Sparse Approximations for Drum Sound Classification

Simon Scholler and Hendrik Purwins, *Member, IEEE*

***Abstract*—Up to now, there has only been little work on using features from temporal approximations of signals for audio recognition. Time–frequency tradeoffs are an important issue in signal processing; sparse representations using overcomplete dictionaries may (or may not, depending on the dictionary) have more time–frequency flexibility than standard short-time Fourier transform. Also, the precise temporal structure of signals cannot be captured by spectral-based feature methods. Here, we present a biologically inspired three-step process for audio classification: 1) Efficient atomic functions are learned in an unsupervised manner on mixtures of percussion sounds (drum phrases), optimizing the length as well as the shape of the atoms. 2) An analog spike model is used to sparsely approximate percussion sound signals (bass drum, snare drum, hi-hat). The spike model consists of temporally shifted versions of the learned atomic functions, each having a precise temporal position and amplitude. To obtain the decomposition given a set of atomic functions, matching pursuit is used. 3) Features are extracted from the resulting spike representation of the signal. The classification accuracy of our method using a support vector machine (SVM) in a 3-class database transfer task is 87.8%. Using gammatone functions instead of the learned sparse functions yields an even better classification rate of 97.6%. Testing the features on sounds containing additive white Gaussian noise reveals that sparse approximation features are far more robust to such distortions than our benchmark feature set of timbre descriptor (TD) features.**

***Index Terms*—Dictionary learning, machine listening, matching pursuit, sound classification, sparse approximation, spike coding, unsupervised learning.**

I. Introduction

The large majority of popular features used in audio classification are spectral methods, such as Mel-frequency spectral coefficients (MFCCs), spectral moments, or band-energy ratio. However, as these features require the calculation of a spectrogram, they suffer from a fixed tradeoff between frequency and time resolution. Also, phase information which is crucial for consonant recognition and auditory grouping of sounds [1] gets lost when calculating only features based on the magnitude spectrum. Sparse temporal approximations have become increasingly popular in the last decade and efforts have been made to use such models for audio recognition ([2], [3]).

A code is considered to be sparse if only few units of the total population are needed to encode (or represent) a signal with sufficient accuracy. The basic idea of modeling sparse coding [4] is simple: Given a set of input signals $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, the aim is to find a number of atomic functions $\phi_1, \phi_2, \ldots, \phi_m$, such that each input signal can be approximated sparsely by a linear combination of a relatively small number of atomic functions (also called atoms):

$$\mathbf{x}_i = \sum_{j=1}^{m} s_j \phi_j = \Phi \mathbf{s} \quad \forall i = 1, 2, \ldots, n \tag{1}$$

where the activity vector $\mathbf{s}$ is sparse, i.e., units are either active or inactive (not significantly different from zero), where the fraction of active units is generally quite small given the total number of units. The complete set of the elementary atomic functions is called a dictionary. In sparse coding, overcomplete dictionaries are normally used. In overcomplete dictionaries, the number of atomic functions is higher than the dimensionality of the signal such that a subset of them can span the whole signal space.

This paper is organized as follows. First, we discuss previous work on sparse coding and drum transcription. In Section II, we explain matching pursuit as well as gradient ascent and a shrinkage/extension heuristics for dictionary learning. Then we introduce spike counts and spike amplitudes for the learned as well as for the predefined gammatone dictionary. We cover timbre features and support vector machine classification. In Section III, we compare classification performance under different noise conditions and visualize the specificity of the most prominent atoms. Finally, we discuss the results and give an outlook.

A. Benefits of Sparse Coding

Sparse coding is particularly useful in hierarchical networks such as the brain because the representations are easier to read out by subsequent (higher) levels as compared to other codes. This eases the communication between the areas and (statistical) inference at subsequent processing stages.

Classifying data based on the sparse coding output is therefore a promising approach and will be the main objective of this paper. Although we only will deal with drum data due to its relative simplicity, this is a general approach that can in principle be applied to arbitrary kinds of sound data.

Manuscript received October 15, 2010; revised January 21, 2011, April 17, 2011; accepted June 09, 2011. Date of publication July 05, 2011; date of current version August 17, 2011. The work of S. Scholler was supported by a scholarship from the German Academic Exchange Service (DAAD). The work of H. Purwins was supported by a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jalal Fadili.

S. Scholler was with the Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain. He is now with the Machine Learning Laboratory, Berlin Institute of Technology, 10115 Berlin, Germany (e-mail: simon.scholler@bccn-berlin.de).

H. Purwins is with the Music Technology Group, Universitat Pompeu Fabra, 08002 Barcelona, Spain (e-mail: hendrik.purwins@upf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2011.2161264

B. Previous Work

1) Fixed Versus Learned Dictionaries: The reservoir of atomic functions (dictionary) can either be predefined or learned. Complete and overcomplete dictionaries can be built using the following frameworks: Fourier, Wavelet (e.g., Haar, Daubechies, Gabor), principal component analysis (PCA), and independent component analysis.

