A convolutional neural network approach for acoustic scene classification

A Convolutional Neural Network Approach for Acoustic Scene Classification

Michele Valenti, Stefano Squartini
Università Politecnica delle Marche
Department of Information Engineering
Ancona, Italy
Email: valenti.michele.w@gmail.com

Aleksandr Diment, Giambattista Parascandolo
and Tuomas Virtanen
Tampere University of Technology
Department of Signal Processing
Tampere, Finland

***Abstract*—This paper presents a novel application of convolutional neural networks (CNNs) for the task of acoustic scene classification (ASC). We here propose the use of a CNN trained to classify short sequences of audio, represented by their log-mel spectrogram. We also introduce a training method that can be used under particular circumstances in order to make full use of small datasets. The proposed system is tested and evaluated on three different ASC datasets and compared to other state-of-the-art systems which competed in the "Detection and Classification of Acoustic Scenes and Events" (DCASE) challenges held in 2016[1] and 2013. The best accuracy scores obtained by our system on the DCASE 2016 datasets are 79.0% (development) and 86.2% (evaluation), which constitute a 6.4% and 9% improvements with respect to the baseline system. Finally, when tested on the DCASE 2013 evaluation dataset, the proposed system manages to reach a 77.0% accuracy, improving by 1% the challenge winner's score.**

I. Introduction

When we talk of acoustic scene classification (ASC) we refer to the ability of a human or an artificial system to recognize an *audio context*, either from an on-line stream or from a recording. "Context" or "scene" are concepts that humans commonly use to identify a particular *acoustic environment*, *i.e.,* the ensemble of background noises and sound events that we are used to associate with a specific audio scenario, like a restaurant or a park. For humans this may look like a simple task: complex calculations that our brain is able to perform and our extensive life experiences allow us to easily associate these ensembles of sounds with specific scenes. However, this task is not trivial for artificial systems.

Falling in the field of studies known as *computational auditory scene analysis* (CASA), ASC was firslty defined in Bregman's work [2], in 1994. Ten years later, a collection of important theoretical papers [3] was published with the purpose of deeply analyzing CASA, especially by addressing issues such as the "mixture problem" [4] (*i.e.,* the problem of recovering an individual source from an ensemble of overlapping sounds). Practical interest in computational ASC lies in its many possible applications, therefore some applicative analyses were published throughout the years. For example, in [5] the authors aim at defining a "context-aware" computation paradigm, in particular by examining how a computer can became able to react and reconfigure itself by smartly monitoring the nearby environment. Sequent works also analyzed the use of ASC applied to intelligent wearable interfaces configuration [6] or mobile robot navigation enhancement [7].

In the field of machine learning different models and audio feature representations have been proposed to deal with the ASC task. In particular, examples of classifiers featuring neural networks go back to 1997 [8] and 1998 [9]. In [8] a recurrent neural network is combined with a nearest neighbour classifier to discriminate five different environmental sounds, whereas in [9] a neural network is used to recognize five different types of TV shows. Despite the fact that new systems have been proposed throughout the years, the interest towards ASC has raised especially thanks to the DCASE 2013 challenge [10]. In particular, 18 different systems were evaluated for the DCASE 2013 ASC task, some examples being based on Gaussian mixture models (GMMs) [11], support vector machines [12] and tree bagger classifiers [13].

Nowadays, application of CNNs for audio-related tasks is becoming more and more widespread. In [14] a speech recognition task is addressed and a CNN approach is evaluated on two different datasets. In addition, other examples highlight how CNNs can reach a very high performance also on tasks such as environmental sound classification [15] or robust audio event recognition [16]. To the best of our knowledge no former works addressing ASC with CNNs were presented before the DCASE 2016 challenge, for the occasion of which this study was realized.

The system proposed in this paper has been specifically studied for ASC. We designed a CNN able to work with short audio sequences, characterized by their log-mel spectrogram features. Each input spectrogram is classified by the CNN as belonging to one of 15 different acoustic scenes, as provided by the ASC task setup. Moreover, at training time we make use of batch normalization, a recently-proposed technique developed to speed up and regularize the network training. In addition, we propose a training procedure that allows the classifier to achieve a good generalization performance on both the development and evaluation datasets. Therefore, we here propose a system capable of improving the baseline

[1] A copyright-free and preliminary version of this paper [1] has been presented at DCASE 2016 workshop, Budapest, Hungary.

system, represented by a GMM [17] classifier, and to give a novel contribution for future development in the use of neural networks for the ASC task.

