Neural Computing and Applications (2020) 32:941–954
https://doi.org/10.1007/s00521-018-3704-x

S.I. : DEEP LEARNING FOR MUSIC AND AUDIO

Basic filters for convolutional neural networks applied to music: Training or design?

Monika Dörfler[1] · Thomas Grill[2] · Roswitha Bammer[1] · Arthur Flexer[2]

Received: 1 December 2017 / Accepted: 10 September 2018 / Published online: 24 September 2018

Abstract
When convolutional neural networks are used to tackle learning problems based on music or other time series, raw one-dimensional data are commonly preprocessed to obtain spectrogram or mel-spectrogram coefficients, which are then used as input to the actual neural network. In this contribution, we investigate, both theoretically and experimentally, the influence of this pre-processing step on the network's performance and pose the question whether replacing it by applying adaptive or learned filters directly to the raw data can improve learning success. The theoretical results show that approximately reproducing mel-spectrogram coefficients by applying adaptive filters and subsequent time-averaging on the squared amplitudes is in principle possible. We also conducted extensive experimental work on the task of singing voice detection in music. The results of these experiments show that for classification based on convolutional neural networks the features obtained from adaptive filter banks followed by time-averaging the squared modulus of the filters' output perform better than the canonical Fourier transform-based mel-spectrogram coefficients. Alternative adaptive approaches with center frequencies or time-averaging lengths learned from training data perform equally well.

Keywords Machine learning · Convolutional neural networks · Adaptive filters · Gabor multipliers · Mel-spectrogram · End-to-end learning

1 Introduction

Convolutional neural networks (CNNs), first introduced in learning tasks for image data [26], have revolutionized state-of-the-art results in many machine learning (ML) problems, also in the domain of audio and, specifically, music. CNNs have been applied to many tasks of the music information research (MIR) realm, like genre and artist classification [11, 27], onset detection [33, 34], structural segmentation [19, 36], chord recognition [22, 25], singing voice detection [23, 28, 29, 35], emotion detection [30], modeling polyphonic music [7, 10] and automatic tagging [8, 9, 12]. In convolutional neural networks, when applied to image data, all filter coefficients are usually learned. For applications to time series, such as audio data, on the other hand, it is common practice to first apply a fixed filter bank to the raw, one-dimensional data in order to generate a feature representation. In traditional audio signal processing methods, used, e.g., in music information retrieval (MIR) or speech processing, FFT-based features such as *mel-spectrograms* are typically used as such inputs. These first level features are two-dimensional arrays, derived from some kind of windowed Fourier transform with subsequent mel-scale averaging.

Recently, the natural question arose, what kind of filters a network would learn if it was given the raw audio input. To date, encouraging results are scarce and so far, a true end-to-end approach for music signals, i.e., acting on raw audio without any pre-processing, has not been able to

✉ Thomas Grill
thomas.grill@ofai.at

Monika Dörfler
monika.doerfler@univie.ac.at

Roswitha Bammer
roswitha.bammer@univie.ac.at

Arthur Flexer
arthur.flexer@ofai.at

[1] Faculty of Mathematics, University of Vienna, 1090 Vienna, Austria

[2] Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6, 1010 Vienna, Austria

outperform models based on linear-frequency spectrogram or mel-spectrogram input [12]. It has been argued that these two ubiquitous representations automatically capture invariances which are of importance for *all* audio signals, in particular, a kind of translation invariance in time (guaranteed by introducing the nonlinear magnitude operation) and a certain stability, introduced by the mel-averaging, to frequency shifts and time-warping (cp. [2]).

In this contribution, we give a formal description of the action of mel-scale averaging on spectrogram coefficients. We show that the resulting mel-spectrogram coefficients can indeed be mimicked by applying *frequency-adaptive* filters, however, *followed by time-averaging* of each filter's squared amplitude output. In order to obtain a close approximation to mel-spectrogram coefficients, the frequency-adaptive filter bank's squared output signals must each undergo a time-averaging operation and the time-averaging window is different for each channel. Furthermore, only dense sampling of the short-time Fourier transform (STFT) leads to almost perfect approximation. Note that the similarity of mel-spectrogram coefficients to the result of time-averaging wavelet coefficients has already been observed in [2], without giving a precise formulation of the connection.[1]

We derive the necessary conditions on the filters, a different one for each bin in the mel-scale, by using the theory of Gabor multipliers and their spreading function, cf. [15]. Considering the description of an operator by means of its spreading function gives interesting insight in the nature of the correlations invoked by the application of the corresponding operator on the signal coefficients. In the case of mel-spectrogram coefficients, it turns out that applying wide triangular windows in the high frequency regions actually corresponds to the application of an operator with little spreading in time. This seems to be the intuitively correct choice for audio signals such as music and speech. While a similar effect can be realized by applying wavelet or constant-Q type filters, the subsequent time-averaging alleviates the significant frequency-spreading effect introduced by rather narrow filtering windows. The observation gained from investigating the classical mel-spectrogram coefficients is, thus, that time- and frequency-averaging spectrogram coefficients provide invariances which are useful in most audio classification tasks, cf. [3]. On the other hand, relaxing the strict averaging performed by computing mel-spectrogram coefficients may intuitively open the opportunity to keep information on details which may be necessary in certain learning tasks.

In our numerical experiments, we thus strive to understand how time- and frequency-averaging influences CNN prediction performance on realistic data sets. The observations drawn from the experiments on learning filters can be summarized as follows:

- Using mel-spectrogram coefficients derived from convolutions with a small sub-sampling factor leads to improved results compared to the canonical FFT-based mel-spectrograms, due to more beneficial influence of the time–frequency sub-sampling parameters.
- Allowing the net to learn center frequencies or time-averaging lengths from the training data leads to comparable improved prediction results.
- Tricks are required to make the CNNs adapt the feature processing stage at all. Otherwise, the classification part of the network takes over the adaptation required to minimize the target loss.

This paper is organized as follows: In the next section, we introduce necessary concepts from time–frequency analysis. In Sect. 3, we give a formal description of the network architecture, since we have not found any concise exposition in the literature. Section 4 then gives the formal result linking mel-spectrogram coefficients with adaptive filter banks. In Sect. 5, we report on the experiments with a real-world data set for the problem of singing voice detection. Finally, we conclude with a discussion and perspectives in Sect. 6.

2 Time–frequency concepts

The Fourier transformation of a function $f \in \mathcal{H}$, for some Hilbert space $\mathcal{H}$, will be denoted by $\mathcal{F}(f)$. We use the normalization $\mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(t) e^{-2\pi i \omega t} \mathrm{d}t$ and denote its inverse by $\mathcal{F}^{-1}(f)(t) = \int_{\mathbb{R}} f(\omega) e^{2\pi i \omega t} \mathrm{d}\omega$. For $x, \omega \in \mathbb{R}$, the translation or time shift operator of a function f is defined as

$$T_x f(t) = f(t - x).$$

and the modulation or frequency shift operator of a function f is defined as

$$M_\omega f(t) = e^{2\pi i t \omega} f(t).$$

The operators of the form $T_x M_\omega$ or $M_\omega T_x$ are called time–frequency shifts. To obtain local information about the frequency spectrum, we define the STFT of a function f with respect to a window $g \neq 0$, where $f, g \in \mathcal{H}$, as

[1] This observation seems to have served as one motivation to introduce the so-called scattering transform, which consists of repeated composition of convolution, a nonlinearity in the form of taking the absolute value and time-averaging. In that framework, mel-spectrogram coefficients are interpreted as first-order scattering coefficients.

$$\mathcal{V}_g f(b,k) = \int_t f(t)\overline{g(t-b)}e^{-2\pi ikt}\mathrm{d}t = \mathcal{F}(f \cdot T_b g)(k). \tag{1}$$

The STFT can be written as an inner product combining the above operators

$$\mathcal{V}_g f(b,k) = \langle f, M_k T_b g \rangle.$$

Taking the absolute value squared, we obtain the spectrogram as $S_0(b,k) = |\mathcal{V}_g f(b,k)|^2$ and $\mathcal{V}_g g$ is called ambiguity function of g, reflecting the time–frequency concentration of g. In practice, sub-sampled and finite versions of the STFT (1) are used, cf. [13]. Sub-sampling obviously corresponds to choosing certain parts of the available information, and this choice can have influence in particular for subsequent processing steps, as we will see in Sect. 4.

3 The structure of CNNs

The basic, modular structure of CNNs has often been described, see, e.g., [18]. Here, we will give a formal statement of the specific architecture used in the experiments in this paper. This architecture has been successfully applied to several MIR tasks and seems to have a prototypical character for audio applications, cf. [19].

The most basic building block in a general neural network may be written as

$$x_{n+1} = \sigma(A_n x_n + b_n)$$

where x_n is the data vector, or array, in the n-th layer, A_n represents a linear operator, b_n is a vector of biases in the n-th layer and the nonlinearity σ is applied component-wise. Note that in each layer the array x_n may have a different dimension. Now, in the case of convolutional layers of CNNs, the matrix A has a particular structure for the convolutional layers, namely, it is a block-Toeplitz matrix, or, depending on the implementation of the filters, a concatenation of circular matrices, each representing one convolution kernel. There may be an arbitrarily high number of convolutional layers, followed by a certain number of so-called dense layers, for which A_n is again an arbitrary linear operator. In this paper, the chosen architecture comprises up to four convolutional and two or three dense layers.