Several methods have been developed to learn sparse atomic functions [5]–[9], based on optimizing the tradeoff between minimizing the reconstruction error $||\mathbf{x}-\mathbf{\Phi s}||^2$ and maximizing a sparseness cost function, e.g., $-\sum_{j=1}^{m}|s_j|^p(0<p\leq 1)$. The resulting dictionary can represent the input signals sparsely, i.e., only few functions are needed to describe a signal. For natural images, Lewicki and Olshausen [10] showed that learned dictionaries have a higher coding efficiency than traditional complete Fourier and wavelet bases. Kreutz-Delgado *et al.* [5] provided a dictionary learning algorithm that simultaneously updates the dictionary as well as the amplitude coefficients. They show that using their method, a learnt overcomplete dictionary is more efficient than a complete one with the same reconstruction error on natural images. Since optimizing a cost function with a sparseness constraint is computationally expensive, learning is normally done on small image patches of 8×8 or 16×16 pixels. Single audio events such as a hi-hat sound can have durations of more than half a second. Given a sampling frequency of 16 000 Hz, this would result in more than 8000 dimensions. For this reason, we employed the sparse optimization method of [4] which uses matching pursuit for building the signal approximation. Besides being computationally feasible, it has a number of advantages compared to other approaches: In [1], Smith and Lewicki compare various methods to perform a sparse approximation according to (2): maximum *a posteriori* (MAP) optimization [11], filter threshold, matching pursuit (Section II-A) and hybrid optimization that uses MAP after an initialization by filter threshold or matching pursuit.

Whereas MAP alone is computationally too costly for reasonably long atoms to be applied in its pure form, the hybrid optimization is feasible. In an experiment with speech segments, music and environmental sounds, for spike rates of less than 250 spikes per second as in our experiments, the SNR for matching pursuit is by a large margin better than for filter threshold and its optimization.

2) Sparse Methods in Audio Classification: Adiloğlu *et al.* ([12], [13]) decomposed sounds in terms of a predefined dictionary of Gammatone functions (Section II-D2). Applying matching pursuit [1], they yielded a point pattern of time–frequency components. They then defined a dissimilarity of two sounds by introducing a point correspondence that assigns each point of one sound to exactly one point of the other sound. The comparison measure is not very robust with respect to nonlinearly stretching of the point pattern of the original sound. On everyday sound samples, they yielded an average classification performance in the range of MFCCs, when using a large (192) number of spikes, which is computationally very expensive, since the number of possible mappings between N spikes explodes combinatorically for each pair of sounds. Their method operates with fixed length atomic functions. Sturm *et al.* [14] used MFCCs over several pre-selected time scales. In our approach, we focus on adapting the atom length to the signal type, without considering the temporal pattern generated by sequences of atoms.

In the sound recognition literature, the sparse optimization model of [4] has been used for music genre recognition by [2]. Short pieces of music were sparsely approximated using the learnt atomic functions and features from the resulting spike representation were computed. The classification performance, however, was lower compared to classifying with the widely applied MFCCs. Also, the results using the learned atomic functions were not as good as using a dictionary of gammatone atoms of the same size. Their results however increased slightly when using the combined set of sparse approximation features and MFCCs.

A similar approach was taken by [15]. They used features obtained by sparsely approximating audio files with a dictionary of gammatone atoms. Using these features alone, the results on classifying environmental ambient sounds were lower than using MFCCs, but the performance of the combined feature set was significantly higher than using MFCCs alone. Another approach has been suggested by [16].

3) Drum Transcription: There are several approaches used for drum transcription [17], [18]. In [19], an atomic spectral envelope and a temporal envelope of the gain was learned for each instrument, assuming the statistical independence of the spectral and the temporal envelope. These atoms were learnt by non-negative matrix factorization on a database of isolated instrument sounds. In contrast, we will learn temporal atomic functions from sound mixtures. In [19], with spectral atoms at hand, a procedure is introduced to iteratively learn the coefficients for a linear decomposition of the sound spectra for each frame, in terms of the atomic spectral shapes. Then, high deltas between consecutive coefficients of the same instrument spectrum indicate an onset of this instrument.

Since in our approach we classify isolated drum sounds, we chose timbre descriptors (TDs) outlined in [20] as a reference to compare our results with. In Section II-D3, we will explain this method further.

4) Novel Content of This Paper: This paper is a further development and a more elaborated treatment of ideas presented in a previous shorter workshop paper [21]. This paper contains several new aspects. 1) The procedure to estimate the length of the atomic functions is detailed. 2) Classification results are compared using an additional representation (timbre descriptors). 3) Two additional experiments are performed studying the impact of noise on the classification performance using different representations. 4) Spike amplitude and spike count representation are compared. 5) An in-depth discussion of the method and the results is provided in the light of previous research.

II. Methods

A. Matching Pursuit

Matching Pursuit (MP) [22] is a method to derive a sparse approximation of a signal using a dictionary of atomic functions. Choosing MP has practical considerations: Using a linear model, the problem of finding the best approximation using k out of a total of N atomic functions is NP-hard [23]. All combinations $\binom{N}{k}$ must be taken into consideration in order to find the best solution. Since this is computationally unbearable for most real-world problems, alternative solutions like MP have been proposed in order to find reasonably good solutions using few atomic functions. Those are not optimal in general but greatly reduce the computational costs.

MP is an iterative method. In each step, the atomic functions of the dictionary are correlated with the signal and the atomic function with the highest correlation is subtracted from the (residual) signal. This process is repeated until a stopping criterion is reached.

B. Learning Sparse Atomic Functions

In [4], natural sounds were encoded through a sum of shiftable and scalable atomic functions, i.e., an (analog) spike-code model. In an iterative process, the atomic functions are used to approximate the signal. Subsequently, their shape and length are updated in order to sparsely encode the data. The resulting atomic functions showed a close resemblance to the response properties of auditory nerve fibers and their distribution in the frequency-bandwidth plane roughly matched the distribution found by experimental studies in the cat.

We employed this sparse coding optimization method of [4] to obtain sparse atomic functions for a dataset of drum phrases.

