



Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)

Citation

Virtanen, T., Mesaros, A., Heittola, T., Diment, A., Vincent, E., Benetos, E., & Elizalde, B. M. (2017). Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017). Tampere University of Technology. Laboratory of Signal Processing.

Year

2017

Version

Publisher's PDF (version of record)

Link to publication

[TUTCRIS Portal](http://www.tut.fi/tutcris) (<http://www.tut.fi/tutcris>)

Take down policy

If you believe that this document breaches copyright, please contact cris.tau@tuni.fi, and we will remove access to the work immediately and investigate your claim.

Tuomas Virtanen, Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Emmanuel Vincent,
Emmanouil Benetos & Benjamin Martinez Elizalde (eds.)
**Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017
Workshop (DCASE2017)**



Tuomas Virtanen, Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Emmanuel Vincent, Emmanouil Benetos & Benjamin Martinez Elizalde (eds.)

Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)

This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit
<http://creativecommons.org/licenses/by/4.0/>

ISBN 978-952-15-4042-4

| | |
|--|--------------|
| Acoustic Scene Classification by Combining Autoencoder-based Dimensionality Reduction and Convolutional Neural Networks | 7-11 |
| Jakob Abeßer, Stylianos Ioannis Mimalakis, Robert Gräfe, Hanna Lukashevich | |
| Sound Event Detection Using Weakly Labeled Dataset with Stacked Convolutional and Recurrent Neural Network | 12-16 |
| Sharath Adavanne, Tuomas Virtanen | |
| Sequence to Sequence Autoencoders for Unsupervised Representation Learning From Audio | 17-21 |
| Shahin Amiriparian, Michael Freitag, Nicholas Cummins, Björn Schuller | |
| Nonnegative Feature Learning Methods for Acoustic Scene Classification | 22-26 |
| Victor Bisot, Romain Serizel, Slim Essid, Gaël Richard | |
| Convolutional Recurrent Neural Networks for Rare Sound Event Detection | 27-31 |
| Emre Cakır, Tuomas Virtanen | |
| The SINS Database for Detection of Daily Activities in a Home Environment Using an Acoustic Sensor Network | 32-36 |
| Gert Dekkers, Steven Lauwereins, Bart Thoen, Mulu Weldegebral Adhana, Henk Brouckx, Bertold Van den Bergh, Toon van Waterschoot, Bart Vanrumste, Marian Verhelst, Peter Karsmakers | |
| Acoustic Scene Classification by Ensembling Gradient Boosting Machine and Convolutional Neural Networks | 37-41 |
| Eduardo Fonseca, Rong Gong, Dmitry Bogdanov, Olga Slizovskaya, Emilia Gomez, Xavier Serra | |
| Acoustic Scene Classification Using Spatial Features | 42-45 |
| Marc C. Green, Damian Murphy | |
| Convolutional Neural Networks with Binaural Representations and Background Subtraction for Acoustic Scene Classification | 46-50 |
| Yoonchang Han, Jeongsoo Park, Kyogu Lee | |
| Audio Event Detection Using Multiple-input Convolutional Neural Network | 51-54 |
| Il-Young Jeong, Subin Lee, Yoonchang Han, Kyogu Lee | |
| DCASE 2017 Task 1: Acoustic Scene Classification Using Shift-invariant Kernels and Random Features | 55-58 |
| Abelino Jiménez, Benjamín Elizalde, Bhiksha Raj | |
| DNN-based Audio Scene Classification for DCASE 2017: Dual Input Features, Balancing Cost, and Stochastic Data Duplication | 59-63 |
| Jee-Weon Jung, Hee-Soo Heo, IL-Ho Yang, Sung-Hyun Yoon, Hye-Jin Shim, Ha-Jin Yu | |

