Improved Audio Scene Classification Based on Label-Tree Embeddings and Convolutional Neural Networks

Huy Phan, *Student Member, IEEE*, Lars Hertel, Marco Maass, *Student Member, IEEE*, Philipp Koch, Radoslaw Mazur, *Member, IEEE*, and Alfred Mertins, *Senior Member, IEEE*

***Abstract*—In this paper, we present an efficient approach for audio scene classification. We aim at learning representations for scene examples by exploring the structure of their class labels. A category taxonomy is automatically learned by collectively optimizing a tree-structured clustering of the given labels into multiple metaclasses. A scene recording is then transformed into a label-tree embedding image. Elements of the image represent the likelihoods that the scene instance belongs to the metaclasses. We investigate classification with label-tree embedding features learned from different low-level features as well as their fusion. We show that the combination of multiple features is essential to obtain good performance. While averaging label-tree embedding images over time yields good performance, we argue that average pooling possesses an intrinsic shortcoming. We alternatively propose an improved classification scheme to bypass this limitation. We aim at automatically learning common templates that are useful for the classification task from these images using simple but tailored convolutional neural networks. The trained networks are then employed as a feature extractor that matches the learned templates across a label-tree embedding image and produce the maximum matching scores as features for classification. Since audio scenes exhibit rich content, template learning and matching on low-level features would be inefficient. With label-tree embedding features, we have quantized and reduced the low-level features into the likelihoods of the metaclasses, on which the template learning and matching are efficient. We study both training convolutional neural networks on stacked label-tree embedding images and multistream networks. Experimental results on the DCASE2016 and LITIS Rouen datasets demonstrate the efficiency of the proposed methods.**

***Index Terms*—Audio scene classification, label tree embedding, convolutional neural network, multi-stream, template matching.**

I. Introduction

The goal of audio scene classification (ASC) is to recognize a surrounding environment using acoustic signals. It enables many applications, such as surveillance [1], context-aware services [2], [3], and robotic navigation [4]. In addition, the ability to recognize an acoustic scene can also help to improve performance of the closely related task of audio event detection [5]. Therefore, ASC remains to be one of the important challenges in the field of computational auditory scene analysis [6], [7].

An acoustic scene can be thought of as a mixture of background noise and various foreground sound events. In order to automatically recognize a scene, a proper feature representation is needed, which, unfortunately, is not easily obtained due to the complexity of the content. Different low-level features have been proposed in prior works, such as Mel frequency cepstral coefficients (MFCCs) [8], [9] and Gammatone filterbank coefficients [10]. These features are usually borrowed from related problems like speech recognition and audio event classification. Besides that, several features have also been particularly designed for the task and demonstrated good performance. For instance, Histograms of Oriented Gradients (HOG) were proposed in [11]–[13] and a Gabor dictionary was used in [14]. A scene instance can also be separated into background noise and foreground sounds, and the features of both parts can be used to characterize the scene [13], [15]–[18].

Nevertheless, most (if not all) prior works used a "flat" classification scheme. On the other hand, the inherent structure of the scene category set, which may be useful for the feature learning or the classification task at hand, has not been explored. This work aims at filling this gap for the ASC task. The objective is to uncover a class hierarchy by automatically clustering similar scene categories into meta-classes with the proposed label tree learning algorithm. Afterwards, the class hierarchy is used to construct an explicit embedding to transform each segment of an audio scene into a label tree embedding feature vector. Each element of the feature vector carries the likelihood with which a given audio segment belongs to the corresponding meta-class. As a result, the target scene instance is transformed into a two-dimensional image via the learned label tree embedding. The image is formed as the output of classifiers for different subsets of labels (rows) for each time frame in the sound

Manuscript received July 12, 2016; revised November 18, 2016 and March 2, 2017; accepted March 10, 2017. Date of current version May 23, 2017. This work was supported by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative (DFG GSC 235/1). The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Juan P. Bello. *(Corresponding author: Huy Phan.)*

H. Phan is with the Institute for Signal Processing and the Graduate School for Computing in Medicine and Life Sciences, University of Lübeck, 23562 Lübeck, Germany (e-mail: phan@isip.uni-luebeck.de).

L. Hertel, M. Maass, P. Koch, R. Mazur, and A. Mertins are with the Institute for Signal Processing, University of Lübeck, 23562 Lübeck, Germany (e-mail: hertel@isip.uni-luebeck.de; maass@isip.uni-luebeck.de; koch@isip.uni-luebeck.de; mazur@isip.uni-luebeck.de; mertins@isip.uni-luebeck.de).

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/TASLP.2017.2690564

excerpt (columns). We study the class hierarchies learned from different low-level feature sets, including Gammatone cepstral coefficients [10], [19], MFCCs [20], and log-frequency filter bank coefficients [21], [22], as well as their fusion. An average pooling over time can then be applied on a label tree embedding (LTE) image to yield a global LTE feature vector for classification. These learned representations are shown to be useful for the ASC task since a good classification accuracy can be obtained even with a simple linear classifier [23]. A kernel-based fusion scheme is further proposed to combine global LTE features corresponding to different low-level features.

Despite their complex sound composition, audio scenes of the same category expose many things in common, i.e. frequent foreground events and background noise. The question is how to discover and match these patterns to leverage the classification task. We propose to use a convolutional neural network (CNN) [24] for this purpose. The proposed CNN architecture is very simple, as it consists of only three layers. These are a convolutional, pooling, and softmax layer. The combination of the former two is targeted for feature extraction whereas classification is accomplished with the latter one. The convolutional kernels play the role of the templates that will be learned by the CNN. Convolving a kernel on a scene instance, i.e. template matching, results in a feature map which indicates how well the template is matched to different parts of the scene. In turn, the pooling layer retains the single maximum value, i.e. the maximum matching score, of each feature map as the final feature. These features are finally concatenated and fed into the softmax layer for classification. The CNN is trained to maximize the classification accuracy on the training set. Therefore, the networks are supposed to uncover useful patterns from the scenes for classification, opposing to the average pooling over time which tends to blend the foreground events and background noise. During testing, we do not use the learned CNN as the final classifier but only as a feature extractor. The features learned by the network are fed to a linear Support Vector Machine (SVM) classifier as in [25]–[27].

We argue that discovering templates from low-level features would be inefficient due to the rich content of the scenes. Alternatively, we employ the LTE images as the input to the CNN. With LTE features, we have quantized and reduced the complex content of the scenes into the likelihoods of the meta-classes on which the template learning and matching can be performed more easily. We investigate two settings for CNN training. The first one trains a single CNN on multiple-channel LTE images, which consist of stacked individual LTE images learned from different low-level features. The second exploits a multi-stream CNN which combines single-stream CNNs learned on different LTE types using probabilistic fusion [28], [29]. As expected, this method leads to significant accuracy improvements on the employed datasets. Furthermore, similar to the case of classification with global LTE features, combining multiple features either by stacking or multi-stream settings is vital for good performance.

The rest of this paper is organized as follows. Some related works on audio scene classification are presented in Section II. After that, we describe our proposed methods at a high abstraction level in Section III. The learning algorithm for the label tree embeddings and the classification schemes using global LTE features are then elaborated in Section IV followed by the classification using 1-max pooling CNNs in Section V. Subsequently, Section VI presents experimental results on the employed datasets, followed by the discussion in Section VII and conclusions in Section VIII.

II. Related Works

The previous works on audio scene classification can be roughly grouped into two main classes.

Low-level feature-based approach: These approaches represent audio scenes by low-level feature primitives. Time-domain features (e.g. short-time energy, zero crossing rate), frequency-domain features (e.g. spectral centroid, spectral flux), auto-regression based features (e.g. linear prediction coefficients (LPC)), and cepstral features (e.g. MFCCs, Gammatone ceptral coefficients) have been prevalent in the literature [3], [8], [15], [20]. As an improvement, Roma *et al.* utilized recursive quantitative analyzing (RQA) to analyze the recurrent behaviour in the MFCC coefficients over time [30]. Time-frequency representations have also been proposed. Inspired by the HOG representations in the field of image processing, Rakotomamonjy and Gasso adapted HOG representations on constant-Q transform spectrogram images for audio scene representation [11]. Bisot *et al.* demonstrated that a combination of HOG and subband power distribution (SPD) features can further improve the classification accuracy [12]. Chu *et al.* [31] obtained an ensemble of time-frequency features via a matching pursuit decomposition of the audio signal. Agcaer [32] made use of amplitude modulation spectrum features obtained by two-stage recursive filter banks. A small subset of features is then optimized by the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)[33]. After the feature extraction step, the classification is finally accomplished by some back-end classifiers. Various classifiers have been used, such as Linear Discriminant Analysis (LDA) [32], Hidden Markov Models (HMMs) [3], Gaussian Mixture Models (GMMs) [20], [31], SVMs [11], [30], and Deep Neural Networks (DNNs) [8].