The optimization uses a linear signal model. A signal is described as a linear superposition of a number of shiftable atomic functions ϕ_n, each having an associated temporal location τ^n and amplitude a^n. The length of the atomic functions is only restricted by the length of the signal. Further, each atomic function can appear multiple times at different time points τ_i^n. Formally, this yields the following decomposition:

$$x(t) = \hat{x}(t) + \epsilon(t) = \sum_{n=1}^{N} \sum_{i=1}^{c_n} a_i^n \phi_n \left(t - \tau_i^n\right) + \epsilon(t) \qquad (2)$$

where N is the total number of atomic functions (the size of the dictionary), c_n is the total number the atomic function ϕ_n is used in the current decomposition and ϵ represents the residual.

After initializing the atomic functions randomly with a normal distribution, learning is done in an iterative manner. One learning iteration consists of two steps: In the first step, the signal is decomposed with Matching Pursuit. In the second step, the atomic functions used in the MP approximation get updated.

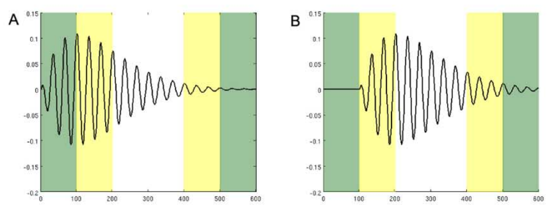
Whereas other authors employed signal-to-residual ratio or a fixed number of iterations as a stopping criterion, we used a "spiking threshold," i.e., the MP decomposition was stopped when the correlation of a newly picked atom fell below 0.2. As the spiking threshold is sensitive to the scale of the signal, all sounds were normalized to a maximum amplitude of 1.

Learning of the atomic functions was done by gradient ascent. The update rule is

$$\phi_n^{t+1} = \phi_n^t + \lambda \sum_i \hat{a}_i^n [x - \hat{x}]_{\tau_i^n} \qquad (3)$$

i.e., each used atomic function gets updated according to the decomposition residual $[x - \hat{x}]$ over the temporal extent of the atomic function. The learning rate λ was set to $\lambda = 0.005$ in our case. Note that the value of the learning rate is highly dependent on the stopping criterion of MP because it influences the decomposition error and thereby has a direct impact on the learning rule. Learning of the atomic function is activity based, i.e., after each decomposition of a sound, only those atomic functions used in the MP-decomposition are updated. After each update step, the atomic functions vectors were normalized to unit ℓ^2-norm. The size of the atoms was kept variable in order to allow atomic functions with low frequency components to grow over the initial length and atomic functions with high frequencies to shrink (cf. Box 1). Each atom was initialized as a 100-dimensional vector with random Gaussian white noise. The first 10% on both ends of the vector were zero-padded.

Eight different atomic functions are used for drum recognition. The number of atomic functions has to be set manually. We have tested the algorithm with different dictionary sizes (4/8/16/32 atomic functions). Results increased slightly when using more atomic functions. This seems to indicate that for more complex classification tasks with more classes, it is likely that more atomic functions are needed.



Box 1: Atomic function with extension and shrinkage regions before and after updating its length (left and right figure, respectively). **A.** Atomic function before update. If the norm of the extension region (green) exceeds the extension threshold θ_e (left side), the atomic function grows. Similarly, if the norm of the shrinkage region (yellow) falls below the shrinkage threshold θ_s (right side), the atomic functions shrinks. When shrinking, the extension region is removed and the old shrinkage region becomes the new extension region. When growing, the extension region becomes the new shrinkage region and a new extension region (initialized with zeros) is appended. Due to this procedure, the thresholds θ_s and θ_e can be set in a reasonable manner such that the size-variation process is stabilized and rapid fluctuations between shrinkage and growth are minimized. If e.g. $\theta_s = \frac{\theta_e}{2}$ and the shrinkage region falls below θ_s, the shrinkage region becomes the new extension region. In order for the extension region to grow again, its norm must roughly double since $\theta_e = 2\theta_s$. Likewise, after an extension, the norm of the new shrinkage region (i.e. the old extension region) must halve in value in order to shrink. **B.** Atomic function after update. In this case, the atomic function was extended on the left side and shrank on the right side.
In our simulations, thresholds were set to $\theta_e = 0.0125$ and $\theta_s = \frac{\theta_e}{2} = 0.00625$.

For the unsupervised learning of the atomic functions, 2000 iterations over 134 drum phrases (duration: 8–33 s, average: 17 s) from the ENST drum database [24] have been performed. Each decomposition on average consisted of more than 750 atoms.

C. Data

We used isolated drum data samples from two databases, the ENST database [24] and the RWC musical instrument database [25]. All samples were normalized to a maximum amplitude of 1. The following datasets were created.

- BSH: A subset of isolated drum sounds from the ENST drum database: bass drum ("bd," 60 sounds), snare drum ("sd," 68 sounds), closed hi-hat ("chh," 46 sounds), open hi-hat ("ohh," 62 sounds). The hi-hat sounds were merged into one hi-hat class ("hh," 108 sounds) which by convention is done in most papers in the drum recognition literature.
- AMIX: Artificial pairwise mixtures of the classes "bd," "sd," and "hh" from BSH (50 samples each) and the single classes were used. We chose three energy ratios (0.5, 1, 2) for mixing the isolated sound classes, making a total of 12 classes. Sounds were mixed with a random offset up to a maximum distance of 50 ms between their amplitude peaks. We used artificial mixtures instead of real ones since this enabled us to vary the energy ratio of the two sounds in a controlled manner. Learning and test data base are not built from separated subsets of isolated sounds. So eventually it could happen that if sample x from "bd" and y from "hh" have been mixed to create AMIX, we can have $x+y$ in the learning set and $x+2\dot{y}$ in the test set.
- DBT: Database transfer data. Classes "bd," "sd," "hh" from BSH are used as training data, and testing was done on isolated drum samples ("bd," 'sd," "hh") taken from the RWC music instrument database (12/34/36 samples, respectively).

