +9 dB. [5] Each sequence of predictions is smoothed in time with a sliding median filter of 800 ms. We determine the optimal classification threshold for the smoothed predictions of the validation set at its original sound level, and apply this threshold to all other predictions. Finally, we compute the classification error for the test recordings, separately for each applied gain.

	Nvidia Titan Xp	Nvidia GTX 970	Intel i7-4770S
baseline	1.7 s	3.0 s	15.2 s
augmentation	1.7 s	3.0 s	15.2 s
instance norm.	42.5 s	103.1 s	643.1 s
delta features	1.7 s	3.0 s	15.2 s
PCEN	6.9 s	9.0 s	15.5 s
zero-mean conv.	1.7 s	3.0 s	15.2 s

Table 1: Computation time required for predicting singing voice in one hour of audio with each method, for two GPUs and a CPU (using a single core).

4.4 Results

Figure 4 depicts our results. The leftmost group of bars shows the classification error of the baseline system: It reaches 5.8% error for the original recordings, but performs worse when scaling the input signals, up to an error of 7.6% for -9 dB (a scale factor of $10^{-9/10} \approx 0.126$).

Training with examples of modified gain apparently does not help: Results at original sound level are comparable to the baseline, and the sound level dependency is as strong as before. Apparently, the augmentation does not sufficiently weaken the dependency between input magnitude and target label. Furthermore, it may not add anything over the existing frequency filtering augmentation, which applies a random gain to a random frequency range.

All remaining methods are invariant to an input gain by construction, so they achieve the same classification error regardless of the gain. [6] In terms of accuracy, spectral delta features perform worst, at an error of 6.6%. Instance Normalization and PCEN (with fixed α_f parameters as explained in Section 3.5) are noticeably better, but still fall significantly behind the baseline system at 6.2% error.

[4] We did not run into any overfitting, possibly because the network was originally designed for a much smaller dataset, and found it beneficial to base the schedule on the training loss rather than the validation loss.

[5] Gains are applied to the input signal expressed as floating-point samples, so positive gains cannot result in clipping.

[6] Note that the converse is not true: a system achieving the same classification error for altered inputs may still be level-dependent, by improving for some examples and failing on others. In [13], we propose an evaluation scheme to rule out this case, but it is not needed here.

When not fixing $\boldsymbol{\alpha}$, PCEN reaches an error of 5.9% at 0 dB, but is as level-dependent as the baseline, with learned α_f between 0.5 and 0.8 (results not included in Figure 4). Finally, zero-mean convolutions slightly exceed the classification accuracy of the baseline system while still being robust to level changes.

As an additional criterion, Table 1 compares the test-time computational demands of the different variants. Using the baseline system, computing framewise singing voice predictions for one hour of audio (with spectrograms already computed) takes 1.7 seconds with a high-end GPU, 3 seconds with a consumer GPU, and 15 seconds on a single CPU core. Since data augmentation and zero-mean convolutions only affect training, and since spectral delta features are cheap to compute, all three are just as fast as the baseline. The IIR filter of PCEN is inherently serial, hindering parallelization. This is not a problem in single-threaded CPU computation, but up to $4\times$ slower than the baseline on GPU. Finally, Instance Normalization requires processing each 115-frame network input separately, preventing reuse of computation in overlapping excerpts. While still fast enough for real-time processing, this poses a huge disadvantage, and is up to $42\times$ slower than the baseline.

5. CONCLUSION

After demonstrating that singing voice detectors are susceptible to partly base their prediction on the absolute magnitude of the input signal, we explore five different ways to reduce or eliminate this dependency in a CNN-based state-of-the-art system. They have different strengths and weaknesses, but one method turned out to be optimal in terms of classification error, robustness to level changes and computational overhead: parameterizing the filters of the first convolutional layer to be zero-mean. When processing logarithmic-magnitude spectrograms, this removes any constant offset resulting from changing the input gain.

Introducing level invariance with zero-mean convolutions is easy and does not measurably affect training time. This might be useful in other machine listening tasks that should not take the sound level into account – either to stabilize predictions against changes in the input gain, as in our case, or even to improve learning from data of varying loudness. To facilitate reuse, our implementation of all five methods is available online. [7]

A dissatisfying aspect of our solution is that it required understanding the problem and introducing a constraint in the parameter space of the neural network. While this is a reasonable way to make progress, it would be helpful to find a method that forces the network to learn this constraint from data. A possible candidate would be Unsupervised Domain Adaptation [7], although initial experiments did not turn out successful. Level-invariant singing voice detection might be a useful test bed, since we already know what a level-invariant CNN can look like.