Remark 1 It has been observed in [31, 37–39] that in the context of scattering networks, most of the input signal's energy is contained in the output of the first two convolutional layers. While the context and the filters here are different, this observation might be interesting also as a background for the usual choice of architecture of CNNs for audio processing.

3.1 The CNN with spectrogram input

The standard input in learning methods for audio signal is based on a sub-sampled spectrogram, either in its raw form, or after some pre-processing such as the computation of mel-spectrogram, defined in (4), which we will consider in detail in Sect. 4. In any case, the input to the CNN is an array of size $M \times N$.

Remark 2 In most MIR tasks, the inputs are derived from rather short snippets, that is, about 2–4 s of sound. Considering a sampling rate of 22050 Hz, a window size of 2048 samples and a time shift parameter of 512 samples, i.e., 23 ms, the resulting spectrogram (containing positive frequencies only) is of size $M \times N = 1024 \times 130$, where the latter is the time dimension. Hence, the frequency dimension is, in some sense, over-sampled. In particular, individual bins in the higher frequency regions contain less energy and thus information than in lower regions. Computing the mel-spectrogram is a convenient and straightforward method of reducing the information to typically 80 frequency channels by averaging over increasingly many frequency bins. The number of 80 channels has been determined with preliminary experiments as a breakpoint for optimal CNN performance, obviously because of a sufficient resolution along the frequency dimension. The same setting has already been used in the reference implementation [35] for the experiments of Sect. 5.

We now define the following building blocks of a typical CNN:

- Convolution:
  $S * w(m,n) := \sum_{m'} \sum_{n'} S(m',n') w(m-m', n-n')$
- Pooling: For $1 \le p \le \infty$, we define $A \times B$ pooling as the operator mapping an $M \times N$ array S_0 to a $M/A \times N/B$ array S_1 by

  $$S_1(m,n) = P_p^{A,B}(m,n) = \|v_{S_0}^{m,n}\|_p$$

  where $v_{S_0}^{m,n}$, for $m = 1,\ldots,M/A$ and $n = 1,\ldots,N/B$, is the vector consisting of the array entries $S_0((m-1)\cdot A + 1,\ldots, \quad m\cdot A; (n-1)\cdot B + 1,\ldots,n\cdot B)$. In this work, we use max-pooling, which has been the most successful choice, corresponding to $p = \infty$ in the above formula.
- A nonlinearity $\sigma : \mathbb{R} \mapsto \mathbb{R}$, whose action is always to be understood component-wise. In all but the last layer we use leaky rectified linear units, which allow for a small but nonzero gradient when the unit is not active:

  $$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ cx & \text{otherwise} \end{cases}$$

  for some $c \ll 1$. The output layer's nonlinearity σ_o is a sigmoid function.

We now denote the input array to a convolutional layer by $S_n \in \mathbb{R}^{M_n \times N_n \times K_n}$, where K_n is the number of feature maps of size $M_n \times N_n$ in layer n, i.e., $S_n(k_n) \in \mathbb{R}^{M_n \times N_n}$ for $k_n = 1, \ldots, K_n$. Using the above definitions, we can now write the output of (convolutional) layer $n+1$ with convolutional kernels $w_{n+1} \in \mathbb{R}^{K_{n+1} \times K_n \times M_n \times N_n}$ as follows:

$$S_{n+1}(k_{n+1}) = P_{\infty}^{A_n, B_n} \sigma \left[\left(\sum_{k_n=1}^{K_n} S_n(k_n) * w_{n+1}(k_{n+1}, k_n) \right) + b^{k_{n+1}} \otimes \mathbf{1} \right] \quad (2)$$

where $\mathbf{1}$ is an all-ones array of size $M_n \times N_n$, $b^{k_{n+1}} \in \mathbb{R}^{K_{n+1}}$ and $S_{n+1}(k_{n+1}) \in \mathbb{R}^{M_n/A_n \times N_n/B_n}$ for $k_{n+1} = 1, \ldots, K_{n+1}$.

To formally describe the final, dense layers, we let D_c denote the number of convolutional layers and $S_{D_c} \in \mathbb{R}^{M_{D_c} \times N_{D_c} \times K_{D_c}}$ the output of the last convolutional layer. Then, the overall action of a CNN with two dense layers and a single output unit emitting x_{out}, can be written as

$$x_{\text{out}} = \sigma_o(\mathcal{A}_2 \cdot [\sigma(\mathcal{A}_1 \cdot S_{D_c} + b_{D_c+1})] + b_{D_c+2}). \quad (3)$$

Here, $\mathcal{A}_1$ and $\mathcal{A}_2$ are weight-matrices of size $N_d \times M_{D_c} N_{D_c} K_{D_c}$ and $1 \times N_d$, respectively, where N_d is the number of hidden units in the first dense layer, $b^{D_c+1} \in \mathbb{R}^{N_d}$ and $b^{D_c+2} \in \mathbb{R}$.

3.2 Modifying the input array

As mentioned in the previous section, the spectrogram of audio is often preprocessed in order to reduce the dimensionality, on the one hand, and in order to obtain a spectral representation that better fits both human perception and properties of speech and music on the other hand. Additionally, the authors in [2] pointed out that using mel-spectrogram instead of the spectrogram guarantees improved stability with respect to frequency shifts or, more generally, deformations of the original audio signals, than the usage of spectrograms. However, *given appropriate choice of network architecture*, comparable results can usually be achieved using either the spectrogram or the mel-spectrogram, i.e., the invariance introduced by the mel-averaging can also be learned. In other respects, omitting the frequency-averaging provided by the mel-spectrogram leads to an increase in the number of weights to be learned. On the other hand, these observations raise the question, whether using filters learned directly in the time-domain, would improve the net's ability to achieve the amount of invariance most appropriate for a particular ML task and thus increase stability. The corresponding approach then implies learning time-domain filters already in a layer prior to the first $2D$-convolution. To put this remark into perspective, we note that the spectrogram may easily be interpreted as the combined (and possibly sub-sampled) output of several convolutions, since, setting $\check{h}(n) = h(-n)$, we can write

$$S_0(m,n) = |\sum_{n'} f(n') h(n'-n) e^{-2\pi i m n'}|^2 = |f * \check{h}_m(n)|^2$$

3.3 Questions

In the two following sections, we thus raise and answer two questions:

1. Is it possible to obtain coefficients which are approximately equivalent to the well-established mel-spectrogram coefficients simply by using the 'correct' filters directly on the audio signal?
2. Can adaptivity in frequency- and time-averaging improve prediction accuracy? In particular, for a given set of frequency-adaptive filters precisely mimicking the mel-scale, can a time-averaging layer with learned averaging width improve learning performance?

4 The mel-spectrogram and basic filters

In this section, we take a detailed look at the mel-spectrogram. This representation is derived from the classical spectrogram by weighted averaging of the absolute values squared of the STFT and can undoubtedly be referred to as the most important feature set used in speech and audio processing, together with MFCCs which are directly derived from it. The number of mel-filters used varies between 80 filters between 80 and 16 kHz [19] and 128 [12] or more. In order to better understand the relation between the result of mel-averaging and FFT-based analysis with flexible windows, we observe the following: denote the input signal by $f \in \mathbb{C}^N$, the window function for generating the spectrogram by $g \in \mathbb{C}^N$ and the mel-filters, typically given by simple triangular functions, by $\Lambda_\nu \in \mathbb{C}^N$ for $\nu \in \mathcal{I} = \{1, \ldots, K\}$, where K is the chosen number of filters. We can then write the mel-spectrogram as

$$\text{MS}_g(f)(b, \nu) = \sum_k |\mathcal{F}(f \cdot T_b g)(k)|^2 \cdot \Lambda_\nu(k). \quad (4)$$

Andén and Mallat postulated in [2] that the mel-spectrogram can be approximated by time-averaging the absolute values squared of a wavelet transform. Here, we make their considerations precise by showing that we can get a close approximation of the mel-spectrogram coefficients if we use adaptive filters.

Remark 3 Note that the resulting transform may be interpreted as a nonstationary Gabor transform, compare [4, 5, 14, 21].