High-level feature-based approaches: These approaches use a set of high-level features to represent audio scenes. These features are usually obtained through classifying or clustering on low-level features. In [34], Aucouturier *et al.* obtained bag-of-features (BOF) representations by estimating the distribution of frame-based MFCC features using a GMM. Lee *et al.* used a sparse restricted Boltzmann machine (RBM) followed by a max-pooling scheme to select the Mel-frequency time-frequency features that correspond to foreground events [35]. The selected features are then averaged to form a scene-level feature vector. Bisot *et al.* demonstrated that time-frequency features can be learned under an unsupervised setting with kernel principal component analysis (KPCA) and nonnegative matrix factorization (NMF). At higher semantic levels, due to the fact that a scene can be very well characterized by its foreground sound events [16], [18], Heittola *et al.* described a scene by the histogram of foreground audio events which are outputted

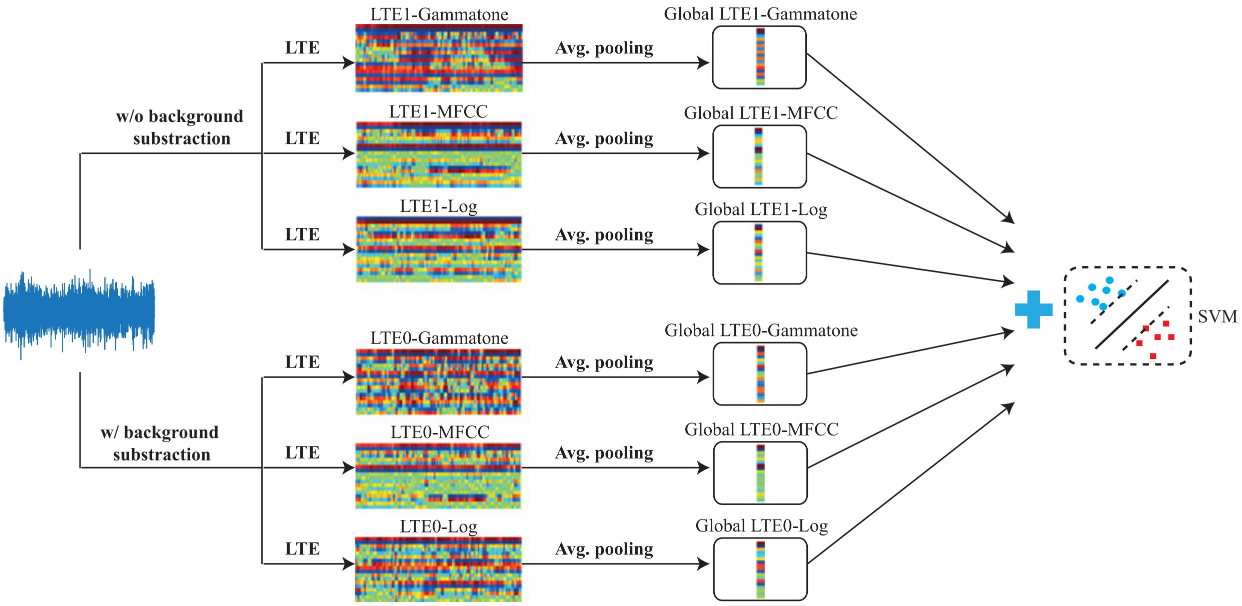


Fig. 1. Overview of scene classification with global LTE features. Via the LTE algorithm, six LTE images with three different low-level feature sets and background noise subtraction switched on and off, respectively, are obtained for a scene instance. The average pooling over time is then carried out to produce global LTE feature vectors, which are used for classification with an SVM. Fusion of multiple LTE feature channels is also investigated.

by an event detector [17]. In [15], background noise, extracted by tracking minimum statistics over time-frequency space [36], has also been shown to be capable of characterizing a scene. Ye *et al.* [13] proposed to take into account both background noise and foreground events to represent a scene. Two BOF vectors were learned via a GMM and Fisher Vector encoding, one for background noise and another for foreground events. The final classification is accomplished by probabilistic fusion of two SVM classifiers on two feature channels. More recently, generic features via similarity to speech patterns [23] and transfer learning from visual knowledge [37] have been reported to give good generalization.

Although good performance on different audio scene benchmark datasets has been reported for the above mentioned approaches, they have a common shortcoming. They used a “flat” classification scheme and do not explore the structured nature of the scene categories for classification. Our previous work [23] demonstrated that learned representations that take into account the structure of scene data can be highly discriminative, as state-of-the-art performance can be obtained even with simple linear classifiers. The label tree embeddings used in this work also bear some resemblance with those in [21], [38] in which label tree embeddings of speech patterns were learned to extract generic features for audio events.

Deep CNNs have also recently been employed to tackle the audio scene classification task in the context of the DCASE 2016 challenge [20], [39], such as those in [40]–[43]. These CNNs share the same processing pipeline. The 30-second scene snippets are first decomposed into multiple small segments. Segment-wise classification is then performed followed by an aggregation step to combine segment-wise classification results with some voting schemes to yield the final classification labels. Compared to these deep and large networks, our CNN architecture is much smaller and simpler, allowing it to handle the whole signals as input and learn features for signal-wise classification. The max pooling scheme used in our proposed network architecture has also been shown useful for robust audio event recognition in our previous work [24] and for text classification [44].

This work extends our previous works in [23], [24] in five major aspects. (1) We study LTE representation learning with different low-level feature sets as well as their combination. (2) We perform audio background subtraction prior to the LTE representation learning and show that the resulting LTE representations are useful, particularly for the processing with the proposed CNN afterward. (3) Instead of employing the trained CNN for classification with its softmax layer as in [23], we treat the network as a feature extractor and use the extracted features to train a linear SVM for classification as in [25]–[27]. The linear SVM offers better generalization thanks to its well-known maximum-margin property. (4) We study training CNNs not only on single LTE types but also on stacked LTE images learned from different types of low-level features. The latter allows the network to learn useful patterns across different LTE channels, leading to better performance compared to the former one. (5) Last but not least, as multi-stream CNNs have been successful for many classification tasks, we also examine and evaluate here a multi-stream CNN which probabilistically fuses single-channel CNNs trained on different LTE types.

III. Approach Overview

Our approach can be divided into three parts, label tree embedding, CNNs for template learning and matching, and classification with linear SVMs, which can be explained at a high abstraction level as in Figs. 1–3, respectively.

Using the proposed LTE learning algorithm in Section IV, a scene instance is mapped into a 2-dimensional LTE image of

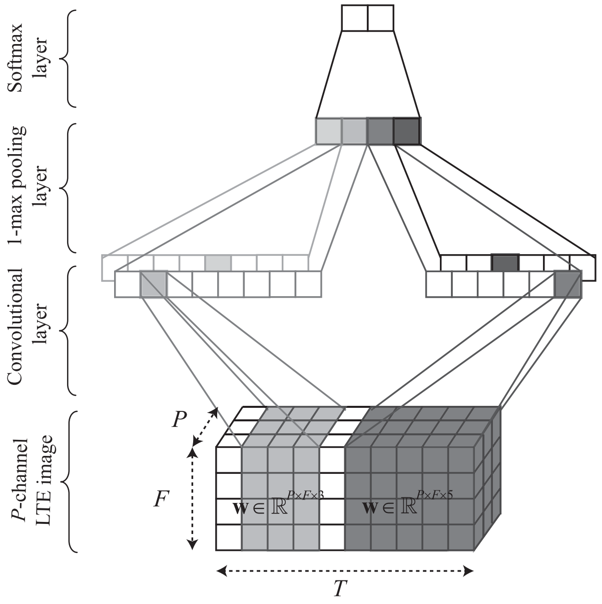

Softmax layer
1-max pooling layer
Convolutional layer
P-channel LTE image
P
F
T
$\mathbf{W} \in \mathbb{R}^{P \times F \times 3}$
$\mathbf{W} \in \mathbb{R}^{P \times F \times 5}$

Fig. 2. Illustration of the proposed CNN architecture on a P-channel LTE image. The network consists of two filter sets with widths $w = 3$ and $w = 5$ at the convolutional layer. Each filter set contains two individual filters.

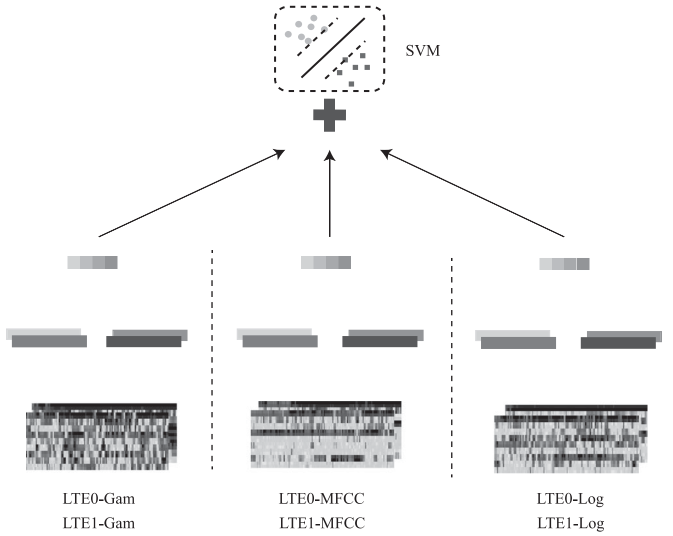


Fig. 3. Illustration of classification with linear SVMs using features extracted from the trained CNNs. Probabilistic fusion with multi-stream CNNs is also studied.