| | |
|---|----------------|
| Neuroevolution for Sound Event Detection in Real Life Audio: a Pilot Study | 64-68 |
| Christian Kroos, Mark D. Plumley | |
| Combining Multi-scale Features Using Sample-level Deep Convolutional Neural Networks for Weakly Supervised Sound Event Detection | 69-73 |
| Jongpil Lee, Jiyoung Park, Sangeun Kum, Youngho Jeong, Juhan Nam | |
| Ensemble of Convolutional Neural Networks for Weakly-supervised Sound Event Detection Using Multiple Scale Input | 74-79 |
| Donmoon Lee, Subin Lee, Yoonchang Han, Kyogu Lee | |
| Rare Sound Event Detection Using 1D Convolutional Recurrent Neural Networks | 80-84 |
| Hyungui Lim, Jeongsoo Park, Yoonchang Han. | |
| DCASE2017 Challenge Setup: Tasks, Datasets and Baseline System | 85-92 |
| Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, Tuomas Virtanen | |
| Generative Adversarial Network Based Acoustic Scene Training Set Augmentation and Selection Using SVM Hyper-plane | 93-97 |
| Seongkyu Mun, Sangwook Park, David K. Han, Hanseok Ko | |
| Acoustic Scene Classification Based on Convolutional Neural Network Using Double Image Features | 98-102 |
| Sangwook Park, Seongkyu Mun, Younglo Lee, Hanseok Ko | |
| The Details That Matter: Frequency Resolution of Spectrograms in Acoustic Scene Classification | 103-107 |
| Karol J. Piczak | |
| Wavelets Revisited for the Classification of Acoustic Scenes | 108-112 |
| Kun Qian, Zhao Ren, Vedhas Pandit, Zijiang Yang, Zixing Zhang, Björn Schuller | |
| Deep Sequential Image Features for Acoustic Scene Classification | 113-117 |
| Zhao Ren, Vedhas Pandit, Kun Qian, Zijiang Yang, Zixing Zhang, Björn Schuller | |
| Multi-temporal Resolution Convolutional Neural Networks for Acoustic Scene Classification | 118-122 |
| Alexander Schindler, Thomas Lidy, Andreas Rauber | |
| Acoustic Scene Classification: From a Hybrid Classifier to Deep Learning | 123-127 |
| Anastasios Vafeiadis, Dimitrios Kalatzis, Konstantinos Votis, Dimitrios Giakoumis, Dimitrios Tzovaras, Liming Chen, Raouf Hamzaoui | |
| Audio Event Detection and Classification Using Extended R-FCN Approach | 128-132 |
| Kaiwu Wang, Liping Yang, Bin Yang | |

Acoustic Scene Classification Using Deep Convolutional Neural Network and Multiple Spectrograms Fusion

133-137

Zheng Weiping, Yi Jiantao, Xing Xiaotao, Liu Xiangtao, Peng Shaohu

Robust Sound Event Detection Through Noise Estimation and Source Separation Using NMF

138-142

Qing Zhou, Zuren Feng

ACOUSTIC SCENE CLASSIFICATION BY COMBINING AUTOENCODER-BASED DIMENSIONALITY REDUCTION AND CONVOLUTIONAL NEURAL NETWORKS

Jakob Abeßer, Stylianos Ioannis Mimalakis, Robert Gräfe, Hanna Lukashevich

Fraunhofer IDMT, Ilmenau, Germany

ABSTRACT

Motivated by the recent success of deep learning techniques in various audio analysis tasks, this work presents a distributed sensor-server system for acoustic scene classification in urban environments based on deep convolutional neural networks (CNN). Stacked autoencoders are used to compress extracted spectrogram patches on the sensor side before being transmitted to and classified on the server side. In our experiments, we compare two state-of-the-art CNN architectures subject to their classification accuracy under the presence of environmental noise, the dimensionality reduction in the encoding stage, as well as a reduced number of filters in the convolution layers. Our results show that the best model configuration leads to a classification accuracy of 75% for 5 acoustic scenes. We furthermore discuss which confusions among particular classes can be ascribed to particular sound event types, which are present in multiple acoustic scene classes.