In practice one always uses a sub-sampled version of the STFT, i.e., we consider time-sampling points in $\alpha\mathbb{Z}$ and the

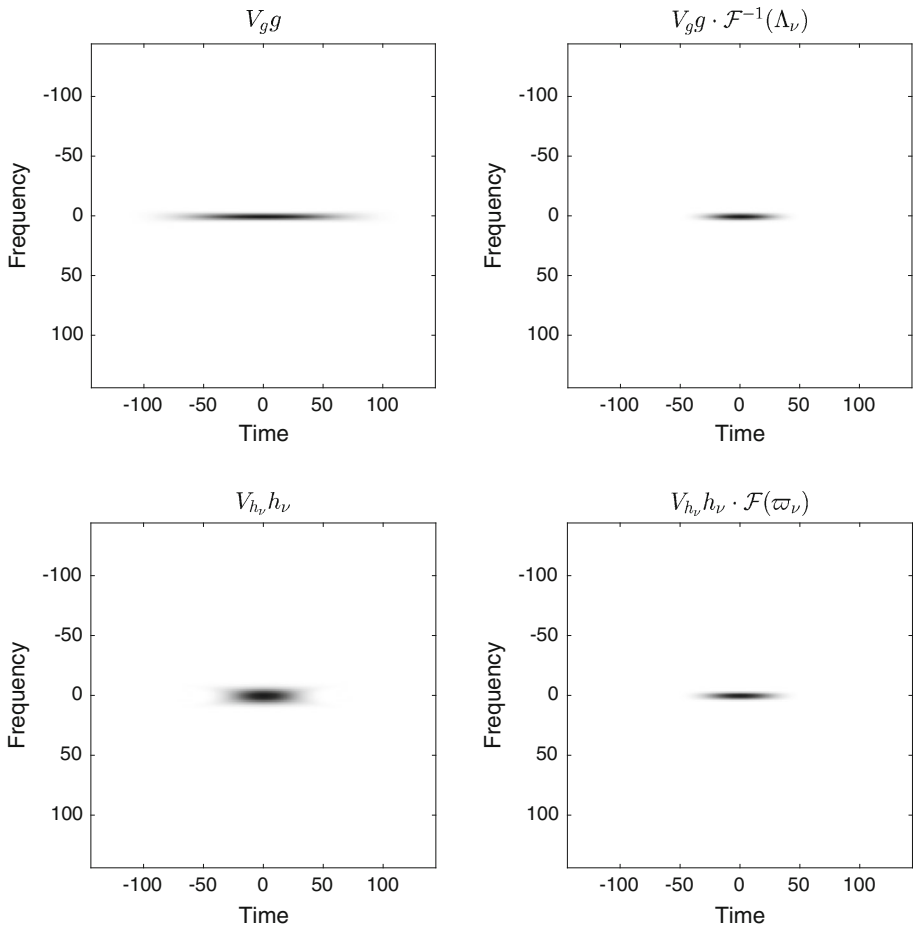

Fig. 1 Ambiguity functions $\mathcal{V}_g g$, $\mathcal{V}_{h_v} h_v$, and weighted ambiguity functions $\mathcal{V}\mathcal{V}_g g g \cdot \mathcal{F}^{-1}(\Lambda_v)$, $\mathcal{V}_{h_v} h_v \cdot \mathcal{F}(\varpi_v)(\xi)$ used for the computation of adaptive filtering, for $v = 50$. It is clearly visible that the surplus in frequency spread introduced by the narrower window h_v is removed by time-averaging. On the other hand, frequency-averaging reduces the time-spread of the wider window g

Fourier transform in (4) is sampled on $\beta\mathbb{Z}$. We then compare two different settings which lead to a time–frequency feature map which is then used as input to the deeper layers of the CNN:

1. STFT-based: Compute spectrogram and take weighted averages over certain regions in frequency; for the classical mel scale, this leads to the mel-spectrogram coefficients, but other choices of Λ_v are possible. Taking time- and frequency-sampling parameters α, β into account, the resulting time–frequency feature map is computed for $b = \alpha l_0$ as follows:

$$\mathrm{MS}_g(f)(b, v) = \sum_k |\mathcal{F}(f \cdot T_b g)(\beta k)|^2 \cdot \Lambda_v(\beta k). \tag{5}$$

2. Filter bank-based: compute filtered version of f with respect to some, possibly adaptive, filter bank h_v, $v \in \mathcal{I}$ and apply subsequent time-averaging using a time-averaging function ϖ_v:

$$\mathrm{FB}_{h_v}(f)(b, v) = \sum_l |(f * h_v)(\alpha l)|^2 \cdot \varpi_v(\alpha l - b). \tag{6}$$

The following central theorem gives an estimate for the difference between the two above approaches for each entry in the feature maps.

Theorem 1 *For all $v \in \mathcal{I}$, let $g, h_v, \Lambda_v, \varpi_v$ be given. Let $\mathrm{MS}_g(f)$ and $\mathrm{FB}_{h_v}(f)$ be computed on a lattice $\alpha\mathbb{Z} \times \beta\mathbb{Z}$ and set*

$$\mathcal{M}^v(x) = \sum_l T_{\frac{l}{\beta}} \mathcal{F}^{-1}(\Lambda_v)(x) \text{ and } \mathcal{M}_F^v(\xi) = \sum_k T_{\frac{k}{\alpha}} \mathcal{F}(\varpi_v)(\xi). \tag{7}$$

Then, the following estimate holds for all $(b, v) \in \alpha\mathbb{Z} \times \mathcal{I}$:

$$|\mathrm{MS}_g(f)(b, v) - \mathrm{FB}_{h_v}(f)(b, v)| \leq \|\mathcal{V}_g g \cdot \mathcal{M}^v - \mathcal{V}_{h_v} h_v \cdot \mathcal{M}_F^v\|_2 \cdot \|f\|_2^2 \tag{8}$$

A technical proof of Theorem 1 is included in Appendix A. The basic idea of the proof lies in expressing both $\mathrm{MS}_g(f)$ and $\mathrm{FB}_{h_v}(f)$ by means of a bilinear form generated by different specific time–frequency multipliers. The underlying operators can then be compared using their respective *spreading functions*, [15, 16], an alternative operator description. An operator's spreading function gives an intuition about the operator's action in the space of

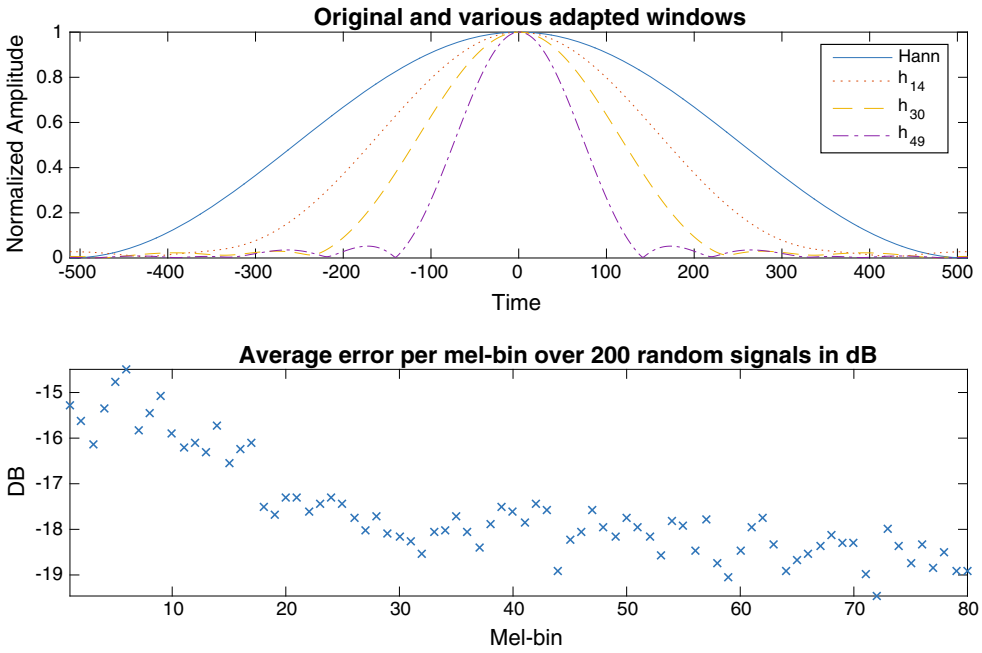

Fig. 2 Upper plot: (original) Hann window and adapted windows; lower plot: error in approximation of mel-spectrogram coefficients by adaptive filtering and subsequent time-averaging on the squared amplitudes

time-lag and frequency-lag. Time–frequency multipliers' spreading functions enjoy a simple form, which is simply the product of the analysis windows' ambiguity function with a two-dimensional Fourier transform of the multiplier sequence. Figure 1 shows the ambiguity functions $\mathcal{V}_g g(x,\xi)$, $\mathcal{V}_{h_v} h_v(x,\xi)$ which would correspond to the operators without frequency- or time-averaging, respectively, and the weighted ambiguity functions $\mathcal{V}_g g(x,\xi)\cdot\mathcal{F}^{-1}(\Lambda_v)(x)$, corresponding to the ambiguity function after mel-averaging in frequency by Λ_v and $\mathcal{V}_{h_v} h_v(x,\xi)\cdot\mathcal{F}(\varpi_v)(\xi)$ corresponding to the filter bank approach after time-averaging by ϖ_v. It is obvious that frequency-averaging reduces the time-lag of the operator while time-averaging reduces the higher frequency-lag introduced by the narrower windows in the adaptive filter bank; overall very close behavior can be achieved with