For drum sounds having a relatively stable and peaky frequency spectrum across samples, only few atomic functions need to be learned compared to music, speech, or most environmental sounds. Drum data is thus perfect for studying the general applicability of sparse coding for audio recognition within a small and manageable recognition problem.

To evaluate the classification performance under noisy conditions, sets with different amounts of additive noise were created from the datasets described above. Five different signal-to-noise ratios [-10 dB, 0 dB, 10 dB, 20 dB, ∞ dB (clean signal)] were used. Example of four different sounds and their noise samples are depicted in Fig. 1.

D. Features

1) MP Features Using a Sparse Coding Dictionary (SC-MP Features): The SC-MP features were obtained by sparsely approximating each drum sound file with Matching Pursuit using the learned sparse coding dictionary.

The feature vector consists of two parts: The first part is a vector containing summed coefficients ("spike amplitudes") of each dictionary atom in the sound decomposition normalized to a maximum absolute value of 1. The following part is a vector containing the total number each atom has been used ("spike counts"), equally normalized to a maximum absolute value of 1. The spike count features can be interpreted as the mean firing frequency of the atomic function over the whole signal. The amplitude vector was normalized. This is based on the notion that the recognition of a sound is normally not done by means

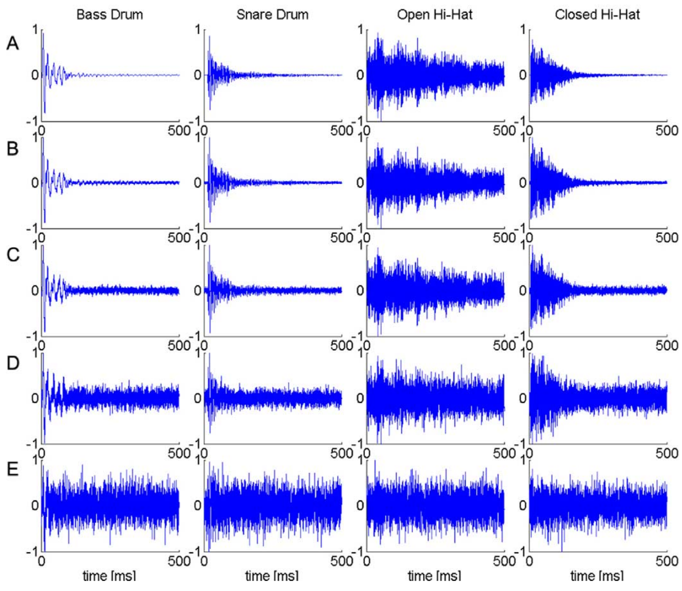


Fig. 1. Example percussion samples with different amounts of additive Gaussian white noise. **A**. $\mathrm{SNR}=\infty$ dB **B**. $\mathrm{SNR}=20$ dB **C**. $\mathrm{SNR}=10$ dB **D**. $\mathrm{SNR}=0$ dB **E**. $\mathrm{SNR}=-10$ dB.

of its signal energy but via characteristic features such as its temporal structure, temporal correlations, or spectral subband correlations. Normalizing the amplitude feature vector makes the classification invariant to the signal energy and only relative contributions of the atomic functions to the signal approximations are retained.

2) MP Features Using a Gammatone Dictionary (GT-MP Features): In this case, MP features using a gammatone dictionary of the same size as the SC-MP dictionary were used. Gammatone functions are a product of a gamma distribution and a sinusoidal. Gammatone filterbanks are widely used to mimic the structure of the peripheral auditory processing stage. They model the cochlea by a bank of overlapping bandpass filters. We used gammatones from such a filterbank as atomic functions, i.e., gammatones equally spaced on the equivalent-rectangular bandwidth (ERB) scale (cf. [26]). The main reason for introducing GT-MP features is that they can be directly compared to SC-MP and thus serve as a test whether a generic dictionary suffices or whether the computationally expensive preprocessing step of learning sparse atomic functions specialized to the data is necessary to achieve a high classification performance.

3) Timbre Descriptors (TD Features): We compared our approach with timbre descriptor features used in [20], a widely referenced method for classification on isolated drum samples. Different features are calculated for the attack and decay part of the drum signal. Features for the attack part include *Attack Energy*, *Temporal Centroid*, *Log-Attack Time*, *Zero Crossing Rate*. The large majority of features are extracted from the decay part, e.g., *Spectral Flatness*, *Spectral Centroid*, *Spectral Kurtosis*, *Zero Crossing Rate*, *Zero Crossing Rate Variance*, *Skewness*, and *Relative Energy* in a number of different frequency subbands. In addition, the mean and variance of 13 *MFCCs* are calculated for the whole signal. Additionally to applying the three classes of features individually (TD, SC-MP, GT-MP), the combinations of MP features with TD features were tested (TD+SC-MP, TD+GT-MP).