Index Terms— Acoustic Scene Classification, Convolutional Neural Networks, Stacked Denoising Autoencoder, Smart City Networks

1. INTRODUCTION

Particularly in urban environments, various acoustic scenes such as road traffic and railway transport, construction sites, open air concerts, or sport events often cause noise pollution and lead to resident complaints. Following the idea of a smart city network, a distributed system for intelligent acoustic analysis allows to objectively identify causes of noise pollution to the local city administration. A more effective processing of the incoming noise complaints allows to better plan future events in the residential area(s) of the city. As part of the *StadtLärm* [1] (City noise) research project, we first focus on identifying the acoustic scene that causes a potential noise exposure. In the given application scenario, the classification models furthermore need to be robust towards unwanted background noises such as wind and rain. Secondly, we aim to measure the exposure of citizens to noise in different parts of the city based on the German technical guidelines for noise reduction (*TA Lärm* [2]).

In this paper, we present a distributed system for automatic scene classification, which consists of two units: i) the acoustic sensor units with microphones placed around the city and ii) the central server application unit, where the audio scene analysis is performed. On the acoustic sensor side, non-negative time-frequency patches are extracted from a continuously audio input stream. Due to mobile communication bandwidth restrictions, we reduce the dimensionality of the aforementioned patches using a deep denoising auto-encoder (DAE) [3]. The encoded information is transmitted to the central server unit, where the patches are reconstructed using the decoder part of the DAE. The reconstructed time-frequency patches are then used for the acoustic scene classification. Due to

project constraints, we focus on the five acoustic scene classes: i) music event, ii) sport event, iii) traffic, iv) roadworks, and v) public place. We compare two state-of-the-art systems based on Convolutional Neural Networks (CNN), recently proposed by Salamon and Bello [4] (**SB**) and Takahashi et al. [5] (**TAK**).

2. RELATED WORK

Several research projects investigated how to integrate intelligent audio analysis algorithms into smart city application scenarios. For instance, the LIFE+ project DYNAMAP focuses on noise measurement in road infrastructures [6], the EU FP7 EAR-IT project [7] investigated large-scale indoor and outdoor acoustic sensor networks, and the SONYC research project developed algorithms and devices for monitoring noise pollution in the urban environment of New York City[8, 4].

The application of Convolutional Neural Networks (CNN) led to state-of-the-art results in various image processing tasks. Consequently, CNNs were successfully adopted to audio recognition tasks such as speech recognition [9], music transcription [10], and environmental sound classification [11]. Most methods for acoustic scene classification apply CNNs to learn characteristic spectro-temporal patterns for different sound classes from the audio signal. Commonly used spectrogram representations are either perceptually or musically motivated. In [12] for example, Lidy and Schindler propose to apply the constant-Q transform as input to the network as it allows to analyze low and mid-low frequencies with a better time resolution compared to the commonly used mel-frequency spectrogram [4]. Similarly to image recognition tasks where the RGB channels of an image control the depth of the convolutional filters of the CNN, researchers in acoustic scene classification have proposed to incorporate additional features to that aim. More specifically, Piszak [11] has proposed to use the first-order derivative of the magnitude spectrogram as an additional depth dimension, while Takahashi et al. [5] proposed to use the first and the second order derivatives as input to the CNN.

The abovementioned methods vary regarding hyper-parameters such as the filter size, the stride of the pooling layers, the number of convolutional and fully-connected layers, as well as regularization techniques such as dropout or weight regularization. Takahashi et al. followed the idea of the VGG CNN architecture [13] and replaced larger convolution kernels (e.g. 5 x 5) by stacking pairs of layers with 3 x 3 kernels without intermediate pooling [5]. This approach leads to a reduction of the number of model parameters but at the same time to more expressive features due to the additional non-linearity. Lidy and Schindler proposed two parallel convolutional layers to separately capture relevant patterns in audio signals along frequency and time [12]. Other approaches, such as the two compared architectures **SB** and **TAK**, and the recently proposed stacked CNNs and recurrent neural networks (RNN) [14, 15] use