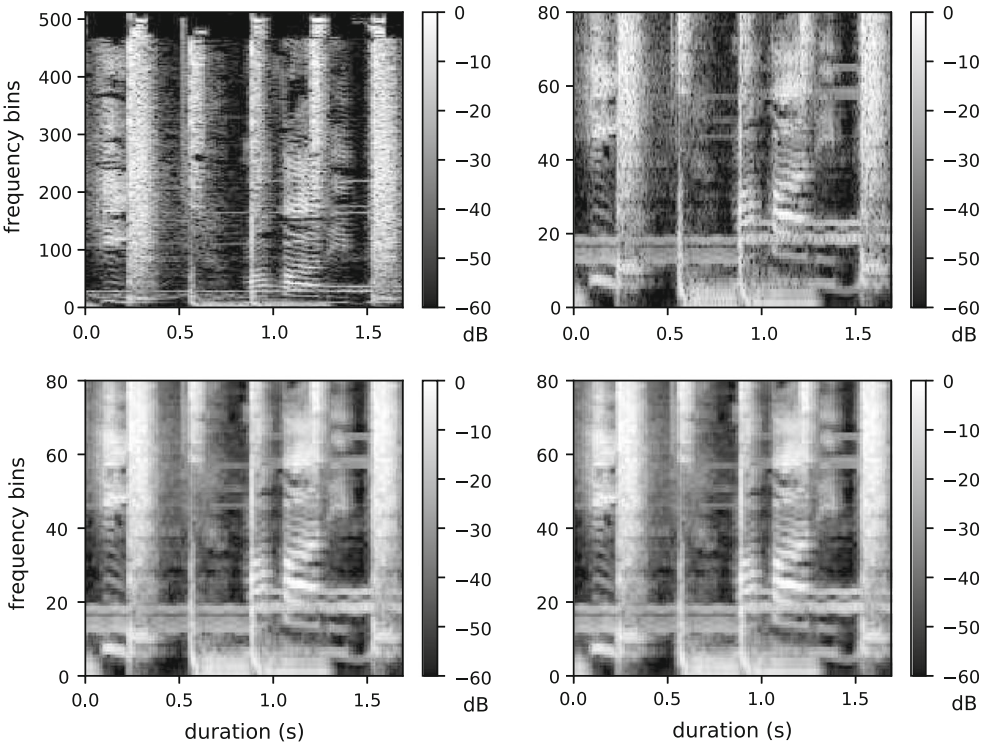

Fig. 3 Time–frequency representations for the problem of singing voice detection. The spectrograms shown are STFT (upper left), STFT-based mel-spectrogram (bottom left), filter bank computed (top right), and filter bank with time-averaging (bottom right)

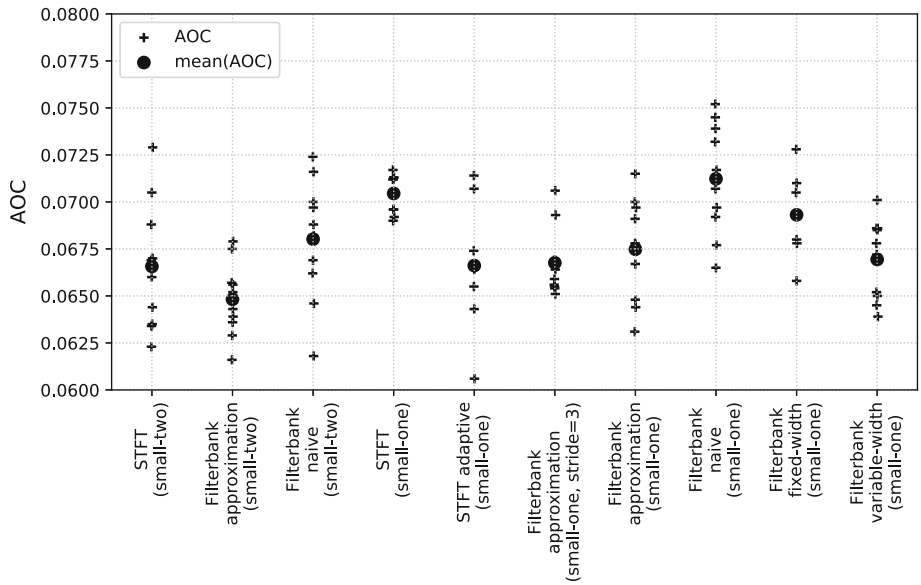

Fig. 4 AOC measures for the problem of singing voice detection. Models compared are results for multiple runs of fivefold cross-validation on batch-normalized features, for the CNN architectures 'small-two' and 'small-one.' On the one hand, the features are mel-spectrograms (*STFT*), or spectrograms with trained center frequencies (*STFT adaptive*). On the other hand, we evaluated the approximative filters derived in Sect. 4 (*Filter bank approximation*), filter banks and fixed-width temporal averaging (*Filter bank naive* with a rectangular window, and *Filter bank fixed-width* with a Hann window), and adaptive variable-width Hann-window averaging (*Filter bank variable-width*). The default convolution stride is 21, unless otherwise noted. Shown are individual results (gray crosses) and their mean values (black dots)

both approaches, in particular with small α, β. For the fully sampled case, i.e., $\alpha = \beta = 1$, we obtain the following expression:

$$\|\mathrm{MS}_g(f) - \mathrm{FB}_{h_\nu}(f)\|_\infty \leq \|\mathcal{V}_{h_\nu} h_\nu \cdot \mathcal{F}(\varpi_\nu) - \mathcal{V}_g g \cdot \mathcal{F}^{-1}(\Lambda_\nu)\|_2 \cdot \|f\|_2^2 \tag{9}$$

This leads to a statement about precise recovery of mel-spectrogram coefficients by filter bank approximation.

Remark 4 A preliminary version of the following statement has been announced without proof in [14].

Corollary 1 *Let an analysis window g and mel-filters Λ_ν be given, for $\nu \in \mathcal{I}$. If, for each ν, the windows h_ν and time-averaging functions ϖ_ν are chosen such that*

$$\mathcal{V}_{h_\nu} h_\nu(x,\xi) \cdot \mathcal{F}(\varpi_\nu)(\xi) = \mathcal{V}_g g(x,\xi) \cdot \mathcal{F}^{-1}(\Lambda_\nu)(x), \tag{10}$$

then the mel-spectrogram coefficients can be obtained by time-averaging the filtered signal's absolute value squared, i.e., for all $(b,\nu) \in \mathbb{Z} \times \mathcal{I}$:

$$\mathrm{MS}_g(f)(b,\nu) = \mathrm{FB}_{h_\nu}(f)(b,\nu). \tag{11}$$

Example 1 While it is in general tedious to explicitly derive conditions for the optimal filters h_ν and the time-averaging windows ϖ_ν, we obtain a more accessible situation if we restrict the choice of windows to dilated Gaussians $g(t) = \varphi_\sigma(t) = (\frac{2}{\sigma})^{\frac{1}{4}} e^{-\pi \frac{t^2}{\sigma}}$, for which $\mathcal{V}_{\varphi_\sigma} \varphi_\sigma(x,\xi) = e^{-\frac{\pi x^2}{2\sigma}} e^{-\frac{\pi}{2}\sigma\xi^2} e^{-\pi i x \xi}$. Thus, fixing $g = \varphi_\sigma$ for some scaling factor σ, letting the filters Λ_ν be given as shifted and dilated versions of a basic shape (e.g., in the case of mel-filters, asymmetric triangular functions), i.e., $\Lambda_\nu(\xi) = T_\nu D_{a(\nu)} \Lambda(\xi)$, for $\nu \in \mathcal{I}$ and assuming that each filter h_ν is a dilated and modulated Gaussian window, i.e., $h_\nu(t) = e^{2\pi i \nu t} \varphi_{\rho(\nu)}(t)$, condition (10) leads to the following conditions in separate variables:

$$e^{-\frac{\pi}{2}x^2(\frac{1}{\rho(\nu)} - \frac{1}{\sigma})} = \mathcal{F}^{-1}(D_{a(\nu)}\Lambda)(x) \text{ and } e^{-\frac{\pi}{2}\xi^2(\sigma - \rho(\nu))} = \mathcal{F}(\varpi_\nu)(\xi).$$

From the example, it can be seen that even in the case of Gaussian analysis windows a precise recovery of standard mel-spectrogram coefficients is possible, if the involved analysis windows and averaging windows are appropriately scaled Gaussians. In the more realistic case of compactly supported analysis windows such as Hann windows, triangular frequency-averaging functions Λ_ν typically used for computing the mel-spectrogram coefficients and coarser sampling schemes, we have to resort to alternative methods for obtaining the filter bank-based approximation.