TABLE I
LTE Combination Systems

LTE system	Constituents
LTE-Gam	LTE0-Gam, LTE1-Gam
LTE-MFCC	LTE0-MFCC, LTE1-MFCC
LTE-Log	LTE0-Log, LTE1-Log
LTE0-Fusion3	LTE0-Gam, LTE0-MFCC, LTE0-Log
LTE1-Fusion3	LTE1-Gam, LTE1-MFCC, LTE1-Log
LTE-Fusion6	LTE0-Gam, LTE0-MFCC, LTE0-Log, LTE1-Gam, LTE1-MFCC, LTE1-Log

TABLE II
Single-Stream CNN Systems With Combined LTE Images

CNN system	LTE image constituents
CNN-Gam	LTE0-Gam, LTE1-Gam
CNN-MFCC	LTE0-MFCC, LTE1-MFCC
CNN-Log	LTE0-Log, LTE1-Log
CNN0-Fusion3	LTE0-Gam, LTE0-MFCC, LTE0-Log
CNN1-Fusion3	LTE1-Gam, LTE1-MFCC, LTE1-Log
CNN-Fusion6	LTE0-Gam, LTE0-MFCC, LTE0-Log, LTE1-Gam, LTE1-MFCC, LTE1-Log

TABLE III
Multistream CNN Systems That Fuses Multiple Single-Stream CNNs

Multi-stream CNN system	Single-stream CNN constituents	Fusion scheme
CNN-Multi-Mean	CNN-Gam, CNN-MFCC, CNN-Log	mean
CNN-Multi-Max	CNN-Gam, CNN-MFCC, CNN-Log	max
CNN-Fusion3-Multi-Mean	CNN0-Fusion3, CNN1-Fusion3	mean
CNN-Fusion3-Multi-Max	CNN0-Fusion3, CNN1-Fusion3	max

size $F \times T$ where F is the number of derived features and T is the time frames. The exact interpretation of F and T will be described later in Section IV. We investigate three different low-level feature sets for LTE learning, including Gammatone cepstral coefficients [10], [19], MFCCs [20], and log-frequency filter bank coefficients [21], [22]. We also study how the presence/absence of background noise affects the LTE representations. We preprocess the input signals using minimum statistics noise estimation and subtraction [36] whenever we need to remove background noise. As a result, six LTE images are obtained for a single scene instance, namely LTE0-Gam, LTE0-MFCC, LTE0-Log, LTE1-Gam, LTE1-MFCC, and LTE1-Log where "0" and "1" denote presence/absence of the background noise. The average pooling over time is then applied to the LTE images to produce global LTE feature vectors which are presented to SVM classifiers for classification. We study the combinations of complementary LTE channels derived from the same types of low-level features (LTE-Gam, LTE-MFCC, and LTE-Log), the combinations of those LTEs with the presence/absence of background noise (LTE0-Fusion3, LTE1-Fusion3), and the combination of all the six LTE feature vectors altogether (LTE-Fusion6). The summary of these combinations is given in Table I.

Arguing the drawbacks of the above-described average pooling, we alternatively propose to automatically learn templates that are useful for the task from the LTE images using the proposed CNN architecture in Fig. 2. We train different CNNs with different combinations of LTE images as summarized in Table II. In order to combine multiple LTE images, we stack them together to make a multiple-channel LTE image on which 3-dimensional kernels will be learned by the CNNs. Afterwards, the trained CNNs are utilized to perform template matching on inputted LTE images for feature extraction. Finally, the extracted features are classified by linear SVMs. We will show that this classification scheme leads to significant improvements over the one with global LTE features.

We further study probabilistic fusion of different CNNs in multi-stream settings. These multi-stream CNN systems either combine three single-stream CNNs with different feature types or those two with background noise switched on/off. Table III

Algorithm 1: Partition algorithm at a split node.

```
Data: $\ell \in \mathcal{L}$, $\mathcal{S}^\ell \subset \mathcal{S}$
Result: the optimal partition $\{\ell_L, \ell_R\}$ and $\{\mathcal{S}^\ell_L, \mathcal{S}^\ell_R\}$
begin
    divide $\mathcal{S}^\ell$ into two equal halves $\mathcal{S}^\ell_{\text{train}}$ and $\mathcal{S}^\ell_{\text{eval}}$;
    train a multi-class classifier $\mathcal{M}^\ell$ using $\mathcal{S}^\ell_{\text{train}}$;
    classify $\mathcal{S}^\ell_{\text{eval}}$ with the classifier $\mathcal{M}^\ell$;
    obtain the classification confusion matrix
    $\mathbf{A} \in \mathbb{R}^{|\ell| \times |\ell|}$;
    symmetrize $\mathbf{A}$ by $\bar{\mathbf{A}} = (\mathbf{A} + \mathbf{A}^\mathsf{T})/2$;
    partition $\ell$ into $\{\ell^L, \ell^R\}$ and $\mathcal{S}_\ell$ into $\{\mathcal{S}^L_\ell, \mathcal{S}^R_\ell\}$ to
    minimize
        $E(\ell) = \sum_{i,j \in \ell^L} \bar{\mathbf{A}}_{ij} + \sum_{m,n \in \ell^R} \bar{\mathbf{A}}_{mn}$        (1)
end
```

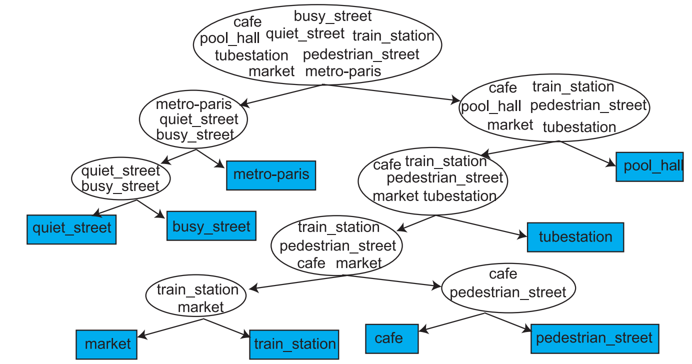


Fig. 4. A part of the label tree learned from the LITIS Rouen dataset with Gammatone cepstral coefficients.

illustrates a summary of them. Both mean and max fusion strategies will be investigated.

IV. Classification With LTE Representations

A. LTE Representations for Audio Scenes

1) Learning a Label Tree: Let us consider a database (e.g. a scene database) with a label set $\mathcal{L} = \{1, \ldots, C\}$ of C categories. Given the label set and the examples from the database, we aim at learning a label tree that encodes the hierarchical structure of the class labels [21], [38]. The idea is to recursively partition the label set $\mathcal{L}$ into disjoint subsets in such a way that the examples of the obtained subsets can be easily separated from one another. To explain the procedure, let the set of examples extracted from the training data be given by $\mathcal{S} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{|\mathcal{S}|}$. Moreover, let $\mathbf{x} \in \mathbb{R}^M$ denote a low-level feature vector of size M, let $c \in \mathcal{L}$ be a class label, and let $|\cdot|$ denote the set cardinality.

A learning algorithm is used to grow the label tree in a recursive manner so that each of its nodes is associated with a label subset of the entire set $\mathcal{L}$. The algorithm starts with the root node which is linked to $\mathcal{L}$. Without loss of generality, let us consider a current split node with a label subset $\ell \subset \mathcal{L}$. We then want to split ℓ into two smaller subsets ℓ^L and ℓ^R that fulfill the following conditions: $\ell^L \neq \emptyset$, $\ell^R \neq \emptyset$, $\ell^L \cup \ell^R = \ell$, and $\ell^L \cap \ell^R = \emptyset$. Among $2^{|\ell|-1} - 1$ such possible partitions $\{\ell^L, \ell^R\}$, we then select the optimal one such that ℓ^L and ℓ^R can be separated with as few errors as possible using a binary classifier. Afterwards, the subsets ℓ^L and ℓ^R are forwarded to the left and right child nodes of the current node, respectively. The recursive splitting procedure is terminated as soon as a leaf node with a single class label is reached.

Let us denote the sample subset corresponding to a label subset ℓ as $\mathcal{S}^\ell \subset \mathcal{S}$. The algorithm for partitioning $\mathcal{S}^\ell$ into $\{\mathcal{S}^L_\ell, \mathcal{S}^R_\ell\}$ is presented as Algorithm 1. In the algorithm, the multi-class classifier $\mathcal{M}^\ell$ is trained using random forest classification [45] with 200 trees. The elements $\mathbf{A}_{ij}$ of the matrix $\mathbf{A}$ are computed by

$$\mathbf{A}_{ij} = \frac{1}{\left|\mathcal{S}^\ell_{\text{eval},i}\right|} \sum_{\mathbf{x} \in \mathcal{S}^\ell_{\text{eval},i}} P(j|\mathbf{x}, \mathcal{M}^\ell), \tag{2}$$

where $\mathcal{S}^\ell_{\text{eval},i} \subset \mathcal{S}^\ell_{\text{eval}}$ is the set of samples with the label i. $P(j|\mathbf{x}, \mathcal{M}^\ell)$ denotes the probability that the classifier $\mathcal{M}^\ell$ predicts the sample $\mathbf{x}$ as class j. $\mathbf{A}_{ij}$ with $i \neq j$ expresses how likely a sample of class i is wrongly predicted to belong to class j by the classifier.

With the partition criterion in (1), categories that are difficult to separate from one another are clustered into the same subset. As a result, we can expect to obtain meta-classes ℓ^L and ℓ^R that can be easily separated from each other. Since it is hard to solve the optimization problem in (1) directly, we alternatively solve a relaxed version of it using spectral clustering [46] applied on the matrix $\bar{\mathbf{A}}$.