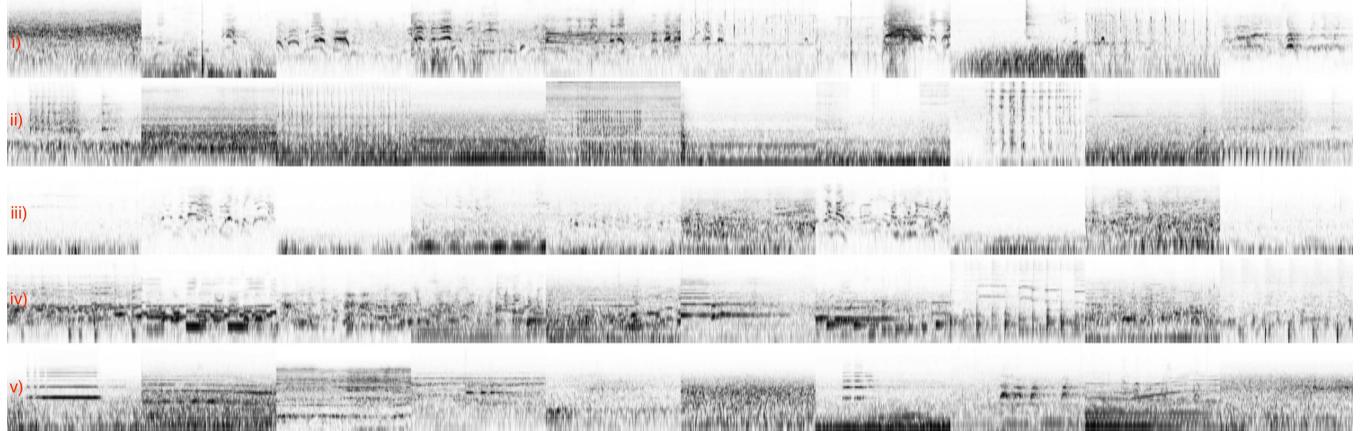


Figure 1: Examples of time-frequency patches of length three seconds, sorted row wise as follows: i) *sports event*, ii) *roadworks*, iii) *public place*, iv) *music event*, and v) *traffic*. The vertical and horizontal axes correspond to frequency and time, respectively.

successive layers instead. In [16], the authors combined deep denoising autoencoder architectures for feature learning in the context of acoustic event detection.

3. PROPOSED METHOD

3.1. System Overview

The proposed system architecture comprises of 25 distributed sensor units and one server unit as illustrated in Figure 3. On each sensor unit, the incoming audio stream is recorded at a sample rate of $f_s = 44.1$ kHz and processed with a short-time Fourier Transform (STFT) using a hop size of 882 (20 ms), a window size of 1024 (23.2 ms), and a zero-padding of factor 4. Then, the STFT magnitude spectrogram is logarithmically compressed and mapped to a logarithmically-spaced frequency axis of 49 bins (between 50 Hz and 15 kHz with a resolution of 6 bins per octave) using a triangular shaped filter-bank.

Spectrogram patches of size 49×50 (1 s duration) are reshaped to the dimensionality of 2450 and forwarded to the encoder part of the DAE described in Section 3.4. Then, the encoded patches are transmitted to the server unit where the decoder part of the DAE reconstructs the spectrogram patches and stores them in a buffer of size 3 seconds. Therefore, three consecutive patches are concatenated and forwarded to the classifier.

3.2. Dataset

Based on the given application scenario described in Section 1, we focus on the five acoustic scenes *sport event* (soccer games in stadium), *roadworks* (jackhammer, construction site), *public place* (conversations, walking), *music event* (busking, open air concerts), and *traffic* (car, train, tram). Therefore, we compiled a new dataset from the TUT Sound Events (real audio) 2016 development set [17], the Urban Sound Dataset [8], and the IEEE AASP public & private datasets [18], as well as various Youtube videos (particularly for soccer game recordings in the sport event class). Table 1 summarizes the number of files and total duration of files in hours for each class in our dataset.

| Class | Short name | # Files | Total Duration (h) |
|--------------|------------|---------|--------------------|
| Sport event | SE | 34 | 2.37 |
| Roadworks | RW | 35 | 1.29 |
| Public place | PP | 127 | 3.10 |
| Music event | ME | 72 | 3.67 |
| Traffic | TR | 97 | 1.56 |

Table 1: Compiled dataset—number of files and total duration in hours per class.