4.1 Computation and examples of adaptive filters

We now describe the strategy for computing adaptive filters leading to a filter bank-based approximation of mel-like coefficients based on general windows. Very often, these windows will be compactly supported and their STFT will not factorize in two components in separate variables x, ξ. Since g and Λ_ν are fixed, and $\mathcal{F}(\varpi_\nu)(\xi)$ only allows for a multiplicative constant in each frequency bin, we can only perfectly adapt h_ν in one frequency bin. We thus use the following trick for computing h_ν for a given mel-filter Λ_ν: we consider the right-hand side of (10) in $\xi = 0$. This is justified if $\mathcal{F}(g)$, and thus $|\mathcal{V}_g g(x,\xi)| \leq (|\hat{g}| * |\hat{g}|)(\xi)$, decays fast in the frequency variable ξ; this is typically the case for the windows used in practice, such as Hann windows, since we strive to obtain separation between the frequency bins. Therefore, the component at $x = 0$ will have by far the strongest influence on the error made when minimizing (9), and we will use it to obtain h_ν. We then have, with $\check{g}(x) = \overline{g}(-x)$, the following version of (10):

$$\begin{aligned}&\mathcal{V}_{h_\nu} h_\nu(x,0) \cdot \mathcal{F}(\varpi_\nu)(0) = \mathcal{V}_g g(x,0) \cdot \mathcal{F}^{-1}(\Lambda_\nu)(x)\\ &\quad \Rightarrow (h_\nu * \check{h_\nu})(x) \cdot \mathcal{F}(\varpi_\nu)(0) = (g * \check{g})(x) \cdot \mathcal{F}^{-1}(\Lambda_\nu)(x).\end{aligned}$$

Taking Fourier transform on both sides, we obtain

$$\mathcal{F}(h_\nu * \check{h_\nu}) = |\mathcal{F}(h_\nu)(\xi)|^2 = \mathcal{F}((g * \check{g}) \cdot \mathcal{F}^{-1}(\Lambda_\nu))(\xi),$$

and compute h_ν as

$$h_\nu(t) = \mathcal{F}^{-1}\left(\sqrt{\mathcal{F}((g * \check{g}) \cdot \mathcal{F}^{-1}(\Lambda_\nu))}\right)(t)$$

Similarly, by setting $x = 0$ in the left-hand side of (10), we compute

$$\mathcal{F}(\varpi_\nu)(\xi) = \mathcal{V}_g g(0,\xi) \cdot \mathcal{F}^{-1}(\Lambda_\nu)(0) / \mathcal{V}_{h_\nu} h_\nu(0,\xi)$$

where we only consider values of $\mathcal{V}_{h_\nu} h_\nu(0,\xi)$ above a threshold ε.

We now give some examples of filters h_ν computed to obtain mel-coefficients $\mathrm{MS}_g(f)$ by time-averaging $|(f * h_\nu)(l)|^2$ as in (11) following the procedure described above. We consider Hann windows, which is the standard choice in audio processing, also applied in the computation of mel-spectrogram coefficients and their approximation in Sect. 5. Starting from a Hann window g, we compute adaptive filters h_ν for 80 bins of the mel-scale. Figure 1 shows the ambiguity functions $\mathcal{V}_g g$, $\mathcal{V}_{h_\nu} h_\nu$, and the weighted ambiguity functions $\mathcal{V}_g g \cdot \mathcal{F}^{-1}(\Lambda_\nu)$, $\mathcal{V}_{h_\nu} h_\nu \cdot \mathcal{F}(\varpi_\nu)(\xi)$, for $\nu = 49$, which corresponds to 2587.6 Hz.

In Fig. 2, the upper plot shows the original Hann window g, which had been used to compute the spectrogram from which the mel-spectrogram coefficients are derived, and three adapted windows. Note that the adapted windows get shorter in time with increasing mel-number; this effect serves to realize the mel-averaging by adaptivity in the frequency domain. The lower plot shows the average error per bin obtained from computing the mel-spectrogram coefficients and their approximations for 200 (normally distributed) random signals.

For an illustration of the time–frequency representations applied to a real audio signal, cf. Fig. 3.

5 Experiments on singing voice detection

In Theorem 1, it is shown that coefficients with mel-characteristics (and other related nonlinear scales) can be closely approximated by applying appropriately chosen filters directly to raw audio data and allowing for a subsequent time-averaging step on the squared absolute output. Now, we are interested in investigating if the theoretical findings translate to typical real-world problems that have already been successfully treated with CNNs. Thereby, we are motivated by the fact that state-of-the-art results for several MIR problems are based on mel-spectrogram coefficients which show certain desirable invariance and stability properties. In particular, due to the modulus, they are invariant to translation and, due to the frequency-averaging, they exhibit stability to certain deformations such as time-warping, cf. [2]. However, in general, the required invariance and stability with respect to deformations will depend on data characteristics and the learning task, cf. [32]. In our experiments, we hence start from the filter bank-based computation of approximative mel-coefficients, cf. (6). In the sequel, the results obtained from the filter bank-based coefficients can serve as a reference point and they should not be significantly worse than the results achieved when using the standard mel-coefficients as input. This reference is necessary, since certain implementation details are different for the original mel-coefficients and their filter bank approximation. This concerns, in particular, pre-processing steps such as batch normalization or padding. The adaptation of the time-averaging starts from this implementation, so that we needed to rule out adversarial effects stemming from sources other than the adaptation process. For the adaptive scenarios, we allow parts of the feature processing stage (time-averaging lengths, center frequencies) to be learned by the network, posing the question whether adaptivity in this step can improve the network's performance.

We need to note that, when trained on a specific problem, both the feature layers (including the adaptive time-averaging step) and the classification part of a CNN will concurrently adapt their parameters toward optimally

predicting the given targets. We will discuss the implications of this behavior for our experiments in Sect. 5.4.

Our hypothesis is that a CNN with an architecture that is adapted to a given learning task will learn filters—in this case their adaptive components—which alleviate the extraction of stabilities and invariance properties and are thus beneficial in the given context.

5.1 Data

We investigate the effects of learning filters directly on raw audio by revisiting the problem of *singing voice detection* [35] we have studied before. In the referenced publication, a CNN was tuned for maximum prediction accuracy both in the absence or presence of various forms of data augmentation.

The experiments were performed on a non-public dataset of 188 30-s audio snippets from an online music store (dataset 'In-House A'), covering a very wide range of genres and origins. For the evaluation, we used a fivefold cross-validation with slightly unequal folds, for each iteration 150 or 151 files for training, the remaining 37 or 38 for evaluation. The testing folds are non-overlapping and add up to the total of 188 items. The audio was sub-sampled to a sampling rate of 22.05 kHz and down-mixed to mono. The mel-spectrograms were calculated using an STFT with Hann windows, a frame length of 1024 and a frame rate of 70 per second (equivalent to a hop size of 315 samples).

For this paper, instead of magnitude spectra, as in the reference model, we use power spectra as in (4), also following the convention used in [2]. We apply a filter bank with 80 triangular mel-scaled filters from 27.5 Hz to 8 kHz and then logarithmize the squared magnitudes (after clipping values below 10^{-7}).

We have also left out any form of data augmentation. For the context of this paper, where we are interested in fundamental qualities of feature representation rather than maximum prediction performance, data augmentation would not be beneficial, but would rather negatively impact training times.

5.2 CNN training procedure and architecture

The training procedure used in our experiments is slightly different than in the reference publication [35]. The networks are trained on mel-spectrogram excerpts of 115 spectrogram frames ($\sim$1.6 s) paired with a binary label denoting the presence or absence of human voice in the central frame. Training is performed using stochastic gradient descent on cross-entropy error based on mini-batches of 64 randomly chosen examples. Updates to the network weights are computed using the ADAM update rule [24] with an initial learning rate of 0.001 and an adaptive scheme reducing the learning rate twice by a factor of 10 whenever the training error does not improve over three consecutive episodes of 1000 updates. Evaluation is performed running a complete fivefold cross-validation run to obtain predictions for the whole set of training data, with this procedure repeated multiple times with different network initialization and data ordering.

As described in Sect. 3.1, the applied CNN architecture employs three types of feed-forward neural network layers: convolutional *feature processing layers* convolving a stack of 2D inputs with a set of learned 2D kernels, *pooling layers* sub-sampling a stack of 2D inputs by taking the maximum over small groups of neighboring pixels, and dense *classification layers* flattening the input to a vector and applying a dot product with a learned weight matrix $\mathcal{A}_j$.

The architecture used in [35] has a total number of 1.41 million weights, with the dense connections of the classification layers taking up the major share (1.28 million, or 91%). It can be expected that the actual output of the convolutional feature stage is of subordinate importance when the classification stage with its high explanatory power dominates the network.

If data augmentation is not considered, the network size—especially the classification part—can be drastically reduced while largely preserving its performance. This size reduction is possible, since, as a general rule, the necessary number of parameters determining the network is correspondent to the complexity of the training data set. As we are interested in the impact of the convolutional feature stage's properties, we reduce the architecture for our experiments as follows: We use four convolutional layers, two 3×3 convolutions of 32 and 16 kernels, respectively, followed by 3×3 non-overlapping max-pooling and two more 3×3 convolutions of 32 and 16 kernels, respectively, and another 3×3 pooling stage.