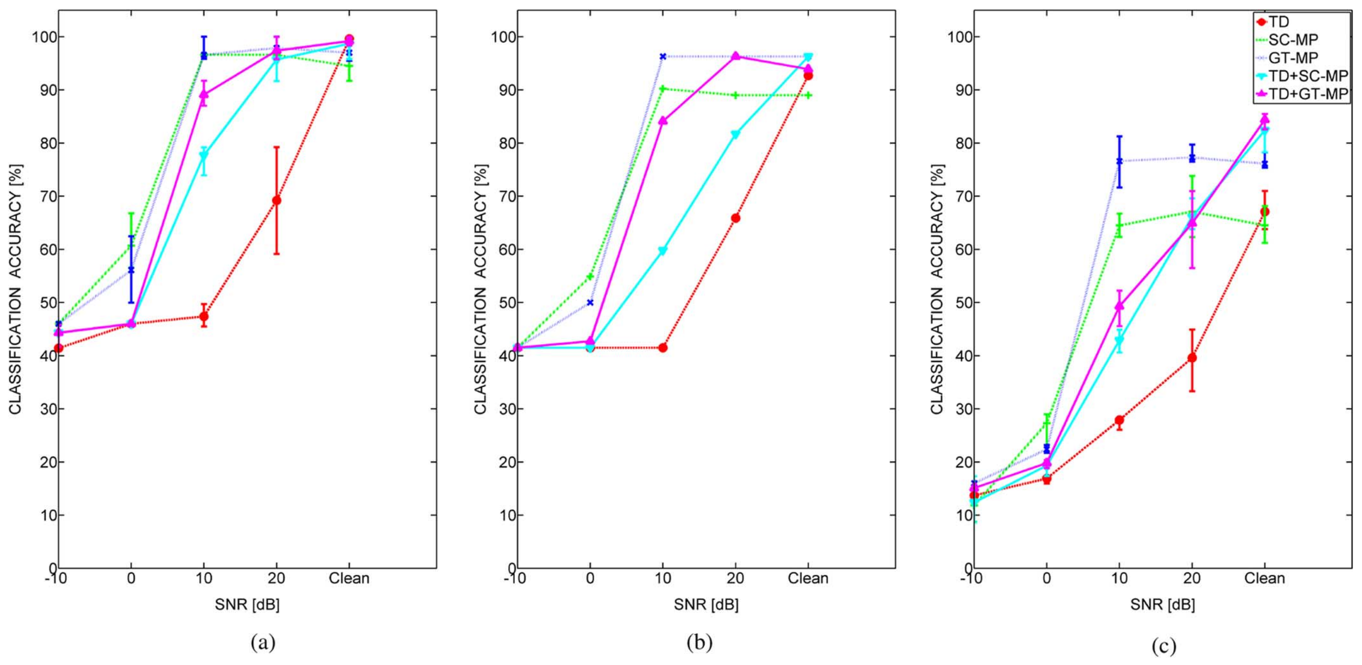


Fig. 2. Drum classification accuracy for different noise levels, datasets and feature combinations. The MP features were obtained using a dictionary of eight atoms. Training is done on clean data; testing is done on data with different SNRs (x-axis). Error bars mark the 25% and 75% percentiles of the cross-validation performance. Since the total number of samples differs between sound classes, the baseline performance is given by a classifier whose class output is always the class with the largest number of samples. The baseline performance thus is 46% for BSH, 44% for DBT and 16% for AMIX, (a) BSH, (b) DBT, and (c) AMIX.

E. *Classification*

Classification was done with a multi-class support vector machine using a radial-basis function (RBF) kernel [27]. The used LIBSVM package [28] implements a “one-against-one” approach to multi-class classification, i.e., a classifier is built for each pair of classes, making a total of $(k(k-1))/(2)$ binary classifiers. To classify, a voting strategy is used, i.e., a feature vector is assigned to the class which gets the most votes from all classifiers. Classification on the BSH and AMIX datasets were tested by ten-fold crossvalidation; for the database transfer, the BSH dataset was used for training and the DBT dataset for testing. A parameter search on the SVM parameter C and γ has been performed on each training set in order to optimize the classification accuracy.

III. Results

The classification results for the BSH dataset are shown in Fig. 2(a), for the database transfer in Fig. 2(b), and for the artificial mixtures in Fig. 2(c). MP features were obtained from a signal representation using a dictionary of eight atomic functions. Informal tests with different dictionary sizes showed that classification performance for the MP features increases with the number of atomic functions, especially when comparing eight to four atoms.

For the simple task of classifying clean drum sounds from the same database (BSH), all classifiers perform well (classification accuracies over 95%). For the database transfer, GT-MP features alone yielded the best results. When testing on clean sound samples on the AMIX dataset, the combination of GT-MP and TD features performs best.

When training on clean data and testing on noisy data, the classification performance of TDs drops rapidly. In contrast, the accuracy of MP features remains robust until a SNR of 10 dB for all datasets, whereas GT-MP features yield even better results than SC-MP features. The almost constant classification performance of MP features until 10-dB SNR might be due to the relative robustness of MP against noise (e.g., MP is also used successfully for signal denoising [29]). For TD-MP features, while overall performing better for the clean case, significant drops in performance can be already observed for SNRs of 10 and 20 dB.

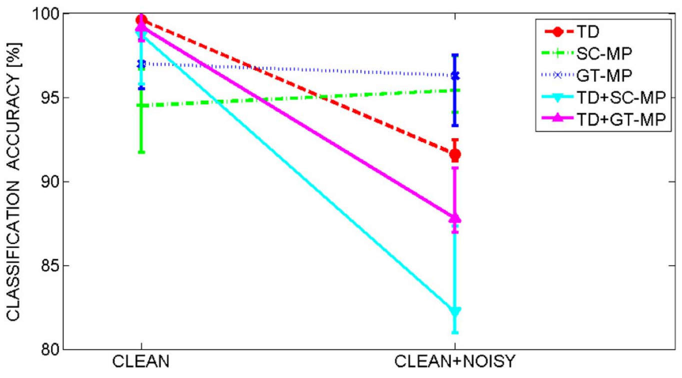


Fig. 3. Comparison of the classification accuracy of different features and feature combinations for two different cases: In first case, clean data is used in the training and in the test set. In the other case, a mixture of clean data and noisy data with different SNRs (−10 dB, 0 dB, 10 dB, 20 dB) is used in both training and test set. Error bars mark the 25% and 75% percentiles of the cross-validation performance.