3.3. Data Augmentation

We apply a two-step data augmentation procedure to enrich our data set. Firstly, each audio file is processed using pitch shifting (± 1 semitone), time stretching (stretch factors of 0.93 and 1.07), and dynamic range compression using the *sox* library [19]. In addition to this “clean” version of the dataset, we created a second “noisy” version of the dataset by mixing each file with environmental background noise using a random signal to noise ratio (SNR) spanning between -14 and -10 dB. This was done in order to simulate the recording conditions in the targeted urban areas. For this purpose, we randomly select segments from five long-term recordings (total length of 135 min) of rain, thunderstorms, and wind (including microphone pop sounds), which were extracted from Youtube videos. Finally, for both datasets, we select 20 random excerpts of three second duration from each file. In total, the datasets each comprise of 43800 time frequency patches.

Figure 1 illustrates 10 randomly selected time-frequency patches for each of the acoustic scene classes. Sport event patches show both transient structures that result e.g. from hand claps as well as harmonic structures that are caused by screaming, speaking, and singing of fans and athletes. Patches from the roadworks class exhibit mostly repetitive structures from machine-like sounds such as drilling or jackhammer. Recordings from the public place class are more sparse with vehicle sounds (e.g. cars, motorbikes) in the background and often harmonic sounds (e.g. people talking, bird singing) in the foreground. The music event class shows clear harmonic structures that result from different musical instruments. Finally, in the traffic class, we observe noise-like structures

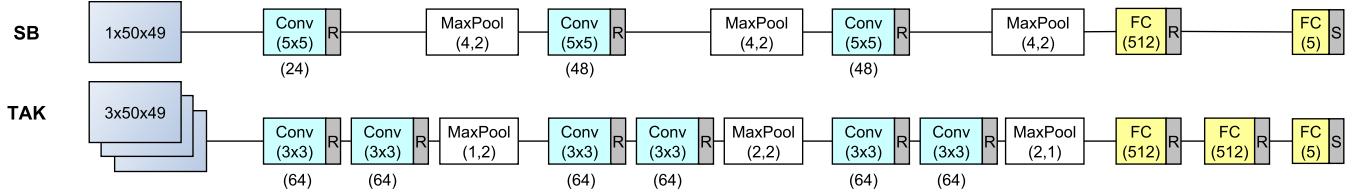


Figure 2: Flowchart of compared CNN architectures **SB** and **TAK**. Number of filters is given below the convolutional layers in brackets.

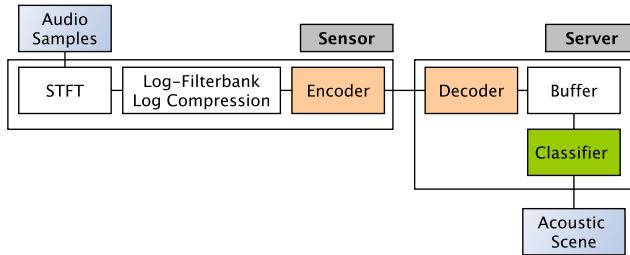


Figure 3: System architecture—distributed sensors and a central server applications.

from moving vehicles and harmonic structures from car honking (constant pitch) and sirens (continuously increasing and decreasing pitch).