[7] `https://github.com/f0k/ismir2018` or `http://jan-schlueter.de/pubs/2018_ismir.zip`

In the broader context of the discussion on *horses* [24] (systems that rely on confounding factors for their predictions), our work identified a system to be a horse, and found a way to fix the aspect it identified. Most probably, the system is still partly using the wrong cues, and future work could iteratively find and fix this. However, this may not be the best road to follow: both finding and avoiding confounds is difficult. We discovered the loudness confound after noticing that including the 0^{th} MFCC in the feature set of a classifier unexpectedly improved results, following this trail by testing classifiers with altered examples. Avoiding it required very different approaches for a hand-designed feature set [13] and the CNN addressed here. Another confound, a hypersensitivity of our system to sloped lines in a spectrogram, was discovered by looking at false negatives and false positives, but attempts to avoid it were fruitless [21, p. 190]. A different angle of attack on horses would be to research ways to constrain the learning system to mimic human perception, such that it cannot use cues that humans would not consider in the first place.

6. ACKNOWLEDGEMENTS

This research is supported by the Vienna Science and Technology Fund (WWTF) under grants NXT17-004 and MA14-018. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Tesla K40 GPUs and a Titan Xp GPU used for this research. Last, but not least, we would like to thank the anonymous reviewers for their valuable input.

7. REFERENCES

[1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. C. Catanzaro, et al. Deep Speech 2: End-to-end speech recognition in english and mandarin. *arXiv e-prints*, abs/1512.02595, 2015.

[2] A. L. Berenzweig and D. P. W. Ellis. Locating singing voice segments within music signals. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 119–122, New Paltz, NY, USA, October 2001.

[3] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, October 2014.

[4] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.

[5] F. Eyben, S. Böck, B. Schuller, and A. Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *Proceedings of the 11th*

International Society for Music Information Retrieval Conference (ISMIR), pages 589–594, Utrecht, Netherlands, August 2010.

[6] B. Friedman and H. Nissenbaum. Bias in computer systems. *ACM Transactions on Information Systems*, 14(3):330–347, July 1996.

[7] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, July 2015. PMLR.

[8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, pages 287–288, Paris, France, October 2002.

[9] E. J. Humphrey, N. Montecchio, R. Bittner, A. Jansson, and T. Jehan. Mining labeled data from web-scale collections for vocal activity detection in music. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, October 2017.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, July 2015. PMLR.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[12] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *Proceedings of the 40th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–125, Brisbane, Australia, April 2015.

[13] B. Lehner, J. Schlüter, and G. Widmer. Online, loudness-invariant vocal detection in mixed music signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(8):1369–1380, August 2018.

[14] B. Lehner, G. Widmer, and S. Böck. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, pages 21–25, Nice, France, August 2015.

[15] M. Mauch and S. Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, Curitiba, Brazil, November 2013.

[16] 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 *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 233–238, Miami, FL, USA, October 2011.

[17] N. Ono, Z. Rafii, D. Kitamura, N. Ito, and A. Liutkus. The 2015 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 387–395, Liberec, France, August 2015.

[18] M. Ramona, G. Richard, and B. David. Vocal detection in music with support vector machines. In *Proceedings of the 33rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1885–1888, Las Vegas, NV, USA, March 2008.

[19] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, April 2014.

[20] J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, NY, USA, August 2016.

[21] J. Schlüter. *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*. PhD thesis, Johannes Kepler University Linz, Austria, July 2017.

[22] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 2015.

[23] T. Sercu and V. Goel. Dense prediction on sequences with time-dilated convolutions for speech recognition. In *NIPS Workshop on End-to-end Learning for Speech and Audio Processing*, Barcelona, Spain, November 2016.

[24] B. L. Sturm. A simple method to determine if a music information retrieval system is a "horse". *IEEE Transactions on Multimedia*, 16(6):1636–1644, October 2014.

[25] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv e-prints*, abs/1607.08022, July 2016.

[26] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous. Trainable frontend for robust and far-field keyword spotting. In *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5670–5674, March 2017. arXiv:1607.05666.