In a second classification run, we used noisy and clean signals from dataset BSH in the training and test set. This allowed to test for cases in which the fluctuations in the test set are known in advance and can be already incorporated in the training set. Results for the different features and feature combinations are given in Fig. 3. While the isolated MP features stay constant, the performance of the feature classes containing TD features decreases in all cases when training and testing on clean data.

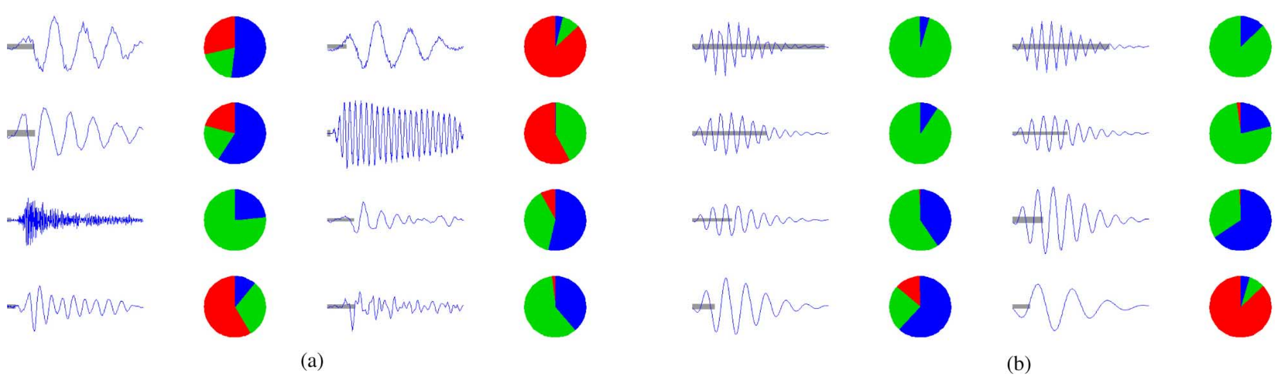


Fig. 4. Sparse Coding and Gammatone dictionary with eight atoms (upper and lower plot, respectively). The gray bars at the beginning of the atomic functions show a 5-ms interval. The pie charts depict the (relative) summed weights for the BSH dataset (red: "bd," blue: "sd," green: "hh"). (a) Sparse Coding dictionary. (b) Gammatone dictionary.

However, compared to training on clean and testing on noisy data, the performance of TD increased considerably although it did not reach the performance of MP features. Thus, in general, TD features seem to be useful only if training and test set are similar or if the distortions to be expected in the test set can be already incorporated in the training set. If the distortions are unknown in advance, MP features could be the better choice if the distortion type is comparable to the one used in our experiment.

Fig. 4 shows a sparse coding and a gammatone dictionary along with the relative use of the atomic functions for the three drum sound classes depicted as pie charts. The atomic functions most closely resemble gammatone functions except for two atomic functions, a sinusoidal and a high-frequency transient. As the pie charts reveal, some of the atomic functions are almost exclusively used in the decomposition of one sound class. This is in particular the case for the gammatone dictionary where high-frequency gammatones are primarily used for hi-hat sounds. With increasing frequency, the proportion of snare drum sounds increases. The gammatone with the lowest frequency is almost exclusively used for the bass drum sounds. This effect can be seen even more detailed in Fig. 5 which depicts the class-wise distributions (over sound samples) of the computed features for a sparse coding dictionary and a gammatone dictionary, respectively. The distributions over the features derived with the sparse coding dictionary show that some atomic functions seem to respond primarily to sounds of one class (e.g., sparse coding atoms with feature indices 2/10 and 5/13). This tendency is again more pronounced for the gammatone dictionary (e.g., atoms with feature indices 1/9, 7/15, 8/16) and is therefore consistent with the higher classification accuracies using a gammatone dictionary. The fact that this effect is also visible for the gammatone dictionary should not surprise as in particular bass and snare drum show single peak magnitude spectra for which gammatone functions with a similar central frequency will primarily be used [cf. Fig. 5(b)]. For instance, since only hi-hat sounds contain a considerable amount of high-frequency components, the high-frequency gammatones are primarily used for this class.

Notice also that for the gammatone dictionary, the spike count features of the specialized atoms seem to separate the classes more clearly than the amplitude features (Fig. 5). Using a binary

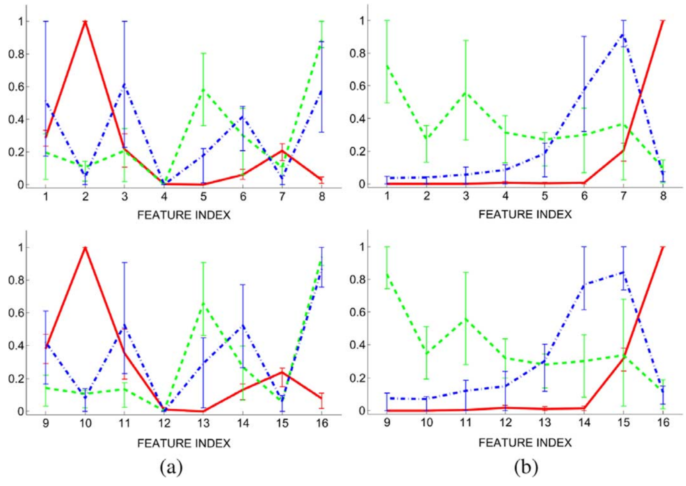


Fig. 5. Averaged MP feature values for classes "bd" (solid), "sd" (dashed-dotted) and "hh" (dashed) of dataset BSH. The error bars show the 25% and 75% percentiles. The first half of the feature vector (upper plot) refers to the spike amplitudes of the atomic functions, the second half (lower plot) to their respective spike counts (firing frequency). (a) SC-MP. (b) GT-MP.

spike model instead of an analog one as done in our case thus is unlikely to degrade the classification accuracy.