The outline of this paper is the following: in Section II we give a brief background about CNNs, then, in Section III, we describe in detail the system proposed for the DCASE 2016 ASC task. In Section IV we present some comparisons between different system configurations and other neural network classifiers, together with results obtained with the older DCASE 2013 evaluation dataset. Finally, in Section V we summarize the main results and draw our conclusions.

II. Convolutional neural networks

In this section we aim to give a basic overview of how a typical CNN works, introducing some of the main keywords that are useful for its comprehension. For a better understanding of what follows hereafter it is also possible to refer to Fig. 2, in which we show the proposed CNN's model that, on the other hand, will be analysed in detail in Section III.

In a CNN inputs are processed by small computational units (neurons) organized in a layered structure. A neuron commonly operates by applying a non-linear filtering to a weighted sum of its inputs, with the weights being the network parameters learned during its training. Some of the most common non-linear functions used in a neuron are the tanh, the sigmoid or the rectifier function, the last giving name to neurons called *rectifier linear units* (ReLUs) [18].

The most remarkable feature that distinguish CNNs as a particular subset of feed-forward neural networks is the presence of *convolutional layers*. These layers are formed by neurons (also called *kernels*) that sequentially perform non-linear filtering operations by shifting a small context window (*i.e.,* the kernel's *receptive field*) on the layer's input, therefore computing a series of activations as the window sequentially covers it. This localized filtering is a characteristic known as *local connectivity* and it represents the key feature for obtaining invariance against shifts of recognizable input patterns. It is precisely this kind of invariance that is considered to be a desirable (if not essential) feature in addressing a wide variety of problems. In image recognition, for example, it is crucial that a target object is recognized regardless of its position in the input picture; similarly, in audio-related tasks, it can be desirable that an energy pattern in a spectrogram is recognized regardless of its time (or even frequency) position. Finally, activations computed by each kernel are collected in matrices that we call *feature maps*, which represent the actual convolutional layer's output.

Due to the small dimension of receptive fields, the network manages to achieve shift invariance at the cost of losing the overall overview of the original input. This issue can be addressed with the sequential stacking of more convolutional layers, but, in order to quickly achieve wider overviews, it is common to use a *pooling layer* between two sequential convolutional ones. This kind of layer essentially subsamples each feature map, hence reducing its dimension, and the most common way it acts is by simply extracting the highest value from a window that slides along each feature map (*i.e., max-pooling*). It is thanks to pooling layers that the network is able to learn higher level features on wider input overviews without the need of too many convolutional layers, which, in some cases, can result in overfitting issues, or lead to a too large number of parameters to be trained.

The last layer of a CNN is the one that will output the actual network's prediction. Unlike previous layers, it is usually composed of *fully-connected* neurons, so that each of them will take as input the whole previous layer's output at once. For classification problems it is common to use the softmax function as neurons' nonlinearity, therefore associating each of the possible classes to a specific neuron.

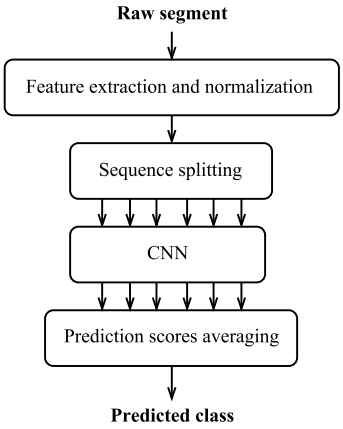

Fig. 1. Block diagram of the proposed method

III. Proposed method

In this section we describe our method and the architecture chosen for the proposed system. In Fig. 1 we show that the raw audio segment is firstly processed so to extract its feature representation, which is then normalized. Then, the processed file is split into short classification units which are fed one by one to the CNN. Finally, the label for the entire segment is chosen according to the average of scores obtained for each classification unit.

A. Feature representation and preprocessing

Despite new mid or high-level feature representations are frequently designed and analysed for audio-related purposes (*e.g.,* speech recognition [19], [20], sound event detection [21], or music information retrieval [22]), we expect that a CNN will be able to autonomously learn new features based on relevant patterns occurring in the training data. Hence, the feature representation we choose for our system is the log-mel spctrogram, which, in addition to being a convenient image-like (*i.e.,* a two-dimensional matrix) input for a CNN, is also a well-established and perceptually motivated representation of the audio spectra.