We demonstrate in Fig. 4 a part of the label tree learned from the LITIS Rouen dataset [11] with Gammatone cepstral coefficients (more details in Section IV-B).

2) LTE Representations: After completion of the learning process, the obtained label tree consists of $(C-1)$ split nodes in total. Furthermore, the original label set $\mathcal{L}$ has been divided into $(C-1) \times 2$ disjoint subsets. Let us consider a split-node index i with $1 \leq i \leq C-1$. We then want to derive the label tree embedding $\Psi : \mathbb{R}^M \to \mathbb{R}^{(C-1)\times 2}$ where

$$\Psi(\mathbf{x}) = \left(\psi^L_1(\mathbf{x}), \psi^R_1(\mathbf{x}), \ldots, \psi^L_{C-1}(\mathbf{x}), \psi^R_{C-1}(\mathbf{x})\right). \tag{3}$$

In the above expression, $\psi^L_i(\mathbf{x})$ and $\psi^R_i(\mathbf{x})$ represent the likelihoods with which the test sample $\mathbf{x}$ belongs to two meta-classes associated with the left and right child nodes of the split node i. That is, using the embedding, we transform $\mathbf{x}$ into a vector $\Psi(\mathbf{x})$ containing meta-class likelihoods. Finally, the vector $\Psi(\mathbf{x})$ is used as a high-level representation for $\mathbf{x}$.

At the split node i associated with the label subset ℓ_i and the optimal partitioning $\{\ell^L_i, \ell^R_i\}$, the likelihoods $\psi^L_i(\mathbf{x})$ and $\psi^R_i(\mathbf{x})$ can be computed as follows. Considering the samples with their labels in ℓ^L_i and ℓ^R_i as negative and positive examples, respectively, we train a binary random-forest classifier $\mathcal{M}^{\ell_i}$ using the sample set S^{ℓ_i} as training data. The number of trees is set to 200. The likelihoods $\psi_i^L(\mathbf{x})$ and $\psi_i^R(\mathbf{x})$ then read

$$\psi_i^L(\mathbf{x}) = P(\text{negative}|\mathbf{x}, \mathcal{M}^{\ell_i}), \quad (4)$$

$$\psi_i^R(\mathbf{x}) = P(\text{positive}|\mathbf{x}, \mathcal{M}^{\ell_i}), \quad (5)$$

where $P(\text{negative}|\mathbf{x}, \mathcal{M}^{\ell_i})$ and $P(\text{positive}|\mathbf{x}, \mathcal{M}^{\ell_i})$ denote the posterior probabilities for classifying the test sample $\mathbf{x}$ into the negative and positive class, respectively, given the classifier $\mathcal{M}^{\ell_i}$. These posterior probabilities can be obtained easily, as the random forest classification naturally supports probability output [45].

B. Recognition Using LTE Representations

Using the above framework, we derived the following LTE representations with different low-level feature sets: (1) Gammatone cepstral coefficients (LTE0-Gam and LTE1-Gam), (2) MFCCs (LTE0-MFCC and LTE1-MFCC), log frequency filter banks (LTE0-Log and LTE1-Log), and their fusion (LTE-Gam, LTE-MFCC, LTE-Log, LTE0-Fusion3, LTE1-Fusion3, and LTE-Fusion6). An overview of the LTE representations was given in Section III. In the experiments, we did not use the whole 30-second audio snippets as data samples in the label tree embedding algorithm. Instead, the snippets were decomposed into $T = 238$ segments of length 250 ms with a hop size of 125 ms. Furthermore, the segments were labeled with the label of the snippet. These segments were then used as data examples in the algorithm. By doing this we try to capture meaningful foreground events occurring in the long recordings, whose lengths are typically in the order of some hundreds of milliseconds. With each audio segment being represented by an LTE feature vector, we obtain an $F \times T$ LTE image for the 30-second scene instance where $F = (C - 1) \times 2$.

LTE0-Gam and LTE1-Gam: In this case, we characterize an audio segment by $M = 64$ Gammatone ceptral coefficients. To accomplish this, the audio segment is decomposed into 50 ms frames with a hop size of 25 ms. 64 Gammatone cepstral coefficients are then extracted for each frame [19]. The feature vector for the whole segment is finally computed by averaging the frame-wise feature vectors.

LTE0-MFCC and LTE1-MFCC: For these LTE features, we employ $M = 60$ MFCC features in replacement for Gammatone cepstral coefficients in LTE0-Gam and LTE1-Gam. MFCCs are calculated for each 50 ms frame with a Hamming window and 40 mel bands. Beside the first 20 coefficients (including 0th order coefficients), 20 delta coefficients, and 20 acceleration coefficients are also calculated using a window length of nine frames.

LTE0-Log and LTE1-Log: Here, we utilize 20 log-frequency filter bank coefficients, their first and second derivatives in frequency direction, zero-crossing rate, short-time energy, four sub-band energies, spectral centroid, and spectral bandwidth, similar to our previous works [21], [22]. The total number of features is $M = 65$.

In order to perform classification with the LTE features, we use average pooling on each $F \times T$ LTE image over time to obtain the global F-dimensional feature vector for each scene instance. Note that in order to extract LTE images for the training instances, we conducted 10-fold cross-validation on training data.

LTE-Gam, LTE-MFCC, LTE-Log, LTE0-Fusion3, LTE1-Fusion3, and LTE-Fusion6: In order to take advantage of representations from different perspectives (i.e. the different low-level feature types and the presence/absence of background noise), we combine different global LTE feature vectors using the extended Gaussian-χ^2 kernel [47] given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\sum_k \frac{1}{\bar{D}^k} D\big(\Psi^k(\mathbf{x}_i), \Psi^k(\mathbf{x}_j)\big)\Big) \quad (6)$$

where $D\big(\Psi^k(\mathbf{x}_i), \Psi^k(\mathbf{x}_j)\big)$ is the χ^2 distance between the global LTE feature vectors of the embedded scene instances $\Psi^k(\mathbf{x}_i)$ and $\Psi^k(\mathbf{x}_j)$ with respect to the k-th channel where

$$k \in \{\text{LTE0-Gam}, \text{LTE1-Gam}\}, \quad (7)$$

$$k \in \{\text{LTE0-MFCC}, \text{LTE1-MFCC}\}, \quad (8)$$

$$k \in \{\text{LTE0-Log}, \text{LTE1-Log}\}, \quad (9)$$

$$k \in \{\text{LTE0-Gam}, \text{LTE0-MFCC}, \text{LTE0-Log}\}, \quad (10)$$

$$k \in \{\text{LTE1-Gam}, \text{LTE1-MFCC}, \text{LTE1-Log}\}, \quad (11)$$

$$k \in \{\text{LTE0-Gam}, \text{LTE0-MFCC}, \text{LTE0-Log}, \text{LTE1-Gam}, \text{LTE1-MFCC}, \text{LTE1-Log}\}, \quad (12)$$

for LTE-Gam, LTE-Gam, LTE-Gam, LTE0-Fusion3, LTE1-Fusion3, and LTE-Fusion6, respectively. $\bar{D}^k$ denotes the average χ^2 distance between the embedded scene instances in the training data for the k-th channel.

V. LTE Template Learning and Matching With CNNs

A. Potential Issues With the Average Pooling

We argue that the average pooling on the LTE images results in global feature vectors that are not optimal. Beside background noise, an audio scene typically contains different kinds of foreground events, which are sparsely and irregularly distributed. It can be interpreted as foreground events embedded in background noise. Although foreground events [16]–[18], [35] and background noise [15] have been used as signatures for audio scenes, they should be considered separately [13]. Unfortunately, with the average pooling, we tend to mix up the sparse foreground events into the dominating background noise. To overcome this issue, we alternatively propose to discover templates that are useful for the classification task from the LTE images. The proposed CNN architecture designed for this purpose is relatively simple. It consists of one convolutional layer, one pooling layer, and one softmax layer as illustrated in Fig. 2. Different from typical CNN architectures, the size of the convolutional filters at the convolutional layer is not fixed. We allow multiple filters with different sizes to be learned simultaneously. In addition, since our intention is to perform pattern matching, we do not pursue subsampling at the pooling layer as usual but reduce each feature map to the most prominent matching score. The learned templates potentially correspond to discriminative foreground events as well as background noise.

B. CNNs for Pattern Learning and Matching

1) Multichannel LTE Images: The inputs to the networks are the entire LTE images. Our experiments reveal that different low-level features (e.g. Gammatone cepstral coefficients, MFCCs, and log-frequency filter banks) used to derive LTE images are good for different scene categories. In addition, background noise is also shown to be useful. Therefore, it is reasonable to let the CNNs look at multiple LTE images at the same time to discover the most useful templates across different channels. To accomplish this, we stack the individual LTE images to produce a multi-channel LTE image of size $P \times F \times T$ for the scene instance when P is the number of single LTE images. We train the following CNNs: CNN-Gam, CNN-MFCC, CNN-Log, CNN0-Fusion3, CNN1-Fusion3, and CNN-Fusion6 as described in Section III.