IV. Discussion

We have presented a biologically inspired approach to classify arbitrary sound classes by 1) learning sparse atomic functions on unlabeled data and 2) supervised learning of the classes using features derived from a temporal approximation of the signal.

This approach has certain advantages over other audio recognition methods. First, it is a general method that does not make any assumptions about the data except that the data can be sparsely approximated by a small number of atomic functions. In contrast to Fourier or wavelet analysis, no predefined atomic functions are used but the most efficient ones from the data are learned. Selecting those atomic functions that account for most of the energy leads to a sparse approximation of the signal. It remains to be investigated further to what degree sparsity aids classification. The presented approach can be used to segregate structured from noisy components. If the noise components contribute to the sound characteristics, the recognition system could be complemented by a spectro-temporal noise model.

However, the presented method also has some drawbacks. The number of atomic functions to be learned has to be set manually and it is not easy to determine what a reasonable number for a particular class of sounds would be.

In almost all cases where sounds are corrupted with an unknown level of white Gaussian noise, MP-based features clearly outperformed TD features. This is especially true for the difficult tasks, the database transfer and the artificial mixtures. If training and test set are homogenous, TD performs comparable to MP features. However, if training and test set differ (clean signals in the training set, noisy signals in the test set), MP-based methods are far more robust to white noise distortions than TDs.

Testing our algorithm on a variety of different distortions (e.g., pink noise or echoes) has yet to be done. Also, the mid-level temporal structure of a sound decomposition such as the temporal sequence of the atomic functions or the co-occurrances of two atomic functions is not yet considered in the feature extraction process. This mid-level temporal information however is often necessary as sounds mostly have a clearly defined temporal structure, e.g., many instruments consist of a characteristic initial transient followed by a decaying tonal component. A basic approach to include further information about the temporal structure is to calculate first- and second-order derivatives of the amplitude and function count vectors by windowing the signal and calculating a signal approximation for each window. Using simple time derivatives of the features has been shown to increase the performance considerably [30]. Spectro-temporal patterns of atomic functions can be compared by a one-to-one pattern mapping as in [12] or by dynamic time warping. Another approach consists in clustering coefficients and then comparing statistical models (e.g., Markov chains [31]) of cluster index sequences, trained with sound samples of particular instruments.

To our surprise, using a generic dictionary of gammatone functions led to higher classification accuracies than using a dictionary learned by sparse coding which is consistent with the results of a music-genre recognition study [2] using a similar approach. There might be several reasons why this is the case. Percussion sounds such as an open hi-hat or a snare drum do not have a clear temporal structure in the millisecond range which impedes finding efficient atomic functions to encode these structures. Also, the atomic functions that evolve during the learning process are chosen based on their contribution to the overall energy of the signal. As a result, only functions contributing much to the signal energy will evolve. These "high-energy" structures, however, might not be the ones that are important for distinguishing between different classes of sounds. In the sparse optimization on drum sounds, mainly functions in the lower frequency range evolved. In contrast, a generic dictionary of gammatone atoms is distributed over the whole frequency space which, at least for the drum recognition tasks in this study, was apparently beneficial for the classification. Using preemphasis to attenuate the low frequency range of the input sounds of the sparse coding learning optimization might be a way to obtain atomic functions that are more widely distributed in frequency space.

It remains an open task to compare the performance of our dictionary learning algorithm with others ([5]–[9], [32]). In a follow-up study, the authors plan to compare the performance of their approach to other sparse approximation methods [35] such as non-negative matrix factorization, which has previously been used for drum sound classification [19]. This will allow a more informative assessment of its performance comparing related sparse methods. Also it will be up to future work to use the algorithm to analyze other types of audio data, e.g., environmental sounds.

Whether to favour a generic dictionary or a dictionary adapted to the data is probably dependent on the recognition task in question. Tonal components of natural sounds are often characterized by a fast onset and a slow decay, which is why gammatones can efficiently represent a large variety of natural sounds (e.g., speech, cf. [4]). If the distinctive features of the classes are temporal characteristics on a very short time scale, learning a sparse coding dictionary adapted to the data is likely to perform better than a generic gammatone dictionary. When sounds show no clear temporal structure, spectral-based feature extraction methods are probably the better choice. In this case, our approach can just as well be applied by learning sparse atomic functions on the signals' spectral envelope or pitch curve instead of the time-domain signal.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers and B. L. Sturm for their helpful comments on this work.

REFERENCES

[1] E. Smith and M. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, no. 7079, pp. 978–982, 2006.

[2] P. Manzagol, T. Bertin-Mahieux, and D. Eck, "On the use of sparse time-relative auditory codes for music," in *Proc. Int. Conf. Music Inf. Retrieval (ISMIR)*, Philadelphia, PA, 2008, pp. 14–18.

[3] M. Plumbley, S. Abdallah, T. Blumensath, and M. Davies, "Sparse representations of polyphonic music," *Signal Process.*, vol. 86, no. 3, pp. 417–431, 2006.

[4] E. Smith and M. Lewicki, "Efficient coding of time-relative structure using spikes," *Neural Comput.*, vol. 17, no. 1, pp. 19–45, 2005.

[5] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T. Lee, and T. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, pp. 349–396, 2003.

[6] J. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1395–1408, Jul. 2009.

[7] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[8] A. Bruckstein, D. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, 2009.