In order to extract the log-mel features, we firstly apply a short-time Fourier transform (STFT) over windows of 40 ms of audio with 50% overlap and Hamming windowing. We then

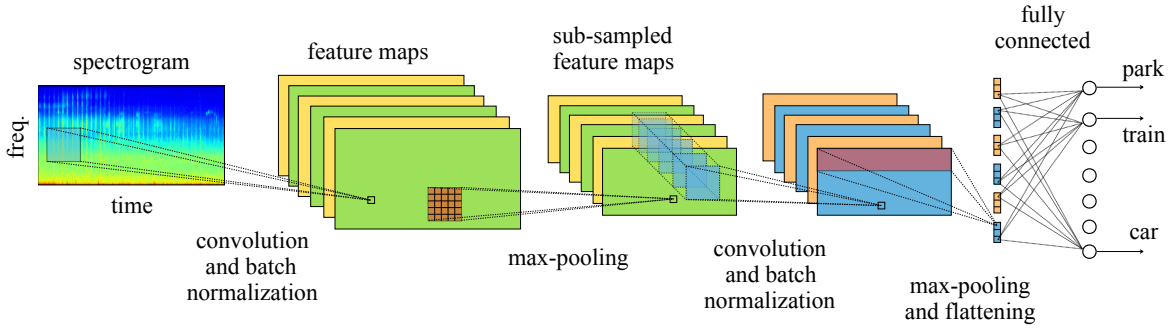

Fig. 2. Block scheme of the used convolutional neural network. Max-pooling windows are represented in red, kernel receptive fields are in light blue

square the absolute value of each bin and apply a 60-band mel-scale filter bank. Finally, the logarithmic conversion of the mel energies is computed. The whole feature extraction process has been implemented in Python with the support of the librosa [23] library.

After the extraction, we normalize each bin by subtracting it its mean and dividing it by its standard deviation, both calculated on the dataset used for the network's training. Then, we split normalized spectrograms into shorter, non-overlapping spectrograms, which we will call *sequences* hereafter.

B. Proposed system

The proposed model consists of a deep CNN and is represented in Fig. 2. The parameters we report here are chosen as a result of preliminary experiments conducted on the DCASE 2016 ASC development dataset with the aim of finding the best performing configuration in terms of number of kernels and receptive field areas.

The first CNN layer performs a convolution over the input spectrogram with 128 ReLU kernels characterized by 5×5 receptive fields and unitary depth and stride in both dimensions. The obtained feature maps are then subsampled with a max-pooling layer, which operates over 5×5 non-overlapping squares. Then, the second convolutional layer is the same as the first one, with the difference that more ReLU kernels (256) are used in order to grant higher level representation. The second and last subsampling is then performed aiming to "destruct" the time axis. In doing so, we use a max-pooling layer which operates over the entire sequence length and, on the frequency axis, only over four non-overlapping frequency bands. Finally, since the classification involves 15 different classes, the last is a softmax layer composed of 15 fully-connected neurons.

The classification for the whole segment is achieved by averaging all prediction scores obtained for its respective sequences. If we define the CNN output $\mathbf{y}^{(i)}$ as a vector containing all class-wise prediction scores for the i^{th} sequence, the predicted class c^* for the whole segment is calculated as:

$$c^* = \arg\max_c \left[\frac{1}{M} \sum_{i=1}^{M} y_c^{(i)} \right], \qquad (1)$$

where M is the number of sequences into which the segment is split and $y_c^{(i)}$ is the c^{th} entry of $\mathbf{y}^{(i)}$. In other words, the predicted class is the position of the maximum entry y_{c^*} in the vector given by the average of all prediction vectors obtained for each sequence.

The neural network is implemented with the Keras library (vers. 1.0.4) [24] for Python and its training is performed with an Nvidia Tesla K80 GPU showing an average training time of 50 seconds per epoch. The loss function we use for training is the categorical cross-entropy and the optimization algorithm we choose for its minimization is the adaptive momentum (adam) [25]. Based on preliminary experiments, we propose to use the optimizer default parameter configuration.

C. Regularization and model training

Batch normalization, introduced in [26], is a technique that addresses the issue described by Shimoidara *et al.* [27], known as internal covariate shift. This problem araises during the network training due to the variation that weight's statistics inevitably have when they are updated. Because of this variation, neurons in the following layers are forced to adapt their own weigths to these shifts, therefore slowing the overall training speed.