With the conventions of (2), with a slight abuse of notation by noting the number of nonzero elements in the convolutional kernels instead of the underlying dimension of convolution, the applied setting corresponds to

- $K_0 = 1$, $K_1 = K_3 = 32$, $K_2 = K_4 = 16$;
- $w_1 \in \mathbb{R}^{32\times1\times3\times3}$, $w_2 \in \mathbb{R}^{16\times32\times3\times3}$;
- $A_1 = B_1 = 1$, $A_2 = B_2 = 3$
- $w_3 \in \mathbb{R}^{32\times16\times3\times3\times16}$, $w_4 \in \mathbb{R}^{16\times32\times3\times3}$;
- $A_3 = B_3 = 1$, $A_4 = B_4 = 3$

For the classification part, we experimented with two variants: one with two dense layers of 64 and 16 units ('small-two'), and the other one with just one dense layer of 32 units ('small-one'). In both cases, the final dense

layer is a single sigmoidal output unit. For the first variant, the total number of weights is 94,337, with the classification stage taking up 79,969 units, or 85%. The second variant features a considerably smaller classification network: the total number of weights is 53,857, with the classification stage taking up 39,489 units, or 73%. The different network sizes, especially the ratio of feature to classification stage, allow us to analyze the influence of the different parts. Specifically, we expect the performance of the 'small-one' architecture to be more directly connected to the quality of the time–frequency representation.

5.3 Experimental setup

In the following, we will compare the behavior of the CNNs applied to the STFT-based mel-spectrogram features to features computed using filter banks as described in Sect. 4. Both are computed in end-to-end fashion ad hoc from the audio signal. The maximum kernel sizes of the filter banks are set to 1024, identical to the frame length of the previously used STFT. The training examples are snippets of the audio signal with a length of $115 \times 315 + 1024 - 1 = 37{,}248$ samples each with a hop size of 315 samples.

To judge the influence of adaptivity, four different approaches have been compared:

1. 'Filter bank, approximation': Filter bank and time averaging as derived in Sect. 4.
2. 'Filter bank, naive': Filter bank with Hann envelopes. The kernel size equals the time support for the lowest frequency band (50 Hz) and reduces, according to the band-width requirements of the mel frequency scale, down to 94 samples for the highest band at 7740 Hz. After the filter bank, fixed-size time-averaging by pooling for improved computational efficiency.
3. 'Filter bank, fixed-width': Filter bank as in 2, but with fixed-size time-averaging using a convolution with a Hann window.
4. 'Filter bank, variable-width': Adaptive time-averaging after the filter bank, with individual adaptation per frequency bin, learned from the training data.

For reasons of computational cost, it is not feasible to perform a full sample-by-sample convolution for the filter bank. For the bulk of our filter bank experiments, we have chosen a convolution stride for the filters of 21 samples, that is, the resulting spectrum is down-sampled along the time axis by a factor of 21. The subsequent non-overlapping averaging is computed on 15 frames each, in order to stay comparable with the STFT hop size of $315 = 21 \times 15$ samples. Note that the stride is a factor of about 4.5 lower than the shortest kernel support (21 vs. 94). For comparison, we have also experimented with smaller convolution strides (3 and 1) to assess their impact on the results.

For the 'naive' fixed-size time-averaging variant standard average-pooling is used, implemented as a 15×1 2D-pooling layer acting on the power spectrum. In this case, the temporal averaging length is uniform over the frequency axis which is a crude approximation of the mathematical findings. The 'fixed-width' and 'variable-width' cases are implemented using Hann windows, the latter with adaptive width, individual for each frequency bin. The maximum time support of this Hann window is 8 times the STFT hop size, equivalent to 2520 samples. The choice of Hann in contrast to a Boxcar window (as in the 'naive' case) is motivated by its smoothness which aids adaptivity for the CNN training process.

Figure 3 illustrates the time–frequency representations used in this paper. The STFT case is shown on the left-hand side with the full Fourier spectrum (512 bins) on top and its mel-spectrogram (80 bins) at the bottom. On the right-hand side, the top shows a filter bank-computed mel-scaled spectrogram using the filters derived in Sect. 4, and the time-averaged counterpart at the bottom. Note that the two bottom spectrograms are equivalent.

5.4 Experimental results

Figure 4 shows the results of our CNN experiments for the problem of singing voice detection. For our evaluations, we have switched from the simple error measure with the 'optimal' (in the sense of maximum accuracy) threshold per experiment to the more informative 'area over the ROC curve' measure (AOC), fusing classification errors for all possible thresholds into one measure. A lower measure indicates a better result.

The reference implementation in [35] uses pre-computed spectrograms, with a normalization globally on the training set and eventual padding performed also on the spectrogram. End-to-end learning as performed in our experiments demands on-line normalization (using a batch normalization layer) and padding directly on the audio time signal. We could verify that this yields a performance equivalent with the reference experiments.

We can also confirm that the performance of our 'small-two' network with two classification layers is comparable to the large baseline architecture. For AOC in the STFT case, the smaller networks score 6.66% ('small-two') and 7.05% ('small-one'), respectively, compared to 6.74% of the original architecture (the latter not shown in Fig. 4). The difference between the reference and the 'small-two' architecture is not significant (t test, $p = 5\%$), while the difference between 'small-two' and 'small-one' is.

In the course of experimentation, it has become apparent that the time-averaging widths of the adaptive models hardly train at all, especially for larger classification stages. They rather stay close to the initial values, while the CNN weights adapt instead. As a trick, we have *boosted* the widths' gradients for the back-propagation by a factor 3 to force the width parameters to adapt at a higher rate. Higher factors have proven unfeasible, causing the adaptation to run out of bounds. Since the adaptation process is intricate, the choice of a starting value for the variable averaging length (time support of the Hann window) is important. We have tried values of 0.1, 0.2, 0.3, 0.5 of the maximum filter bank time support, with 0.2 (equivalent to 504 samples) leading to the best results.

5.4.1 Interpretation of results

As a first observation, we see in Fig. 4 that for both architectures the filter bank approximation scores better than the canonical STFT case (significant for the 'small-one' architecture at $p<5\%$). This can be explained by considering the different kinds of aliasing terms which affect the computation of the feature maps: Eq. (8) shows that the STFT-based approach leads to aliasing in time while the filter bank-based approach leads to aliasing in frequency. From the results, we can deduce that the impact of the time aliases imposed by mel-averaging is stronger than that of the frequency aliases stemming from the time-averaging. Furthermore, reducing time-aliases in the first approach would require using a longer FFT in the computation of the underlying STFT, while reducing the influence of the frequency aliases is accomplished by decreasing the convolution stride: Using a default convolution stride of 21 corresponds to a sub-sampling factor $\alpha = 21/1024 = 0.02$ in time as opposed to $\alpha = 315/1024 = 0.3$ in the STFT case. Heuristically, we obtain a more stable estimate for the local frequency components, cp. the recent work in [1]. We were able to confirm this trend by using even smaller convolution strides (sub-sampling factor 3 instead of the standard 21) which led to a slightly, albeit insignificantly, better score. These observations indicate that the actual time–frequency resolution of the signal representation used in the first processing step can lead to advantages in the overall performance of the CNN, which cannot necessarily be provided by subsequent convolutional or dense layers. To our knowledge, this is the first formal description of such an effect.

The second observation concerns the influence of leveraging adaptivity in the learning process: In comparison with the filter banks with filter coefficient approximations according to the theory, the 'naive' (significant at $p = 5\%$) and the 'fixed-width' (not significant) variations exhibit slightly worse performance for both architectures. The 'variable-width' variation with adaptive time-averaging scores significantly better than its 'fixed-width' counterpart, and is statistically equivalent ($p > 60\%$) to the filter bank 'approximation' case.

At the same level of performance lies the 'STFT adaptive' case which is a variation of the STFT case; here, we applied frequency-averaging of the spectrogram coefficients, just as in the STFT-based computation of the mel-coefficients, but allowed the CNN to learn—and thus adapt—the center frequencies during training. The low and high frequency bounds remained fixed, but the intermediate frequencies were free to adapt with the condition of monotonicity. It is noticeable that the adapted center frequencies remain relatively close to the mel scale, with only a few percent of relative deviation ($Q_1 > -8.6\%$, $Q_3 < +3.7\%$ over all bands for 10 runs with 5 folds each).

In general, the different adaptive models exhibit very similar performance measures AOC $<6.75\%$ which is significantly better than the canonical STFT-based case at AOC $= 7.05\%$ for the 'small-one' architecture. As expected, the effects of adaptivity on the evaluation results are more pronounced for the smaller architecture, with less explanatory power in the classification stages. We remark that in previously performed experiments on a fully adaptive approach (adaptive filter lengths + adaptive time-averaging) the learning process did not converge and the achieved results were consistently worse than those based on standard approaches. Therefore, we restricted the relaxation of fixed parameters to either time-averaging length or frequency centers in the computation of the now variable, adaptive frequency filters and the experiments showed that both approaches perform almost identically. In these scenarios, only 80 trainable parameters are added to the number of networks weights described in Sect. 5.2 and the increase in computational cost as well as required amount of training data is negligible.

Finally, note that the filter bank approximation with stride 3 in the convolution performs identically to the setup with the adaptive time-averaging (variable-width), which, in turn, is slightly better than the stride 21 approximation setting. The fact that the improvement is only small, can be seen as an indication that the ideal mel-coefficients indeed yield a representation that is sufficiently good for the subsequent convolutional layers to get close to an optimal result. Furthermore, as stated before, it seems that the expressivity of the network architecture is so high that it can actually obtain good results from different representations which are sufficiently reasonable. In this sense, the observation that some adaptivity in the primary representation, on the one hand, and the geometry of the sampling grid and consequential nature of occurring aliases do have some influence on the final performance, is quite remarkable.