2) Convolutional Layer: Let $\mathbf{S} \in \mathbb{R}^{P \times F \times T}$ denote an input LTE image and let $\mathbf{w} \in \mathbb{R}^{P \times F \times w}$ be the impulse response of a 3-dimensional linear filter with a temporal width of w. We convolve the filter with the LTE image in the time direction. Let $\mathbf{S}[i:j]$ further denote the audio segments (i.e. the adjacent LTE image slices) from i to j. Convolving a filter $\mathbf{w}$ with the LTE image $\mathbf{S}$ results in an output vector $\mathbf{O} = (o_1, \dots, o_{T-w+1})$ whose elements are given by

$$o_i = (\mathbf{S} * \mathbf{w})_i = \sum_{k,l,m} (\mathbf{S}[i:i+w-1] \odot \mathbf{w})_{k,l,m}. \qquad (13)$$

Here $*$ and $\odot$ indicate the convolution and element-wise multiplication operations, respectively. After that, an activation function h is applied to the output vector to yield the feature map $\mathbf{A} = (a_1, \dots, a_{T-w+1})$ where

$$a_i = h(o_i + b). \qquad (14)$$

In (14), $b \in \mathbb{R}$ denotes a bias term. We use *Rectified Linear Units* (ReLU) [48] as the activation function due to their low computational cost:

$$h(x) = \max(0, x). \qquad (15)$$

To encourage the network to learn multiple complementary templates, we design the network to have Q different filters of the same temporal width concurrently. Moreover, since patterns in a scene (e.g. foreground events) may have different durations, we include R such filter sets with different temporal widths, to be able to to capture them more efficiently. The total number of filters is therefore $Q \times R$.

3) Max Pooling Layer: The feature map obtained by convolving a filter over an LTE image indicates how well the template is matched to different parts of the images. We then employ max pooling on the feature map to obtain a single most dominant feature [24], [44] which corresponds to the maximum matching score. This pooling strategy offers a unique advantage. Despite the varying dimensionalities of the feature maps (due to different widths of the filters and variable lengths of the input signals), the pooled feature vectors always have the same size [24], [44], [49]. Therefore, the signals can be of any arbitrary size. There is no need to fix them to a uniform duration (e.g. 30 seconds), as in the common setting for the task.

With its feature map reduced to a single most dominant feature by the 1-max pooling function, each filter in the convolutional layer is expected to be optimized to capture a useful pattern that could occur at any time in a scene. Pooling all feature maps of $Q \times R$ filters results in a feature vector of size $Q \times R$.

4) Softmax Layer: Classification is accomplished by a standard softmax layer. Being presented with the fixed-size feature vector obtained after the pooling layer, the softmax layer computes the posterior probability over the class labels. The network parameters $\boldsymbol{\theta}$ are eventually tuned to minimize the cross-entropy error for N training samples:

$$E(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i(\boldsymbol{\theta})) + \frac{\lambda}{2} ||\boldsymbol{\theta}||_2^2. \qquad (16)$$

By doing this, the KL-divergence between the predicted posterior distribution $\hat{y}$ and the one-hot encoded groundtruth distribution y will be minimized. In (16), λ is the hyper-parameter that trades off the error term and the ℓ_2-norm regularization term. For further regularization, we exploit dropout [50] by randomly setting zeros to the entries of the weight vector with a predefined probability. The network training is performed using the *Adam* optimizer [51].

C. Classification With CNN Features

Instead of using a trained CNN directly for classification, we evaluate it on a scene instance and extract the feature vector behind the pooling layer to represent the scene instance. The feature vectors extracted from the training scene examples are then used to train a linear SVM classifier which is finally employed to classify the feature vectors extracted from the unseen examples in the test set.

Using SVMs (especially linear ones) in combination with convolutional nets as part of a multistage process has been proposed in the literature [25]–[27]. A CNN is first trained to learn good invariant representations which are then treated as input and fed into SVMs for classification. The rationale of using support vector machines as an alternative to softmax for classification is their maximum margin property which usually leads to better generalization [52].

To benefit from CNN features learned from different LTE types (e.g. CNN-Gam, CNN-MFCC, and CNN-Log), we perform classification with multi-stream CNNs which have been shown efficient for different classification tasks [28], [29]. We fuse the classification probabilities outputted by the linear SVMs on individual CNN streams using mean and max strategies. The raw SVM scores are first converted and calibrated into a proper posterior probability as in [53], [54]. It should be noted that one can alternatively use the posterior probabilities outputted by the softmax layer for this purpose, but in our experiments, the use of SVMs turned out to be superior. Let us denote the classification probabilities from the k-th out of K streams on a test scene instance as $\mathbf{P}^k = (P_1^k, P_2^k, \dots, P_C^k) \in \mathbb{R}_+^C$ with C being the number of classes. The mean classification probability is then $\bar{\mathbf{P}} = (\bar{P}_1, \bar{P}_2, \ldots, \bar{P}_C)$ where

$$\bar{P}_i = \frac{1}{K}\sum_{k=1}^{K} P_i^k \text{ for } 1 \le i \le C. \tag{17}$$

The predicted label $\hat{c}$ is determined by

$$\hat{c} = \arg\max_i \bar{P}_i. \tag{18}$$

For the max strategy, the fused classification probability is given by $\breve{\mathbf{P}} = (\breve{P}_1, \breve{P}_2, \ldots, \breve{P}_C)$ where

$$\breve{P}_i = \max(P_i^k) \text{ for } 1 \le k \le K. \tag{19}$$

Likewise, the predicted label $\hat{c}$ is determined as in (18).

A similar procedure is conducted for CNN-Fusion3-Multi-Mean and CNN-Fusion3-Multi-Max which combine two streams CNN0-Fusion3 and CNN1-Fusion3.

VI. Experiments

A. Datasets

We employed the following two datasets in our experiments:

DCASE2016 dataset: The setup is based on the development data as described in Task 1 of the DCASE 2016 challenge [20], [39]. The signals were recorded with a sampling frequency of 44100 Hz. The development data consists of 15 scene classes with 78 30-second audio signals per class. The data is divided into 4 folds for cross-validation purpose. The average classification accuracy over all folds will be reported in our experiments. Since it was found by the challenge organizers that there exist errors in some recordings, we simply removed erroneous segments from the signals. This error removal resulted in some LTE images with $T < 238$ columns. We performed circular padding to make them 238 columns.

LITIS Rouen dataset: This dataset includes 19 urban scene classes with 3026 30-second-long examples in total [11]. Its overall duration is 1500 minutes, which is, to our knowledge, the largest publicly available ASC dataset so far. The audio signals were recorded at a sampling frequency of 22050 Hz. Each scene category is associated with a specific location, for example a train station, an airplane, or an open market. The dataset is provided with 20 training/testing splits. Our experiments obey this standard setting and the average performance will be reported. Opposed to the DCASE2016 dataset, F1-score will be used as the main evaluation metric since this dataset exhibits significant imbalance in the number of samples per class.

B. Experimental Setup

For classification with the global LTE features (i.e. LTE0-Gam, LTE0-MFCC, LTE0-Log, LTE1-Gam, LTE1-MFCC, and LTE1-Log), we trained the final scene classifiers using one-vs-one χ^2-kernel SVMs. For LTE-Gam, LTE-MFCC, LTE-Log, LTE0-Fusion3, LTE1-Fusion3, and LTE-Fusion6, the classification was accomplished using nonlinear SVMs with the kernel given in (6). We conducted 10-fold cross-validation to tune the hyperparameters of the SVMs. For the CNNs, different hyperparameters are involved and specified in Table IV. The filter width w was set to 3, 5, and 7 segments, which is equivalent to durations of 0.5, 0.75, and 1 seconds, respectively. We set the number of filters to $Q = \{100, 200, 300, 400, 500, 1000\}$ in order to study its influence on the classification performance. The CNNs were trained for 500 epochs with a minibatch size of 50. The hyperparameters of the final linear SVMs that classify the CNN features were also tuned via 10-fold cross-validation.

TABLE IV
Hyperparameters of the Proposed CNN Networks

Hyper-parameter	Value
Filter width w	$\{3, 5, 7\}$
Learning rate for the Adam optimizer	0.0001
Dropout rate	0.5
Regularization parameter λ	0.001

TABLE V
Performance Obtained by LTE-Based Classifiers

Classifier Type	DCASE2016	LITIS Rouen
LTE0-Gam	73.4	90.0
LTE0-MFCC	73.9	87.4
LTE0-Log	72.7	89.9
LTE1-Gam	71.1	94.0
LTE1-MFCC	71.4	92.0
LTE1-Log	73.1	92.8
LTE-Gam	75.9	94.7
LTE-MFCC	73.6	93.3
LTE-Log	75.5	94.5
LTE0-Fusion3	75.3	92.4
LTE1-Fusion3	75.3	95.7
LTE-Fusion6	77.6	95.0

Classification accuracy (%) is used for the DCASE2016 dataset and F1-score (%) is used for the LITIS Rouen dataset.