3.4. Dimensionality Reduction via Denoising Autoencoders

Due to restrictions of the available mobile communication transmission bandwidth in the targeted area, the data packages from each sensor unit have to be reduced before being transmitted to the server unit. For that purpose, we train a deep neural network (DNN) that performs dimensionality reduction on the time-frequency patches, using a greedy layer-wise training process and the DAE as proposed by Vincent et al in [3]. Each layer is trained to denoise input features which are corrupted by masking noise. The masking noise is drawn from a zero mean and unit variance normal distribution and has a probability of 20 % to corrupt an input neuron. We iteratively train all layers for 30 epochs per layer using the *adadelta* algorithm [20]. The outcome of the above procedure is the trained DNN, denoted as DAE, consisting of an encoder and a decoder part. The encoder and the decoder parts incorporate 4 trained layers each. Through the encoding layers the dimensionality is linearly decreased until the last hidden layer produces the desired reduced dimensionality. On the other hand, the dimensionality in the decoder is increased accordingly such that its output matches the input data dimensionality. The encoder is encapsulated in the acoustic sensors unit, while the decoder is a part of the central server unit.

We compare four scenarios with different ratios of dimensionality reduction. The first scenario does not incorporate the DAE. This means that the non-negative time-frequency patches are transmitted to the server side directly. The second scenario assumes a dimensionality reduction by 25 % using the DAE denoted as $DAE^{0.75}$ in order to encode the time-frequency patches and transmit the encoded representation. Finally, the third and fourth scenarios employ the same idea but using they reduce the dimensionality by 50 % and 75 %, denoted as $DAE^{0.5}$ and $DAE^{0.25}$, respectively. In the future,

we plan to test other image compression techniques such as JPEG or GIF or dictionary learning methods as alternatives for compressing the spectrogram representation.

3.5. Acoustic Scene Classification

As discussed in Section 1, we compare two model architectures **SB** and **TAK**, which are illustrated in Figure 2. Both models consist of multiple convolutional layers combined with maximum pooling layers, which learn suitable feature representations from the input time-frequency patches, and multiple feedforward neural networks for supervised classification. While **SB** has three layers consisting of convolutional filters of size 5×5 , the **TAK** model has three layers of pairs of smaller convolutional filters of size 3×3 . Concerning the max pooling, the **SB** employs larger downsampling over time than over frequency while the **TAK** model first performs pooling over frequency, then over time and frequency, and finally only over time. Another main difference is that while the **SB** model takes spectrogram patches as input, the **TAK** model also takes the first two time derivatives of the spectrogram as additional depth dimensions. In contrast to the original papers, we used a constant number of 64 filters per convolutional layer for the **TAK** model, and used 512 as the dimensionality of the fully connected layers in both models to have comparable parameter values. Apart from that, we adopt the hyper-parameter settings for both models from original papers.

For model training, we use 100 training epochs with early stopping, the *adam* algorithm [21] with a learning rate of 0.001, and a batch size of 200. All experiments were performed using the Keras python package [22]. For training and testing the **TAK** architecture, we concatenated the spectrogram patches in the dataset with their first-order and second-order derivatives as proposed in [5]. The final tensor X_{TAK} that contains the data is of the shape $X_{TAK} \in \mathbb{R}^{43800 \times 3 \times 150 \times 49}$. It is then split into training set (80 %), development set (10 %), and test set (10 %) based on unique source files. For the **SB** architecture, we only use the first depth dimension (magnitude spectrogram) leading to $X_{SB} \in \mathbb{R}^{43800 \times 1 \times 150 \times 49}$.

4. EVALUATION & RESULTS

4.1. Model Comparison

In the evaluation experiment, we compared several configurations of the **TAK** and **SB** models. Firstly, we investigate the influence of the number of filters in the convolutional layers (compare Figure 2). Here, we try the original number of filters (indicated by the fraction $\gamma = 1$), as well as 50 % ($\gamma = 0.5$) and 25 % ($\gamma = 0.25$) of the original number of filters. The corresponding models are indicated as TAK^γ and SB^γ . Secondly, we investigate the models' performance on two datasets—with and without additional environmental noise (compare Section 3.3). Thirdly, we analyze the influence of the