Batch normalization acts as an intermediate layer with the purpose of whitening the output of the previous layer. Practically, a batch normalization layer acts by applying a linear transformation $\text{BN}_{\beta,\gamma}$ to its input x as follows:

$$\text{BN}_{\beta,\gamma}(x) = \frac{\gamma}{\sqrt{\text{Var}[x] + \epsilon}} \cdot x + \left(\beta - \frac{\gamma \cdot \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} \right), \qquad (2)$$

where $\text{E}[x]$ and $\text{Var}[x]$ are the mean and the variance of the input, calculated for a batch of samples, whereas β and γ are the transformation parameters, which will be learned during training. We use batch normalization to normalize kernel outputs, therefore we have one γ and one β for each kernel. By using batch normalization we increase the model complexity, but, as shown in [26] and in our preliminary experiments, this practice can drastically reduce the training convergence time.

The proposed training method consists of two phases. The first, called *non-full training*, starts with a splitting of the whole training data into two subsets: one for training and one for validation, respectively consisting of 80% and 20% of

the original training data. Every epoch we collect the training spectrograms into class-wise feature lists so to randomly shuffle and time-shift all segment spectrograms before splitting them into sequences. Thanks to this procedure the splitting will always return slightly different sequence spectrograms, therefore increasing the network's input variability.

In order to monitor the network performance during training, we check the segment-wise performance on both training and validation sets once every five epochs. We believe that this checking period is a good compromise between a careful training monitoring and the overall training speed reduction given by the computation of all classification scores. Then, after each check, we save the network parameters if the segment-wise validation score has improved. Finally, we stop the training if no improvement is recorded after 100 epochs.

With this setup it is possible to notice that the segment-wise validation performance is prone to saturate. This means that the score starts to oscillate around a fixed, stable value. When this happens we say that the system has converged, thus allowing us to proceed to the second phase, which we call *full training*. In this phase we decide to retrain the network on all the training data, so now we must force the training to stop after a number of epochs fixed a priori. Considering that the bigger amount of training data is compensated by a usually higher convergence speed on training datasets, we choose this number of epochs to be equal to the convergence time of the non-full training phase. Therefore, we believe that a model trained this way will reach a convergence state without excessive overfit and making the best of all the available training data. In particular, this latter fact may turn out to be desirable especially when dealing with small datasets, such as in the DCASE 2016 ASC scenario.

IV. Evaluation

A. Dataset and metrics

For the system development and preliminary evaluations we use the DCASE 2016 [17] development dataset as provided at the beginning of the challenge. Given that only less than 1% of the total audio duration is affected by some mobile phone interferences, we do not make use of annotations about the corrupted segments, which were made public a few weeks after the first database release.

The dataset consists of 1170 audio segments of thirty seconds, equally distributed between the 15 different classes reported in Table I. Moreover, the challenge organizers also provided text files containing the annotated ground truth (*i.e.,* each file's name-class correct coupling), and the recommended data division designed for the cross-validation setup.

An important fact about the dataset is that it contains different groups of segments, each of which was obtained by splitting a longer audio file recorded in a single location. Due to this, it is important not to train and evaluate the network performance on segments coming from the same location, since this would falsify the generalization score. Because of this, we decide to use the training and test subsets recommended in the dataset annotations. The train/test split gives us approximatively 880 training segments for each fold, some folds having fewer segments. This is due to the fact that different long recordings have been split into a variable number of segments, ranging from three to ten. This means that the distribution of per-location segments is not uniform. Similarly to what was done for the recommended sets, for the training/validation split we decide which locations to use for validation and then select all segments coming from those locations.

TABLE I
Classes provided for the ASC Task of the DCASE 2016 Challenge

DCASE 2016 ASC classes		
beach	forest path	office
bus	grocery store	park
café/restaurant	home	residential area
car	library	train
city center	metro station	tram

During our development, the model was evaluated according to a four-fold cross-validation scheme. For the evaluation of each fold, per-class accuracies are initially calculated on the test set on a segment-wise level. These scores are obtained by dividing the number of correctly classified segments by the total number of segments belonging to that class. Then, scores for each fold are obtained by averaging all the 15 per-class accuracies. Finally, the overall score is calculated by averaging the four per-fold accuracies.

B. Classification accuracy and sequence length

As showed in [28], humans' ability to distinguish different scenes from acoustic information is best if we are able to listen to a long audio sequence (approximately 14 s). Hence, the main focus of this section is a comparison between accuracies obtained by our system when sequences of different lengths are used as classification unit. The average convergence time we estimate for all the different lengths is 200 epochs, hence we choose this number of epochs to be the full training period. Moreover, experiments with the full training configurations were repeated four times with the same system hyperparameters, but different random weight initializations. Thanks to this, in Table II we are able to report the average of the overall accuracies (and their standard deviations) obtained from these separate experiments.