C. Experimental Results

1) Performance of Global LTE Features: The classification performance obtained by the global LTE systems are shown in Table V for the two datasets. In terms of individual LTE features, as can be seen, the three employed low-level feature sets perform differently. For example, while LTE0-MFCC performs best on DCASE2016, LTE1-Gam dominates others on LITIS Rouen. Overall, the LTE features derived from low-level features in the absence of background noise offer better performance than those with background noise on the DCASE2016 dataset, except for LTE0-Log. However, opposite results can be seen on the LITIS Rouen dataset. As expected, the combination of complementary LTE features, i.e. both cases of the presence and absence of background noise, of the same low-level feature types significantly boosts the performance. For instance, LTE-Gam outperforms the best single LTE systems on both datasets (i.e. LTE0-Gam on DCASE2016 and LTE1-Gam on LITIS Rouen by 2.5% and 0.7% absolute, respectively). These results confirm that background subtraction preprocessing before feature learning is useful for the task. LTE0-Fusion3 and LTE1-Fusion3 also show significant improvements over their individual

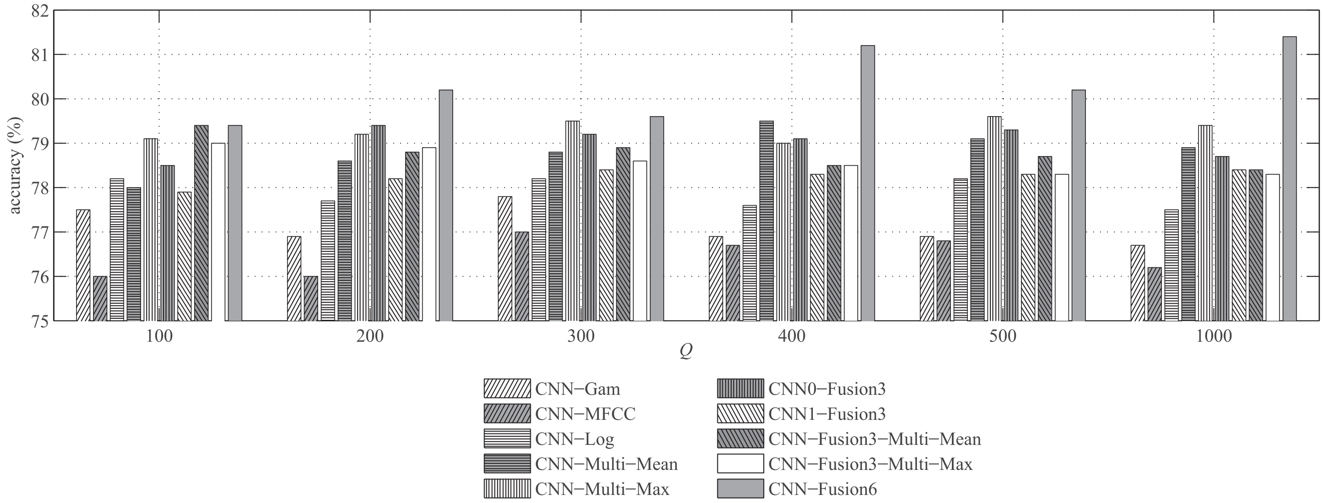


Fig. 5. **DCASE2016 dataset.** Performance obtained by CNNs in terms of classification accuracy with different values of Q.

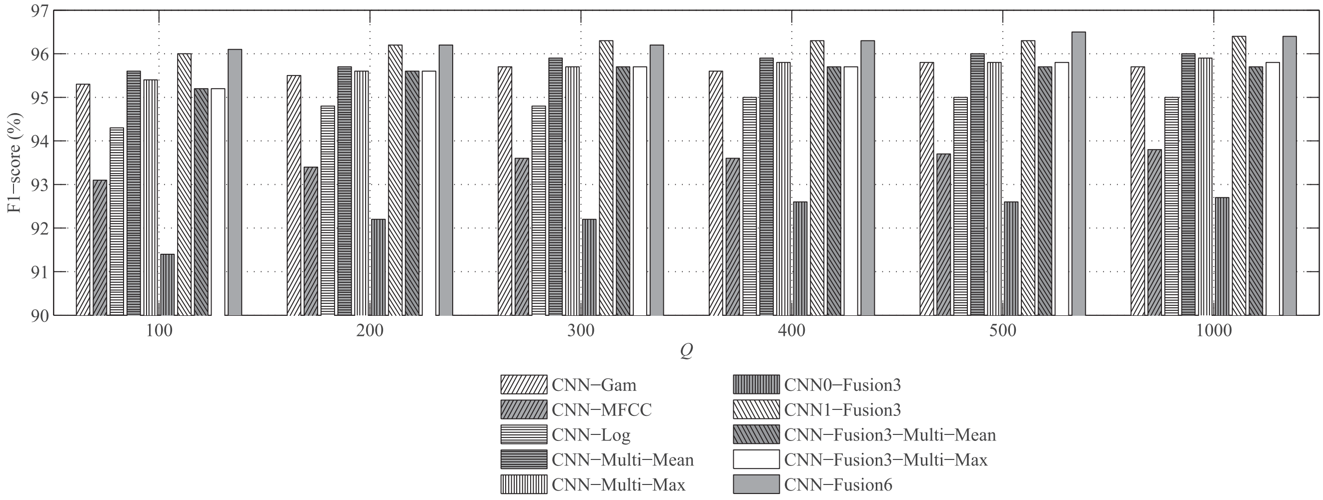


Fig. 6. **LITIS Rouen dataset.** Performance obtained by CNNs in terms of F1-score with different values of Q.

ingredients. Particularly, LTE1-Fusion3 lead to 2.2% and 1.7% absolute gains compared to its best single LTE constituents on the DCASE2016 and LITIS Rouen datasets, respectively. However, LTE-Fusion6, which integrates all six single LTE features, experiences performance drops by 0.7% absolute on LITIS Rouen compared to LTE1-Fusion3. A possible reason is that the fusion kernel given in (6) is suboptimal for this case.

2) Performance of CNN Features: The performances obtained by classification with CNN features with different values of Q are shown in Figs. 5 and 6 for DCASE2016 and LITIS Rouen, respectively.

The previous findings with the global LTE classification can also be seen here. The performance of different low-level features varies depending on datasets. While CNN-Gam outperforms two others on LITIS Rouen, CNN-Log is found the best on DCASE2016. However, their performance differences become much smaller than those in the classification with the global LTEs. Background noise is still essential under this classification scheme. This can be seen from a better classification performance of CNN1-Fusion3 compared to CNN0-Fusion3 on the LITIS Rouen dataset. Removal of background noise, however, plays an even more important role than before. It is not only because CNN0-Fusion3 outperforms its counterpart CNN1-Fusion3 on the DCASE2016 dataset, but also because it helps to leverage the performance of CNN-Fusion6 over both datasets. We actually saw negative results previously with LTE-Fusion6 in Section VI-C1. CNN-Fusion6 outperforms all other single-stream CNNs under this classification scheme. The reason is that stacking all six LTE images enforces the CNNs to learn more robust templates across all LTE channels.

Multi-stream CNN fusion schemes lead to performance gains compared to their individual constituents in most of the cases, but their performance is not comparable with that of CNN-Fusion6. Nevertheless, the rule of thumb is that combination of various feature types is necessary, if not vital, to guarantee good performance.

Although the performance fluctuates for different numbers of filters Q, the variation is small. For instance, with CNN-Fusion6 on the LITIS Rouen dataset, the difference between the peak (96.5% at $Q = 500$) and the worse case (96.1% at

TABLE VI
DCASE2016 DATASET. PERFORMANCE COMPARISON

Systems	Accuracy
LTE-Gam	**75.9**
LTE-MFCC	**73.6**
LTE-Log	**75.5**
LTE0-Fusion3	**75.3**
LTE1-Fusion3	**75.3**
LTE-Fusion6	**77.6**
CNN-Gam	**76.9**
CNN-MFCC	**76.7**
CNN-Log	**77.6**
CNN-Multi-Mean	**79.5**
CNN-Multi-Max	**79.0**
CNN0-Fusion3	**79.1**
CNN1-Fusion3	**78.3**
CNN-Fusion3-Multi-Mean	**78.5**
CNN-Fusion3-Multi-Max	**78.5**
CNN-Fusion6	**81.2**
DCASE2016 baseline [20]	72.5

TABLE VII
LITIS ROUEN DATASET. PERFORMANCE COMPARISON

Systems	Precision	F1-score	Accuracy
LTE-Gam	**94.5**	94.7	94.9
LTE-MFCC	93.0	93.3	93.5
LTE-Log	**94.3**	94.5	94.5
LTE0-Fusion3	92.2	92.4	92.6
LTE1-Fusion3	**95.5**	**95.7**	95.8
LTE-Fusion6	**94.7**	95.0	95.2
CNN-Gam	**95.5**	**95.8**	95.8
CNN-MFCC	**93.4**	93.7	94.0
CNN-Log	**94.7**	95.0	95.1
CNN-Multi-Mean	**95.7**	**96.0**	**96.0**
CNN-Multi-Max	**95.5**	**95.8**	95.9
CNN0-Fusion3	92.2	92.6	92.9
CNN1-Fusion3	**96.1**	**96.3**	**96.3**
CNN-Fusion3-Multi-Mean	**95.3**	**95.7**	95.8
CNN-Fusion3-Multi-Max	**95.5**	**95.8**	95.8
CNN-Fusion6	**96.3**	**96.5**	**96.6**
HOG [11]	91.7	–	–
DNN+MFCC [8]	92.2	–	–
HOG+SPD [12]	93.3	92.8	93.4
Sparse NMF [55]	–	94.1	–
Convolutive NMF [55]	–	94.5	–
Kernel PCA [55]	–	95.6	–
FisherHOG+ProbSVM [13]	–	–	96.0

$Q = 100$) is only 0.4%. This implies that an arbitrarily chosen Q can guarantee a good classification accuracy. In general, for more complex datasets, we need larger number of filters to achieve the best performance. For example, $Q = 400$ seems to be reasonable for the DCASE2016 dataset while $Q = 500$ is most suitable for the LITIS Rouen dataset. A Q larger than these optimal values results in redundancy of the filter set which brings up little help if not degenerating the performance.