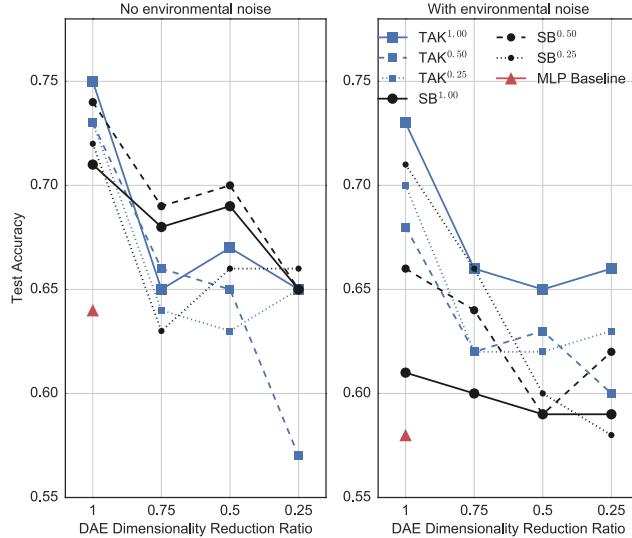


Figure 4: Evaluation results for **SB** and **TAK** models with and without environmental noise, different numbers of filters and different DAE dimensionality reduction ratios.

| | ME | SE | TR | RW | PP |
|---------------------|-------------|-------------|-------------|-------------|-------------|
| Music Event | 0.73 | 0.02 | 0.02 | 0.09 | 0.14 |
| Sport Event | 0.12 | 0.50 | 0.07 | 0.00 | 0.31 |
| Traffic | 0.26 | 0.00 | 0.71 | 0.02 | 0.02 |
| Roadworks | 0.06 | 0.00 | 0.17 | 0.74 | 0.03 |
| Public Place | 0.01 | 0.22 | 0.28 | 0.02 | 0.46 |

Table 2: Confusion matrix for the model $\text{TAK}^{1.00}$ (no DAE dimensionality reduction, with environmental noise).

DAE dimensionality reduction step described in Section 3.4. We test the following dimensionality reduction ratios 1 (implying no dimensionality reduction), 0.75, 0.5, and 0.25, as described in Section 3.4. As a reference, we train and test the DCASE 2017 baseline system, which is based on a multilayer perceptron (MLP), using our given dataset [23]. Here, we do not apply the DAE-based compression. Figure 4 illustrates the test set accuracy values obtained with the different model configurations. For the dataset without environmental noise (left figure) and with additional environmental noise (right figure). Several observations can be made.

All CNN models (both **TAK** and **SB**) clearly outperform the baseline system for the case without DAE dimensionality reduction. If the DAE is used for data dimensionality reduction, we observe lower accuracy values for additional environmental noise, which is somewhat intuitive as the recognition task becomes harder. The **SB** model slightly outperforms the **TAK** model for the “clean” dataset without additional noise. In contrast, in case of additional noise, the **TAK** model with the full number of filters ($\text{TAK}^{1.00}$) shows the best performance throughout all DAE dimensionality reduction ratios. Interestingly, in both noise settings, the **SB** model performs best with half the number of the originally proposed number filters [4] for our dataset (compare $\text{SB}^{0.50}$ vs. $\text{SB}^{1.00}$). In contrast, the **TAK** model shows the best performance for the full number of filters ($\text{TAK}^{1.00}$).

4.2. Class-wise Performance

In order to get further insights into the models’ performance, we show as an example the confusion matrix obtained from the best-performing model $\text{TAK}^{1.00}$ without DAE dimensionality reduction, full number of filters, and with environmental noise in Table 2. It becomes apparent that the classes music event, traffic, and roadworks can be classified with good classification scores above 0.7 while the classes sport event and public place show significantly lower scores. As discussed already in Section 3.3, car honking, which is a prominent sound event in the traffic class, shows similar (horizontal) harmonic structures in the time-frequency patches as music instruments in the music event classes. This is confirmed by a confusion of 0.26 from traffic to music event patches. As both the public place and the sport event class include recordings of people speaking, we observe confusions of 0.22 and 0.31 between public place and sports event and vice versa. A third observation is the high confusion of 0.28 between public place and traffic, which is most likely due to passing vehicles in the background. Finally, the confusion of 0.17 from roadworks to traffic is also interesting, as the confusion from traffic to roadworks is only 0.02. This might be due to the fact that any roadwork scene is much likely to overlap with traffic, but not the other way around.