Results shown in Table II appear to highlight that medium-length sequences, like three or five seconds, perform better than extremely short or long sequences. The best accuracy is achieved by the three-second configuration, with average accuracies of 75.9% and 79.0% obtained with non-full and full training respectively.

A deeper insight into which classes are mostly misclassified can be obtained by looking at the confusion matrix in Fig. 3, which we obtained by grouping results of all the four folds. What emerges is that some classes (*e.g.,* "park" and "residential area", or "bus" and "train") are often confused by the system. This may indicate that our model is relying

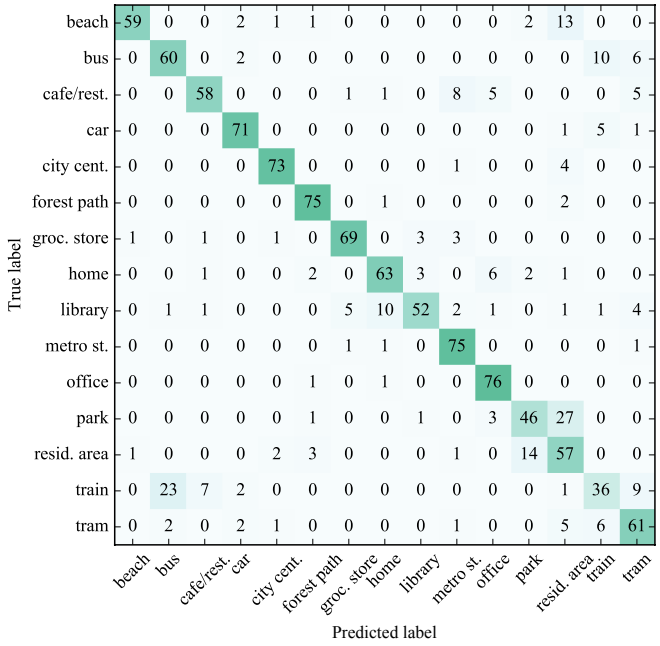

Fig. 3. Confusion matrix for the proposed CNN evaluated on the four folds. Three-second sequences are used

TABLE II
ACCURACY COMPARISON FOR THE PROPOSED MODEL TRAINED WITH DIFFERENT SEQUENCE LENGTHS (SEQ. LEN.) AND TRAINING MODES ("NON-FULL" AND "FULL"). STANDARD DEVIATIONS (S.D.) REFER TO FULL TRAINING ACCURACIES

seq. len. (s)	*accuracy* (%)		
	non-full	*full*	± *s.d.*
0.5	68.2	75.4	0.75
1.5	74.0	78.4	1.19
3	**75.9**	**79.0**	**0.68**
5	74.1	78.3	0.86
10	71.5	77.3	0.88
30	74.0	75.6	0.44

more on the background noise of the sequence rather then on acoustic event occurrences. We believe that this can also explain why classes with very similar background noises are confused even when 30-second sequences are used. Because of the results shown in Table II, we choose three seconds as the sequence length used for the evaluation on the challenge data.

C. Comparison of other systems

Here we report new comparisons with other systems that have been tested during our development. All parameter configurations are chosen in order to give each system a representation capacity that can be comparable with the proposed network's.

The first system we introduce is a multi-layer perceptron (MLP) with two hidden layers. The input layer consists of 900 nodes to which we feed log-mel features for a context window

TABLE III
ACCURACY COMPARISON FOR DIFFERENT SYSTEMS AND TRAINING MODES ("NON-FULL" AND "FULL"). NEURAL ARCHITECTURES ARE IDENTIFIED BY THEIR NUMBER OF HIDDEN LAYERS

system	*seq. len.* (s)	*accuracy* (%)	
		non-full	*full*
two-layer MLP (log-mel)	-	66.6	69.3
one-layer CNN (log-mel)	3	70.3	74.8
two-layer CNN (log-mel)	**3**	**75.9**	**79.0**
two-layer CNN (MFCC)	5	67.7	72.6
baseline GMM (MFCC)	-	-	72.6

of 15 frames. We then stack two hidden layers with 512 ReLU units (both layers being batch-normalized) and an output layer with 15 softmax neurons. The difference from the proposed network is that the output is now the class associated with the context window, which is sliding with unitary stride along the spectrogram. Segment-wise classification is performed as described in Eq. (1), where M now represents the number of frames within a segment. With the non-full training setup the proposed model achieves a 66.6% accuracy with a convergence time of 100 epochs. A full training for 100 epochs lets the model achieve a 69.3% accuracy.