3) Performance Comparison: For the sake of comparison, we present the performance of our systems on the DCASE2016 ($Q = 400$) and LITIS Rouen ($Q = 500$) datasets together with other results reported in the literature in Tables VI and VII, respectively. Note that we only include those of our systems with multiple LTE features for clarity. The results for our classification systems are marked in bold when all competitors are outperformed. Since prior works reported their performances on the LITIS Rouen dataset with different metrics (i.e. average class-wise precision [8], [11], F1-score [12], [55], and overall accuracy [12], [13]), the performances of our systems are also provided on all of these metrics for a proper comparison. Note that the state-of-the-art performance on the LITIS Rouen dataset is reported in our recent work [23]. However, it was achieved with the augmentation of external speech data. Here, we focus on studying the representative power of the scene audio signals per se. For the case of DCASE2016, we employ the baseline provided by the challenge for comparison [20].

TABLE VIII
PERFORMANCE WITH DIFFERENT SEGMENT SIZES

	Systems	250 ms	500 ms
DCASE2016	*LTE-Fusion6*	77.6	77.0
	CNN-Fusion6	81.2	80.2
LITIS Rouen	*LTE-Fusion6*	95.0	95.3
	CNN-Fusion6	96.5	96.1

As can be seen, while our LTE fusion systems surpass the competitors in most of the cases, the CNN systems even perform better, being superior over all the opponents on both datasets. For the DCASE2016 dataset, our systems consistently achieve better accuracies than that of the DCASE2016 baseline. The accuracy gains range from 2.8% with LTE0-Fusion3 to 8.7% with CNN-Fusion6. A similar system submitted for Task1 of the DCASE2016 challenge achieved an accuracy of 83.3% on the test data, which is ranked 14 of 35 submissions. Note that, different from the classification scheme described here, this submission system directly used the softmax layer of the trained CNNs for classification. For the LITIS Rouen dataset, our systems show better performance than most of, if not all, the compared systems. Moreover, CNN-Fusion6 yields top performance on all evaluation metrics and outperforms the best reported results by 3.0%, 0.9%, and 0.6% absolute in terms of precision, F1-score, and accuracy, respectively.

VII. DISCUSSION

A. *Influence of the Segment Size*

In the experiments in Section VI, we fixed the segment size to 250 ms with a step size of 125 ms. It is worth studying how the segment size influences the overall classification performance, taking LTE-Fusion6 and CNN-Fusion6 for example. We doubled the segment size, i.e. 500 ms and a step size of 250 ms, and repeated experiments on these systems. We compare the performance obtained with two different segment sizes in Table VIII. It can be seen that with the shorter segment sizes we achieve better performance than with the larger ones, except for the LTE-Fusion6 system on the LITIS Rouen dataset, most likely due to the drawback of the average pooling. The performance gains obtained by LTE-Fusion6/CNN-Fusion6 on DCASE2016 and LITIS Rouen are 0.6%/1.0% and −0.3%/0.5%, respectively. The benefits of using shorter segments are two-fold. Firstly, we

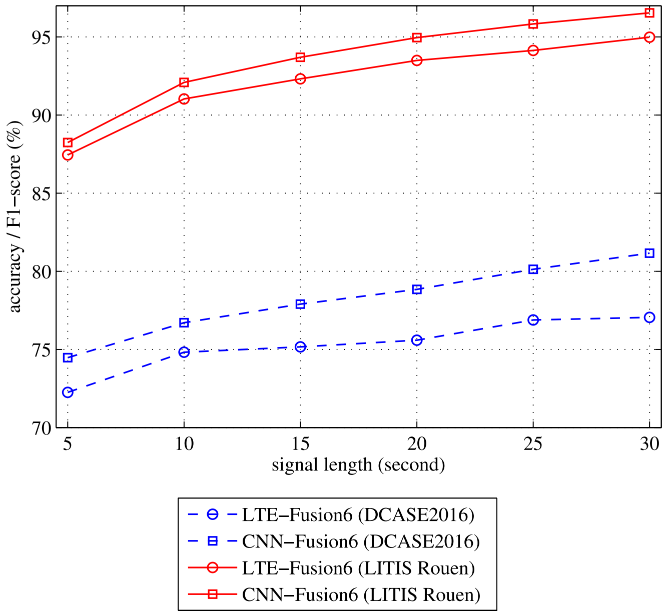


Fig. 7. Classification performance as a function of the length of the test signals.

have more training examples which benefit the LTE representation learning algorithm. Secondly, they will result in larger LTE images which leverage the following averaging/max pooling. However, the segment size should not be too short since then we focus on too much detail of the signals, causing unreliable estimation of the posterior probabilities by the random forest classifier used during LTE feature learning.

B. *Early Recognition*

We also study the possibility that a scene instance can be recognized early, i.e. when a recording less than 30 seconds of the scene is observed. Such early recognition ability is an important property to guarantee the quality-of-service, especially for safety-related applications. Although audio signals are usually provided with a fixed length of 30 seconds [9], [11], [20], we think this should not be strict. With a long signal we expect to accumulate more statistics about the scene and hence gain reliability in recognition, however, this observation relaxes for different kinds of scenes. For instance, for "office" scenes where foreground events are sparse and irregular, the recordings should be long. In contrast, for "busy street" ones, shorter signals may be advantageous. Since investigating this aspect for every scene category would be too demanding and out of scope of this work, we study here the overall classification performance for simplicity.

We, again, employed the LTE-Fusion6 and CNN-Fusion6 systems in this study. We utilized the systems trained on full 30-second long signals to evaluate on test signals with different lengths of $\{5, 10, 15, 20, 25, 30\}$ seconds. We show variations of the classification accuracy in Fig. 7. Note that the duration difference of the training and test signals is not a problem here since the average pooling produces fixed-size global feature vectors for them all in the LTE-Fusion6. In addition, the CNNs can handle input signals with varying lengths thanks to the 1-max pooling scheme [24]. As expected, the overall trend can be clearly seen that the accuracy grows with the signal length. It is due to the fact that with longer signals the systems not only know more about the scenes but also experience less mismatch between training and test data. At 15 seconds, we are able to obtain an accuracy of more than 75% on DCASE2016 and an F1-score of more than 92% on LITIS Rouen. The good thing is that these performances are better or on par with previous works tested on full 30-second long signals (c.f. Tables VI and VII). That is, using our systems, one can recognize a scene 50% faster with only a small penalty in classification accuracy.

VIII. Conclusion

We presented an efficient approach to tackle the audio scene classification task. Our systems relies on label tree embedding image features automatically learned to encode the structure of the data. We studied scene classification using global feature vectors obtained from these images and analyzed the performance of different variants of these features learned from different low-level feature sets as well as their combination. An improved classification method was then introduced. Simple CNNs were trained on LTE images to learn templates that are useful for the classification task. Afterwards, the learned templates were matched on an input LTE image for feature extraction and the final classification was accomplished by linear SVMs. Two different settings were investigated: single-stream CNNs with stacked LTE images as well as multi-stream CNNs followed by probabilistic fusion. Experiments on the DCASE2016 and LITIS Rouen datasets show that the classification accuracies obtained by our systems outperform all the reported results in previous works. Furthermore, combination of various features with and without background noise is essential for a good performance. Finally, in this work we used random forest classifiers in the LTE learning algorithm. Alternatively, stronger classifiers, such as DNNs, can be further explored for this purpose. A high-quality classifier that is able to estimate the metaclass posterior probability more precisely is expected to improve the learned LTE features and, as a result, the subsequent processing steps.

References

[1] R. Radhakrishnan, A. Divakaran, and P. Smaragdis, "Audio analysis for surveillance applications," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2005, pp. 158–161.

[2] Y. Xu, W. J. Li, and K. K. Lee, *Intelligent Wearable Interfaces*. Hoboken, NJ, USA: Wiley, 2008.

[3] A. J. Eronen *et al.*, "Audio-based context recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 321–329, Jan. 2006.

[4] S. Chu, S. Narayanan, C.-C. J. Kuo, and M. J. Mataric, "Where am I? Scene recognition for mobile robots using audio features," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2006, pp. 885–888.

[5] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP J. Audio, Speech, Music Process.*, vol. 2013, 2013, Art. no. 1.

[6] D. Wang and G. J. Brown, *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. New York, NY, USA: Wiley-IEEE Press, 2006.

[7] R. F. Lyon, "Machine hearing: An emerging field," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 131–139, Sep. 2010.

[8] Y. Petetin, C. Laroche, and A. Mayoue, “Deep neural networks for audio scene recognition,” in *Proc. Eur. Signal Process. Conf.*, 2015, pp. 125–129.

[9] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, “Detection and classification of acoustic scenes and events,” *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct. 2015.

[10] X. Valero and F. Alías, “Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification,” *IEEE Trans. Multimedia*, vol. 14, no. 6, pp. 1684–1689, Dec. 2012.

[11] A. Rakotomamonjy and G. Gasso, “Histogram of gradients of time-frequency representations for audio scene classification,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 1, pp. 142–153, Jan. 2015.

[12] V. Bisot, S. Essid, and G. Richard, “HOG and subband power distribution image features for acoustic scene classification,” in *Proc. Eur. Signal Process. Conf.*, 2015, pp. 719–723.