6 Discussion and perspectives

In Sect. 3.3, we posed two questions concerning the application of alternative time–frequency representations for learning problems in music information retrieval.

First, it has been analytically shown under which conditions mel-spectrogram coefficients can be reproduced by applying frequency-adaptive filters followed by time-averaging the squared amplitudes. In practice, this procedure will always lead to approximate values due to their computation from sub-sampled values.

Answering the second question, we have found that these *designed* spectrogram representations yield significantly increased performance on the task of CNN-based singing voice detection. The improvement in performance can be ascribed to a sub-sampling scheme implicit in the usage of the designed adaptive filters, which yields a more advantageous suppression of adversarial time–frequency aliases than the canonical computation of mel-spectrogram coefficients. Furthermore, adaptivity by *training* in the time-averaging layer, or alternatively, using frequency-adaptive triangular filters on the Fourier spectrograms, on the other hand, also lead to improved results relative to the canonical STFT-based mel-spectrograms. These results are performance-wise statistically equivalent to the filters derived by the mathematical theory developed in Sect. 4. Hence, similar results were obtained both with properly *designed* representations and representations whose crucial parameters were *trained* on the data.

Summing up, we conclude that the subtle differences in time–frequency resolution of the basic filters used to obtain the signal representation do influence the overall performance of a CNN applied to a typical MIR task, at least for architectures of rather modest size. The choice of the well-established mel frequency scale in the first place for our experiments seems justified not only by prior work on time-domain filters calculated *ex nihilo* (cf. [12, Section 4.2]), but also by our own findings that adaptive center frequencies deviate from the mel scale only to a small extent. We conclude that the chosen scale provides a useful compromise between time- and frequency-averaging for the task under consideration.

Future work on the problem of learned basic filters in MIR tasks will involve the study of the precise connection between the characteristics of a given data set and the most advantageous analysis windows and sampling schemes used to compute the spectrogram. These investigations will concern both the network's expressivity and the performance of the learning process, cf. preliminary work in [14] and will be based on data sets with different time–frequency characteristics as well as various learning tasks. Finally, future work will also address the more general question of the propagation and alleviation of small approximation errors through the network and their dependence on various network parameters as well as the network's architecture, relying on existing results on stability of CNNs, compare [6, 20, 40].

Acknowledgements This research has been supported by the Vienna Science and Technology Fund (WWTF) through Project MA14-018.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix A: Proof of Theorem 1

In order to include the situation described in Theorem 1, we assume the situation in which the original spectrogram is sub-sampled, in other words, we start the computations concerning a signal f from

$$S_0(\alpha l, \beta k) = |\mathcal{V}_g f(\alpha l, \beta k)|^2 = |\mathcal{F}(f \cdot T_{\alpha l} g)(\beta k)|^2.$$

The proof is based on the observation that the mel-spectrogram can be written via the operation of so-called *STFT- or Gabor multipliers*, cf. [17], on any given function in the sense of a bilinear form. Before deriving the involved correspondence, we thus introduce this important class of operators.

Given a window function g, time- and frequency-sub-sampling parameters α, β, respectively, and a function $\mathbf{m} : \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{C}$, the corresponding Gabor multiplier $G^{\alpha,\beta}_{g,\mathbf{m}}$ is defined as

$$G^{\alpha,\beta}_{g,\mathbf{m}} f = \sum_k \sum_l \mathbf{m}(k,l) \langle f, M_{\beta k} T_{\alpha l} g \rangle M_{\beta k} T_{\alpha l} g.$$

We next derive the expression of a mel-spectrogram by an appropriately chosen Gabor multiplier. Using sub-sampling factors α in time and β in frequency as before, we start from (4) and reformulate as follows:

$$\begin{aligned}
\mathrm{MS}_g(f)(b,v) &= \sum_k |\mathcal{F}(f \cdot T_b g)(\beta k)|^2 \cdot \Lambda_v(\beta k) \\
&= \sum_k \langle f, M_{\beta k} T_b g \rangle \overline{\langle f, M_{\beta k} T_b g \rangle} \Lambda_v(\beta k) \\
&= \left\langle \sum_k \Lambda_v(\beta k) \langle f, M_{\beta k} T_b g \rangle M_{\beta k} T_b g, f \right\rangle \\
&= \left\langle \sum_k \sum_l \mathbf{m}(k,l) \langle f, M_{\beta k} T_{\alpha l} g \rangle M_{\beta k} T_{\alpha l} g, f \right\rangle
\end{aligned}$$

with $\mathbf{m}(k,l) = \delta(\alpha l - b) \Lambda_v(\beta k)$. We see that the mel-coefficients can thus be interpreted via a Gabor multiplier: $\mathrm{MS}_g(f)(b,v) = \langle G^{\alpha,\beta}_{g,\mathbf{m}} f, f \rangle$.

The next step is to switch to an alternative operator representation. Indeed, as shown in [16], every operator H can equally be written by means of its *spreading function* η_H as

$$Hf(t) = \int_x \int_\xi \eta_H(x,\xi) f(t-x) e^{2\pi i t\xi} \mathrm{d}\xi \mathrm{d}x. \tag{12}$$

We note that two operators H_1, H_2 are equal if and only if their spreading functions coincide, see [15, 16] for details.

As shown in [15], a Gabor multiplier's spreading function $\eta^{\alpha,\beta}_{g,\mathbf{m}}$ is given by

$$\eta^{\alpha,\beta}_{g,\mathbf{m}}(x,\xi) = \mathcal{M}(x,\xi)\mathcal{V}_g g(x,\xi), \tag{13}$$

where $\mathcal{M}(x,\xi)$ denotes the (β^{-1},α^{-1})-periodic symplectic Fourier transform of $\mathbf{m}$, i.e.,

$$\mathcal{M}(x,\xi) = \mathcal{F}_s(\mathbf{m})(x,\xi) = \sum_k \sum_l \mathbf{m}(k,l) e^{-2\pi i(\alpha l\xi - \beta k x)}. \tag{14}$$

We now equally rewrite the time-averaging operation applied to a filtered signal, as defined in (6), as a Gabor multiplier. As before, we set $\check{h}_v(t) = \overline{h_v(-t)}$ and have

$$\begin{aligned}
\mathrm{FB}_{h_v}(f)(b,v) &= \sum_l |(f * h_v)(\alpha l)|^2 \cdot \varpi_v(\alpha l - b) = \\
&\sum_l |\sum_n f(n)\check{h}_v(n-\alpha l)|^2 \cdot \varpi_v(\alpha l - b) \\
&= \sum_k \sum_l |\langle f, M_{\beta k} T_{\alpha l} \check{h}_v\rangle|^2 \cdot \varpi_v(\alpha l - b)\delta(\beta k) \\
&= \langle G^{\alpha,\beta}_{\check{h}_v,\mathbf{m}_F} f, f\rangle.
\end{aligned}$$

with $\mathbf{m}_F(k,l) = T_b \varpi_v(l)\delta(\beta k)$. To obtain the error estimate in Corollary 1, first note that by straightforward computation using the operators' representation by their spreading functions as in (12)

$$\begin{aligned}
|\mathrm{MS}_g(f)(b,v) - \mathrm{FB}_{h_v}(f)(b,v)| &= \left|\left\langle \left(G^{\alpha,\beta}_{g,\mathbf{m}} - G^{\alpha,\beta}_{\check{h}_v,\mathbf{m}_F}\right) f, f\right\rangle\right| \\
= \left|\left\langle \left(\eta_{g^\beta_\alpha,\mathbf{m}} - \eta_{\check{h}^\beta_{\alpha v},\mathbf{m}_F}\right), \mathcal{V}_f f\right\rangle\right| &\le \left\|\eta^{\alpha,\beta}_{g,\mathbf{m}} - \eta^{\alpha,\beta}_{\check{h}_v,\mathbf{m}_F}\right\| \cdot \|f\|_2^2
\end{aligned}$$

and we can estimate the error by the difference of the spreading functions. We write the sampled version of Λ_v by using the Dirac comb Ш_β: $\Lambda_v(\beta k) = (\text{Ш}_\beta \Lambda_v)(t) = \sum_k \Lambda_v(t)\delta(t-\beta k)$ and analogously for ϖ_v using Ш_α to obtain $\mathbf{m} = T_b\delta(\alpha l)\cdot\text{Ш}_\beta\Lambda_v$ and $\mathbf{m}_F = \text{Ш}_\alpha T_b \varpi_v \cdot \delta(\beta k)$. Applying the symplectic Fourier transform (14) to $\mathbf{m}$ then gives:

$$\begin{aligned}
\mathcal{M}^\nu(x,\xi) &= \sum_k \sum_l \mathbf{m}(k,l) e^{-2\pi i(\alpha l\xi - \beta k x)} \\
&= \int_t \sum_k \Lambda_\nu(t)\delta(t-\beta k) e^{2\pi i t x}\mathrm{d}t \sum_l T_b\delta(\alpha l) e^{-2\pi i \alpha l \xi} \\
&= \mathcal{F}^{-1}(\text{Ш}_\beta \Lambda_\nu)(x)\cdot e^{-2\pi i b\xi}
\end{aligned}$$

Now it is a well-known fact that the Fourier transform turns sampling with sampling interval β into periodization

$$\mathcal{F}^{-1}(\text{Ш}_\beta \Lambda_\nu)(x) = \text{Ш}_{\frac{1}{\beta}} * \mathcal{F}^{-1}(\Lambda_\nu)(x) = \sum_l T_{\frac{l}{\beta}} \mathcal{F}^{-1}(\Lambda_\nu)(x),$$

by $1/\beta$, in other words, into a convolution with $\text{Ш}_{\frac{1}{\beta}}$:

hence

$$\mathcal{M}^\nu(x,\xi) = \sum_l T_{\frac{l}{\beta}} \mathcal{F}^{-1}(\Lambda_\nu)(x)\cdot e^{-2\pi i b\xi}.$$

Completely analogous considerations for ϖ_v and Ш_α lead to the periodization of $\mathcal{F}(\varpi_v)$ and thus the following expression for the symplectic Fourier transform of $\mathbf{m}_F$:

$$\mathcal{M}^\nu_F(x,\xi) = \sum_l T_{\frac{l}{\alpha}} \mathcal{F}(\varpi_v)(\xi)\cdot e^{-2\pi i b\xi}.$$

Plugging these expressions into (13) gives the bound (8).