The second model to which we compare the proposed CNN consists of a different CNN whose input is a three-second sequence. The input is processed by only one hidden convolutional layer with 2048 ReLU kernels whose receptive fields cover all the mel-energy bands and a context window of 15 frames. In this very first step the frequency dimension is narrowed, hence the output of the first layer is an ensemble of matrices with unitary height. Then, after a batch normalization layer, we stack a max-pooling layer so to shrink also the time dimension, hence obtaining a single number for each feature map. This structure is very similar to a MLP, since each kernel looks at all the mel-energy bands of a single context window at the same time. The key difference is in the max-pooling, which, similarly to the proposed model's, emphasizes the occurrence of a particular input pattern no matter where it appears in the sequence. This network achieves a 70.3% accuracy with the non-full training setup, which rises to 74.8% if a full training is performed for 100 epochs.

The next result we report compares our architecture with the baseline system when both are trained on mel-frequency cepstral coefficients (MFCCs). Details about the feature parameters are reported in [17]. The baseline system trains 15 different mixtures of 16 Gaussians in order to model each of the 15 classes. Classification is performed by comparing each mixture to the test audio file in terms of log-probabilities of the data under each model. The most similar mixture gives the chosen class. We choose a sequence length of five seconds for this comparison. Our model reaches a 67.7% overall accuracy with the non-full training setup and the average convergence time on the validation data is 100 epochs. The proposed system's performance reaches a 72.6% overall accuracy with

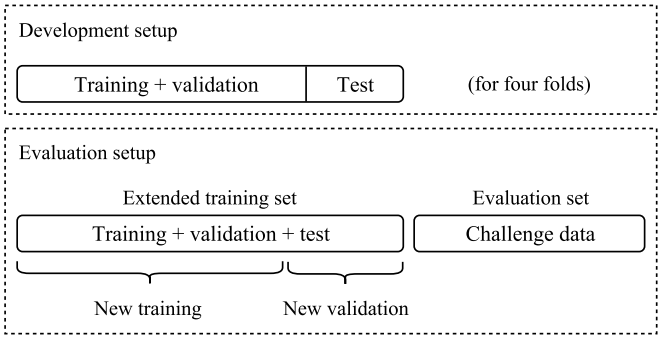

Fig. 4. DCASE 2016 dataset splitting for the development and evaluation phases

TABLE IV
ACCURACY COMPARISON FOR THE SEVEN BEST-PERFORMING SYSTEMS (OUT OF 49) TESTED ON THE DCASE 2016 EVALUATION DATASET

features	*system*	*accuracy* (%)
MFCC + spectrogram	fusion [29]	89.7
MFCC	i-vector [29]	88.7
spectrogram	NMF [30]	87.7
various features	fusion [31]	87.2
MFCC	i-vector [29]	86.4
various features	fusion [32], [33]	86.4
log-mel	**proposed CNN**	**86.2**

a full training setup, which equals the baseline accuracy.

D. Challenge results

For the final training, participants are recommended to use the whole development dataset, therefore including the previous test set in what we can consider as an *extended training set*, as we show in Fig. 4. By running a non-full training with this set we estimate the new convergence time to be 400 epochs. Then, we perform a full training for this period, and compute results for the evaluation dataset. With this configuration our model is able to reach a 86.2% accuracy score and to rank the sixth place in the ASC task of the DCASE 2016 challenge. For this task about 50 systems were submitted, so in Table IV we are able to report only the first six ranks — note that systems corresponding to the fifth and the sixth table entries tie for the fifth rank.

In Table IV we show that, among the first six ranks, the proposed system is the only one operating with a pure CNN approach. The second-best CNN system ranks the eighth place and it consists of a CNN ensemble [34] where each of the networks operates after an unsupervised (adequately pretrained) feature extraction layer.

In addition, it is interesting to notice that all the other best performing systems either are or can be considered as ensembles. Some of these ensembles (reported in Table IV as "fusion" systems) combine more than one classifier, going from i-vectors, introduced in [35], to GMMs, nearest neighbours and neural networks. Moreover, the i-vector stand-alone system is here considered as an ensamble, due to the fact that the classification is based on an average of prediction scores obtained from four different i-vectors (one for each audio channel, one for the channels' average and one for their difference). Finally, also the non-negative matrix factorization (NMF) approach [30], which features a slight modification to the usual learning algorithm [36], devolves the final score computation to an ensemble of four different realizations of the non-negative models.