[13] J. Ye, T. Kobayashi, M. Murakawa, and T. Higuchi, “Acoustic scene classification based on sound textures and events,” in *Proc. ACM Multimedia*, 2015, pp. 1291–1294.

[14] R. Mogi and H. Kasaii, “Noise-robust environmental sound classification method based on combination of ICA and MP features,” *Artif. Intell. Res.*, vol. 2, no. 1, pp. 107–121, 2013.

[15] S. Deng, J. Han, C. Zhang, T. Zheng, and G. Zheng, “Robust minimum statistics project coefficients feature for acoustic environment recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 8232–8236.

[16] D. Barchiesi, D. Giannoulis, D. Stowell, and M. Plumbley, “Acoustic scene classification: Classifying environments from the sounds they produce,” *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 16–34, May 2015.

[17] T. Heittola, A. Mesaros, A. J. Eronen, and T. Virtanen, “Audio context recognition using audio event histogram,” in *Proc. Eur. Signal Process. Conf.*, 2010, pp. 1272–1276.

[18] R. Cai, L. Lu, and A. Hanjalic, “Co-clustering for auditory scene categorization,” *IEEE Trans. Multimedia*, vol. 10, no. 4, pp. 596–606, Jun. 2008.

[19] D. P. W. Ellis, Gammatone-like spectrograms, 2009. [Online]. Available: http://www.ee.columbia.edu/ dpwe/resources/matlab/gammatonegram/

[20] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *Proc. EUSIPCO*, 2016, pp. 1128–1132.

[21] H. Phan, L. Hertel, M. Maass, R. Mazur, and A. Mertins, “Representing nonspeech audio signals through speech classification models,” in *Proc. Interspeech*, 2015, pp. 3441–3445.

[22] H. Phan, M. Maaß, R. Mazur, and A. Mertins, “Random regression forests for acoustic event detection and classification,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 1, pp. 20–31, Jan. 2015.

[23] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, “Label tree embeddings for acoustic scene classification,” in *Proc. ACM Multimedia*, 2016, pp. 486–490.

[24] H. Phan, L. Hertel, M. Maass, and A. Mertins, “Robust audio event recognition with 1-max pooling convolutional neural networks,” in *Proc. Interspeech*, 2016, pp. 3653–3657.

[25] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 4489–4497.

[26] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng, “Tiled convolutional neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 1279–1287.

[27] A. Coates, A. Y. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[28] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proc. Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 568–576.

[29] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue, “Evaluating two-stream CNN for video classification,” in *Proc. 5th ACM Int. Conf. Multimedia Retrieval*, 2015, pp. 435–442.

[30] G. Roma, W. Nogueira, and P. Herrera, “Recurrence quantification analysis features for environmental sound recognition,” in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2013, pp. 1–4.

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

[32] S. Ağcaer, A. Schlesinger, F.-M. Hoffmann, and R. Martin, “Optimization of amplitude modulation features for low-resource acoustic scene classification,” in *Proc. Eur. Signal Process. Conf.*, 2015, pp. 2556–2560.

[33] N. Hansen, “The CMA evolution strategy: A comparing review,” in *Towards a new Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Berlin, Germany: Springer, 2006, pp. 75–102.

[34] J.-J. Aucouturier, B. Defreville, and F. Pachet, “The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music,” *J. Acoust. Soc. Amer.*, vol. 122, pp. 881–891, 2007.

[35] K. Lee, Z. Hyung, and J. Nam, “Acoustic scene classification using sparse feature learning and event based pooling,” in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2013, pp. 1–4.

[36] R. Martin, “Noise power spectral density estimation based on optimal smoothing and minimum statistics,” *IEEE Trans. Speech Audio Process.*, vol. 9, no. 5, pp. 504–512, Jul. 2001.

[37] Y. Aytar, C. Vondrick, and A. Torralba, “Soundnet: Learning sound representations from unlabeled video,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 892–900.

[38] H. Phan, L. Hertel, M. Maass, R. Mazur, and A. Mertins, “Learning representations for nonspeech audio events through their similarities to speech patterns,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 4, pp. 807–822, Apr. 2016.

[39] [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/

[40] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, “DCASE 2016 acoustic scene classification using convolutional neural networks,” in *Proc. Workshop Detection Classif. Acoust. Scenes Events*, Sep. 2016, pp. 95–99.

[41] Y. Han and K. Lee, “Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification,” DCASE 2016 Challenge, Tech. Rep., Sep. 2016.

[42] L. Hertel, H. Phan, and A. Mertins, “Classifying variable-length audio files with all-convolutional networks and masked global pooling,” DCASE 2016 Challenge, Tech. Rep., Sep. 2016.

[43] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, “CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks,” DCASE 2016 Challenge, Tech. Rep., Sep. 2016.

[44] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1746–1751.

[45] L. Breiman, “Random forest,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[46] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 849–856.

[47] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *Proc IEEE Conf. Comput. Vision Pattern Recognit.*, 2008, pp. 1–8.

[48] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011, pp. 315–323.

[49] A. Severyn and A. Moschitti, “Twitter sentiment analysis with deep convolutional neural networks,” in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 959–962.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[51] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

[52] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

[53] T.-F. Wu, C.-J. Lin, and R. C. Weng, “Probability estimates for multi-class classification by pairwise coupling,” *J. Mach. Learn. Res.*, vol. 5, pp. 975–1005, 2004.

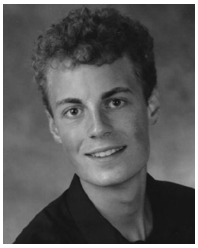
[54] J. Platt, “Probabilistic outputs for support vector machines, and comparisons to regularized likelihood methods,” in, *Advances in Large Margin Classifiers*. Cambridge, MA, USA: MIT Press, 1999.

[55] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Acoustic scene classification with matrix factorization for unsupervised feature learning,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6445–6449.

Huy Phan (S'13) received the B.Sc. degree in computer science from the University of Science, Ho Chi Minh City, Vietnam, in 2007, and the M.Eng. degree in computer engineering from Nanyang Technological University, Singapore, in 2012. He is currently working toward the Ph.D. degree at the Graduate School for Computing in Medicine and Life Sciences, University of Lübeck, Lübeck, Germany. From 2012 to 2013, he was a Lecturer with the University of Information Technology, Ho Chi Minh City. He is a Research Associate at the Institute for Signal Processing, University of Lübeck. His research interests include audio signal processing, biosignal processing, and machine learning, with a special focus on audio event detection and audio scene recognition.

Lars Hertel received the B.Sc. and M.Sc. degrees (Hons.) in computer science in 2012 and 2014, respectively, from the University of Lübeck, Lübeck, Germany, where he is currently working toward the Ph.D. degree. He is a Research Associate at the Institute of Signal Processing, University of Lübeck. His research interests include deep learning for machine hearing and computer vision, with a special focus on acoustic event detection.

Marco Maass (S'13) received the B.Sc. and M.Sc. degrees in computer science in 2010 and 2012, respectively, from the University of Lübeck, Lübeck, Germany, where he is currently working toward the Ph.D. degree at the Graduate School for Computing in Medicine and Life Sciences. He is a Research Associate at the Institute for Signal Processing, University of Lübeck. His research interests include machine learning, filter design, and image processing, with a special focus filter bank design, MRI reconstruction, and MPI reconstruction.



Philipp Koch received the B.Sc. and M.Sc. degrees in medical engineering science in 2013 and 2015, respectively, from the University of Lübeck, Lübeck, Germany, where he is currently working toward the Ph.D. degree. He is a Research Associate at the Institute for Signal Processing, University of Lübeck. His research interests include machine learning, biosignal analysis, audio/acoustic signal processing, and sparse MRI reconstruction.

Radoslaw Mazur (S'09–M'11) was born in Wroclaw, Poland, in 1976. He received the Diplominformatiker degree from the University of Oldenburg, Oldenburg, Germany, in 2004, and the Dr.-Ing. degree in computer science from the University of Lübeck, Lübeck, Germany, in 2010. He was an Assistant Researcher in the Department of Physics, University of Oldenburg, from 2004 to 2006, and then joined the University of Lübeck. The current research interests are digital signal and audio processing, with a special focus on blind source separation.

Alfred Mertins (M'96–SM'03) received the Dipl.-Ing. degree from the University of Paderborn, Paderborn, Germany, in 1984, the Dr.-Ing. degree in electrical engineering and the Dr.-Ing. Habil. degree in telecommunications from Hamburg University of Technology, Hamburg, Germany, in 1991 and 1994, respectively. From 1986 to 1991, he was a Research Assistant with Hamburg University of Technology, and from 1991 to 1995, he was a Senior Scientist with the Microelectronics Applications Center Hamburg, Germany. From 1996 to 1997, he was with the University of Kiel, Germany, and from 1997 to 1998, with the University of Western Australia, Crawley, WA, Australia. In 1998, he joined the University of Wollongong, Wollongong, NSW, Australia, where he was an Associate Professor of electrical engineering. From 2003 to 2006, he was a Professor with the Faculty of Mathematics and Science, University of Oldenburg, Oldenburg, Germany. In November 2006, he joined the University of Lübeck, Lübeck, Germany, where he is a Professor and the Director of the Institute for Signal Processing. His research interests include speech, audio, and image processing; wavelets and filter banks; pattern recognition; and digital communications.