[9] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[10] M. Lewicki and B. Olshausen, "Probabilistic framework for the adaptation and comparison of image codes," *J. Opt. Soc. Amer. A*, vol. 16, pp. 1587–1601, 1999.

[11] M. Lewicki and T. Sejnowski, "Coding time-varying signals using sparse, shift-invariant representations," *Adv. Neural Inf. Process. Syst.*, pp. 730–736, 1999.

[12] K. Adiloglu, R. Annies, H. Purwins, and K. Obermayer, "Visualization and measurement assisted design," NI-BIT, Berlin, Tech. Rep., 2009.

[13] K. Adiloglu, R. Annies, H. Purwins, and K. Obermayer, "Representations and predictors for everyday sounds," NI-BIT, Berlin, Tech. Rep., 2008.

[14] B. Sturm, M. Mordivone, and L. Daudet, “Musical instrument identification using multiscale mel-frequency cepstral coefficients,” in *Proc. EUSIPCO*, 2010.

[15] S. Chu, S. Narayanan, and C.-C. J. Kuo, “Environmental sound recognition with time–frequency audio features,” *Trans. Audio, Speech Lang. Process.*, vol. 17, no. 6, pp. 1142–1158, Aug. 2009.

[16] R. Pichevar, H. Najaf-Zadeh, and L. Thibault, “A biologically-inspired low-bit-rate universal audio coder,” in *Proc. Audio Eng. Soc. Conv.*, Vienna, Austria, 2007.

[17] K. Tanghe, S. Degroeve, and B. De Baets, “An algorithm for detecting and labeling drum events in polyphonic music,” in *Proc. MIREX*, London, U.K., 2005.

[18] K. Yoshii, M. Goto, and H. Okuno, “Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 1, pp. 333–345, Jan. 2006.

[19] J. Paulus and T. Virtanen, “Drum transcription with non-negative spectrogram factorization,” in *Proc. EUSIPCO*, 2005, pp. 4–8.

[20] P. Herrera, A. Yeterian, and F. Gouyon, “Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques,” *Music Artif. Intell.*, pp. 69–80, 2002.

[21] S. Scholler and H. Purwins, “Sparse coding for drum sound classification and its use as a similarity measure,” in *Proc. 3rd Int. Workshop Mach. Learn. Music (MML10) at ACM Multimedia*, 2010.

[22] S. Mallat and Z. Zhang, “Matching pursuits with time–frequency dictionaries,” *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[23] G. Davis, S. Mallat, and M. Avellaneda, “Adaptive greedy approximations,” *Constructive Approx.*, vol. 13, no. 1, pp. 57–98, 1997.

[24] O. Gillet and G. Richard, “ENST-drums: An extensive audio-visual database for drum signals processing,” in *Proc. 7th Int. Symp. Music Inf. Retrieval (ISMIR’06)*, 2006, pp. 156–159.

[25] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Music genre database and musical instrument sound database,” in *Proc. ISMIR*, 2003, pp. 229–230.

[26] M. Slaney, “An efficient implementation of the Patterson-Holdsworth auditory filter bank,” Perception Group, Apple Computer, Tech. Rep., 1993.

[27] K. Mueller, S. Mika, G. Raetsch, K. Tsuda, and B. Schoelkopf, “An introduction to kernel-based learning algorithms,” *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.

[28] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” 2001 [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

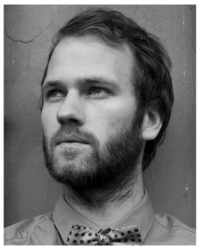
[29] G. Xu and J. Meng, “Signal enhancement with matching pursuit,” in *Proc. IEEE 60th Veh. Technol. Conf. VTC2004-Fall.* , 2004, vol. 3, pp. 1986–1990.

[30] H. Murthy and V. Gadde, “The Modified group delay function and its application to phoneme recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP’03)*, 2003, vol. 1, pp. 68–71.

[31] M. Marchini and H. Purwins, “Unsupervised generation of percussion sound sequences from a sound example,” in *Proc. Sound Music Comput. Conf.*, 2010.

[32] F. Jaillet, R. Gribonval, M. Plumbley, and H. Zayyani, “An L1 criterion for dictionary learning by subspace identification,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP’10)*, 2010, pp. 5482–5485.

[33] J. Tropp and S. Wright, “Computational methods for sparse solution of linear inverse problems,” *Proc. IEEE*, vol. 98, no. 6, pp. 948–958, Jun. 2010.



Simon Scholler received the B.S. degree in cognitive science from the University of Osnabrueck, Osnabrueck, Germany, in 2007 and the M.S. degree in computational neuroscience from the Bernstein Center for Computational Neuroscience, Berlin, Germany, in 2011.

He is currently a Research Associate at the Machine Learning Group, Technical University Berlin. His research interests mainly lie within image and sound analysis, and in applying machine learning techniques to real-world applications.

Hendrik Purwins (M’11) studied electroacoustic music and music theory at Berlin University of the Arts, Berlin, Germany, received the diploma degree in pure mathematics from Bonn and Münster Universities, and the Ph.D. degree from Berlin University of Technology, with a scholarship from the “Studienstiftung des deutschen Volkes.”

He is a Lecturer at the Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain. He has been a Visiting Researcher at IRCAM, Paris, CCRMA, Stanford, and McGill University. He started playing the violin at age of 7, he also played with the German head quarters chamber orchestra. He has written more than 50 scientific papers and co-coordinated work packages in three large European projects. His interests comprise statistical, unsupervised models for machine listening, music generation, sound resynthesis, and failure prediction in semi-conductor manufacturing.

Dr. Purwins has won 12 research grants/prizes.