TABLE V
ACCURACY COMPARISON BETWEEN DIFFERENT SYSTEMS TRAINED AND TESTED ON THE DCASE 2013 EVALUATION DATASET. REPORTED MEAN ACCURACIES AND STANDARD DEVIATIONS (S.D.) ARE CALCULATED ON ACCURACIES OBTAINED IN THE FIVE FOLDS

features	*system*	*accuracies* (%)	$\pm$ *s.d.*
log-mel	**proposed CNN**	**77.0**	**2.7**
various features	majority voting [37]	77.0	-
aug. mel-energy	CNN [38]	69.0	-
RQA$^{(*)}$ features	SVM [39]	76.0	7.2
wavelet and MFCC	tree bagger [40]	72.0	8.4
various features	SVM [12]	69.0	12.0
various features	HMM - GMM [11]	65.0	6.9
MFCC	baseline [10]	55.0	10.0

$^{(*)}$: recurrent quantification analysis.

E. DCASE 2013 evaluation

The DCASE 2013 evaluation dataset consists of 100 30-second audio segments equally divided into ten classes: "bus", "busy street", "office", "open air market", "park", "quiet street", "restaurant", "supermarket", "tube", and "tube station". Despite the dimension of this dataset being smaller than the 2016 challenge's, it is important to notice that each of the ten segments recorded for each class is now coming from a different recording, meaning that, in terms of location diversity, the two dataset dimensions are almost comparable.

In order to compare our results with the other challengers', the system evaluation is now based on the same five-fold cross-validation setup used during the 2013 challenge. According to this setup, each fold now contains 80 segments to be used for training and 20 for testing. At first, we train our model according to a non-full training setup, therefore keeping only 60 segments for the effective training. During this phase we encountered the same saturating behaviour described in Section III in all folds but one, where the validation accuracy kept increasing at a very slow rate until the training reached its end (arbitrarily fixed after 2500 epochs). Due to this, we decided to fix the number of full training epochs to 500.

In Table V we show a comparison between our system trained with full training setup and the best performing systems trained and tested on the DCASE 2013 evaluation dataset. On the upper part of the table we show systems that have been evaluated after the challenge was closed, whereas the bottom part reports the best performing 2013 challengers. From this comparison it emerges that our system manages to reach the highest score tying with an ensemble system based on the majority voting of all 2013 challengers' [37]. Another very

recent evaluation [38] has been carried out in 2016 for a CNN system in which DCASE 2016 challengers trained and tested their network also on the 2013 evaluation dataset. As shown in Table V, this system managed to achieve a 69.0% accuracy on the DCASE 2013 evaluation dataset.

Finally, we highlight that other works managed to effectively reach even higher accuracy scores. In [41] and [42] this was made possible by expanding the training set with the addition of the development one and in both works systems reach up to 86% accuracy scores. Moreover, in [43] a neural network is trained on a big quantity of unlabeled videos and then tested on the DCASE 2013 evaluation set reaching a 88% accuracy. Anyhow, due to such differences between training sets, we believe it is not sensible to compare our score to these systems'.

V. Conclusions

Our work proposes one out of many possible ways of approaching ASC with CNNs. In doing so we reach a 79.0% accuracy on the DCASE 2016 development dataset, demonstrating that a two-layer CNN can achieve higher accuracies if compared to a two-layer MLP (9.7% more), a one-layer CNN (4.2% more) and a GMM-MFCC system (6.4% more). We observed also that, under particular circumstances, training the network without monitoring its generalization performance can lead to a relevant accuracy improvement. This improvement is desirable especially when a low level feature representation is used, since the lack of training data can be a very narrow bottleneck for the network generalization performance.

Moreover, we showed that the proposed model managed to reach the sixth rank of the DCASE 2016 challenge, outperforming all other CNN-based systems and being the best performing system not relying on any ensemble technique. In addition to this, it is interesting to notice how the difference between scores obtained on the evaluation (86.2%) and development set (79.0%) highlights once more that the proposed model, following a common neural network behaviour, is able to take a lot of advantage from the introduction of more training data.

Finally, an evaluation of the proposed model on the DCASE 2013 evaluation dataset is reported. The achieved accuracy score is 77%, which represents a slight increase (1%) with respect to the previous challenge winner. This score is considered to be coherent with the reduced dimensionality of the DCASE 2013 dataset, but nonetheless it is a proof of the proposed system's ability to deal with different ASC datasets with a good degree of flexibility.