Remark 5 It is interesting to interpret the action of an operator in terms of its spreading function. In view of (12), we see that the spreading function determines the amount of shift in time and frequency, which the action of the operator imposes on a function. For Gabor multipliers, if well-concentrated window functions are used, it is immediately obvious that the amount of shifting is moderate as well as determined by the window's eccentricity. At the same time, the aliasing effects introduced by coarse sub-sampling are reflected in the periodic nature of $\mathcal{M}$. Since, for $\mathcal{F}^{-1}(\Lambda_v)$ the sub-sampling density in frequency, determined by β, and for $\mathcal{F}(\varpi_v)$ the sub-sampling density in time, determined by α, determine the amount of aliasing, the overall approximation quality deteriorates with increasing sub-sampling factors.

References

1. Abreu LD, Romero JL (2017) MSE estimates for multitaper spectral estimation and off-grid compressive sensing. IEEE Trans Inf Theory 63(12):7770–7776
2. Andén J, Mallat S (2014) Deep scattering spectrum. IEEE Trans Signal Process 62(16):4114–4128
3. Anselmi F, Leibo JZ, Rosasco L, Mutch J, Tacchetti A, Poggio TA (2013) Unsupervised learning of invariant representations in hierarchical architectures. CoRR arxiv:1311.4158
4. Balazs P, Dörfler M, Jaillet F, Holighaus N, Velasco G (2011) Theory, implementation and applications of nonstationary gabor frames. J Comput Appl Math 236(6):1481–1496
5. Balazs P, Dörfler M, Kowalski M, Torrésani B (2013) Adapted and adaptive linear time-frequency representations: a synthesis point of view. IEEE Signal Process Mag 30(6):20–31
6. Bammer R, Dörfler M (2017) Invariance and stability of Gabor scattering for music signals. In: Sampling theory and applications (SampTA), 2017 international conference on. IEEE, pp 299–302

7. Boulanger-Lewandowski N, Bengio Y, Vincent P (2012) Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. arXiv preprint arXiv:1206.6392
8. Choi K, Fazekas G, Sandler M, Cho K (2018) The effects of noisy labels on deep convolutional neural networks for music tagging. IEEE Trans Emerg Top Comput Intell 2(2):139–149
9. Choi K, Fazekas G, Sandler M (2016) Automatic tagging using deep convolutional neural networks. In: Proceddings of the 17th international society for music information retrieval conference
10. Chuan CH, Herremans D (2018) Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In: Thirty-second AAAI conference on artificial intelligence
11. Dieleman S, Brakel P, Schrauwen B (2011) Audio-based music classification with a pretrained convolutional network. In: 12th international society for music information retrieval conference (ISMIR-2011). University of Miami, pp 669–674
12. Dieleman S, Schrauwen B (2014) End-to-end learning for music audio. In: Acoustics, speech and signal processing (ICASSP), 2014 IEEE international conference on, pp 6964–6968. https://doi.org/10.1109/ICASSP.2014.6854950
13. Dörfler M (2001) Time-frequency analysis for music signals: a mathematical approach. J New Music Res 30(1):3–12
14. Dörfler M, Bammer R, Grill T (2017) Inside the spectrogram: convolutional neural networks in audio processing. In: International conference on sampling theory and applications (SampTA). IEEE, pp 152–155
15. Dörfler M, Torrésani B (2010) Representation of operators in the time-frequency domain and generalized Gabor multipliers. J Fourier Anal Appl 16(2):261–293
16. Feichtinger HG, Kozek W (1998) Quantization of TF lattice-invariant operators on elementary LCA groups. In: Feichtinger HG, Strohmer T (eds) Gabor analysis and algorithms, applied and numerical harmonic analysis. Birkhäuser, Boston, pp 233–266
17. Feichtinger HG, Nowak K (2003) A first survey of Gabor multipliers. In: Feichtinger HG, Strohmer T (eds) Advances in Gabor analysis, applied and numerical harmonic analysis. Birkhäuser, Boston, pp 99–128
18. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
19. Grill T, Schlüter J (2015) Music boundary detection using neural networks on combined features and two-level annotations. In: Proceedings of the 16th international society for music information retrieval conference (ISMIR 2015). Malaga, Spain, pp 531–537
20. Grohs P, Wiatowski T, Bölcskei H (2016) Deep convolutional neural networks on cartoon functions. In: Information theory (ISIT), 2016 IEEE international symposium on. IEEE, pp 1163–1167
21. Holighaus N, Dörfler M, Velasco GA, Grill T (2013) A framework for invertible, real-time constant-Q transforms. IEEE Trans Audio Speech Lang Process 21(4):775–785
22. Humphrey EJ, Bello JP (2012) Rethinking automatic chord recognition with convolutional neural networks. In: Machine learning and applications (ICMLA), 2012 11th international conference on. IEEE, vol 2, pp 357–362
23. Humphrey EJ, Montecchio N, Bittner R, Jansson A, Jehan T (2017) 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
24. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the 6th international conference on learning representations (ICLR). San Diego, USA
25. Korzeniowski F, Widmer G (2016) A fully convolutional deep auditory model for musical chord recognition. In: Machine learning for signal processing (MLSP), 2016 IEEE 26th international workshop on. IEEE, pp 1–6
26. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
27. Lee H, Pham P, Largman Y, Ng AY (2009) Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in neural information processing systems, pp 1096–1104
28. Leglaive S, Hennequin R, Badeau R (2015) Singing voice detection with deep recurrent neural networks. In: Acoustics, speech and signal processing (ICASSP), 2015 IEEE international conference on. IEEE, pp 121–125
29. Lehner B, Schlüter J, Widmer G (2018) Online, loudness-invariant vocal detection in mixed music signals. IEEE/ACM Trans Audio Speech Lang Process 26(8):1369–1380
30. Malik M, Adavanne S, Drossos K, Virtanen T, Ticha D, Jarina R (2017) Stacked convolutional and recurrent neural networks for music emotion recognition. arXiv preprint arXiv:1706.02292
31. Mallat S (2012) Group invariant scattering. Commun Pure Appl Math 65(10):1331–1398
32. Mallat S (2016) Understanding deep convolutional networks. Philos Trans R Soc Lond A Math Phys Eng Sci 374(2065). https://doi.org/10.1098/rsta.2015.0203. URL http://rsta.royalsocietypublishing.org/content/374/2065/20150203
33. Schlüter J, Böck S (2013) Musical onset detection with convolutional neural networks. In: 6th international workshop on machine learning and music (MML), Prague, Czech Republic
34. Schlüter J, Böck S (2014) Improved musical onset detection with convolutional neural networks. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP 2014). Florence, Italy
35. Schlüter J, Grill T (2015) Exploring data augmentation for improved singing voice detection with neural networks. In: Proceedings of the 16th international society for music information retrieval conference (ISMIR 2015). Malaga, Spain
36. Ullrich K, Schlüter J, Grill T (2014) Boundary detection in music structure analysis using convolutional neural networks. In: Proceedings of the 15th international society for music information retrieval conference (ISMIR 2014). Taipei, Taiwan
37. Waldspurger I (2015) Wavelet transform modulus: phase retrieval and scattering. Ph.D. thesis, Ecole normale supérieure-ENS PARIS
38. Waldspurger I (2017) Exponential decay of scattering coefficients. In: 2017 international conference on sampling theory and applications (SampTA), pp 143–146. https://doi.org/10.1109/SAMPTA.2017.8024473
39. Wiatowski T, Grohs P, Bölcskei H (2017) Energy propagation in deep convolutional neural networks. arXiv preprint arXiv:1704.03636
40. Wiatowski T, Tschannen M, Stanic A, Grohs P, Bölcskei H (2016) Discrete deep feature extraction: a theory and new architectures. In: Proceedings of the international conference on machine learning, pp 2149–2158