VI. Acknowledgments

The authors wish to acknowledge CSC — IT Center for Science, Finland, for generous computational resources.

References

[1] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Proc. of Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Budapest, Hungary, 2016.

[2] A. S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT press, 1994.

[3] P. Divenyi, *Speech Separation by Humans and Machines*. Springer Science & Business Media, 2004.

[4] M. Slaney, "The history and future of CASA," in *Speech Separation by Humans and Machines*. Springer, 2005, pp. 199–211.

[5] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proc. of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1994, pp. 85–90.

[6] Y. Xu, W. J. Li, and K. K. Lee, *Intelligent Wearable Interfaces*. John Wiley & Sons, 2008.

[7] 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. of IEEE International Conference on Multimedia and Expo*, 2006, pp. 885–888.

[8] N. Sawhney and P. Maes, "Situational awareness from environmental sounds," *MIT Media Lab Technical Report*, 1997.

[9] Z. Liu, Y. Wang, and T. Chen, "Audio feature extraction and analysis for scene segmentation and classification," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 20, no. 1-2, pp. 61–79, 1998.

[10] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An IEEE AASP challenge," in *Proc. of IEEE Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013, pp. 1–4.

[11] M. Chum, A. Habshush, A. Rahman, and C. Sang, "IEEE AASP scene classification challenge using hidden markov models and frame based classification," 2013.

[12] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *Proc. of IEEE Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013, pp. 1–4.

[13] E. Olivetti, "The wonders of the normalized compression dissimilarity representation," 2013.

[14] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *INTERSPEECH*, 2013, pp. 3366–3370.

[15] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proc. of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2015, pp. 1–6.

[16] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," *arXiv preprint arXiv:1604.06338*, 2016.

[17] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," *Proc. of the 24th Acoustic Scene Classification Workshop 2016 European Signal Processing Conference (EUSIPCO)*, 2016.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. of International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[19] S. Squartini, E. Principi, R. Rotili, and F. Piazza, "Environmental robust speech and speaker recognition through multi-channel histogram equalization," *Neurocomputing*, vol. 78, no. 1, pp. 111–120, 2012.

[20] D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, J. Abramson, R. Mihaescu *et al.*, "The supersid project: Exploiting high-level information for high-accuracy speaker recognition," in *Proc. of IEEE International Conference onAcoustics, Speech, and Signal Processing*, vol. 4, 2003, pp. IV–784.

[21] D. A. Sadlier and N. E. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *Proc. of IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1225–1233, 2005.

[22] F. Pachet and A. Zils, "Evolving automatically high-level music descriptors from acoustic signals," in *Proc. of International Symposium on Computer Music Modeling and Retrieval*. Springer, 2003, pp. 42–53.

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. of the 14th Python in Science Conference*, 2015.

[24] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate

shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[27] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[28] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *Proc. of IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

[29] 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," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[30] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Supervised nonnegative matrix factorization for acoustic scene classification," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[31] S. Park, S. Mun, Y. Lee, and H. Ko, "Score fusion of classification systems for acoustic scene classification," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[32] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, S. Squartini, and B. Schuller, "Pairwise decomposition with deep neural networks and multiscale kernel subspace learning for acoustic scene classification," in *Proc. of the 24th Acoustic Scene Classification Workshop 2016 European Signal Processing Conference (EUSIPCO)*, 2016.

[33] E. Marchi, D. Tonelli, X. Xu, F. Ringeval, J. Deng, and B. Schuller, "The up system for the 2016 DCASE challenge using deep recurrent neural network and multiscale kernel subspace learning," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[34] J. Kim and K. Lee, "Empirical study on ensemble method of deep neural networks for acoustic scene classification," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, September 2016.

[35] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *Proc. of IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[36] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[37] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *Proc. of IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[38] Y. Han and K. Lee, "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation," *arXiv preprint arXiv:1607.02383*, 2016.

[39] G. Roma, W. Nogueira, P. Herrera, and R. de Boronat, "Recurrence quantification analysis features for auditory scene classification," *Proc. of IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2013, tech. report.

[40] D. Li, J. Tam, and D. Toub, "Auditory scene classification using machine learning techniques," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2013.

[41] G. Vikaskumar, S. Waldekar, D. Paul, and G. Saha, "Acoustic scene classification using block based MFCC features," *Proc. of IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, September 2016.

[42] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, "Label tree embeddings for acoustic scene classification," in *Proc. of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 486–490.

[43] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," *arXiv preprint arXiv:1610.09001*, 2016.