4.3. Reference Experiment - DCASE 2017 Task 1

We performed an additional baseline classification experiment using the development dataset from the task 1 of the DCASE 2017 challenge (“Acoustic scene classification”), which includes 4680 10 second long excerpts from 15 acoustic scene classes as well as a predefined partition for a 4-fold cross-validation [24]. We randomly sampled 10 one second long time frequency patches from each recording to enlarge the dataset. For the model $\text{SB}^{1.00}$, we obtain mean accuracy values of 0.91 (standard deviation 0.01) for the development set and 0.64 (0.02) for the test set. The $\text{TAK}^{1.00}$ shows slightly higher values of 0.93 (0.001) and 0.67 (0.02) for development and test set, respectively. It must be noted that we do not exploit the full length of the clips e.g. by late fusion techniques like model averaging but instead classify only short excerpts (1 s).

5. CONCLUSIONS

In this paper, we proposed a distributed system for acoustic scene classification in urban environments. Spectrogram patches, which are extracted on the sensor side, are compressed using a deep denoising autoencoder and transmitted to a central server unit, where they are forwarded to a CNN-based classification model. We compared two state-of-the-art network architectures for the task at hand and evaluate their performance depending on additional environmental background noise, the compression rate of the autoencoder, as well as the number of filters in the convolutional layers. Our results show that good classification scores can be achieved despite challenging class partitions with partially shared sound event types.

Acknowledgments

The *StadtLärm* project is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi, ZF 4072003LF6). Stylianos Ioannis Mimalakis is funded by the European Union’s H2020 Framework Programme (H2020-MSCA-ITN-2014) under grant agreement no 642685 MacSeNet.

6. REFERENCES

- [1] “StadtLärm Project Description,” <http://s.fhg.de/StadtLaerm> (last visited: 08/07/2017), 2017.
- [2] “German technical guidelines for noise reduction,” http://www.verwaltungsvorschriften-im-internet.de/bsvwvbund_26081998_IIG19980826.htm (last visited: 08/07/2017), 2017.
- [3] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.
- [4] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, March 2017.
- [5] N. Takahashi, M. Gygli, B. Pfister, and L. V. Gool, “Deep convolutional neural networks and data augmentation for acoustic event detection,” in *Proceedings of the Interspeech Conference*, San Francisco, USA, September 8-12 2016, pp. 2982–2986.
- [6] P. Bellucci, L. Peruzzi, and G. Zambon, “LIFE DYNAMAP project: The case study of Rome,” *Applied Acoustics*, vol. 117, pp. 193–206, 2017.
- [7] D. Hollosi, G. Nagy, R. Rodigast, S. Goetze, and P. Cousin, “Enhancing wireless sensor networks with acoustic sensing technology: Use cases, applications & experiments,” in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 335–342.
- [8] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *Proceedings of the 22nd ACM International Conference on Multimedia (ACM-MM’14)*, Orlando, FL, USA, 2014, pp. 1041–1044.
- [9] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. C. Courville, “Towards end-to-end speech recognition with deep convolutional neural networks,” *CoRR*, vol. abs/1701.02720, 2017. [Online]. Available: <http://arxiv.org/abs/1701.02720>
- [10] S. Sigtia, E. Benetos, and S. Dixon, “An end-to-end neural network for polyphonic piano music transcription,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, May 2016.
- [11] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *Proceedings of the 25th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Sept 2015, pp. 1–6.
- [12] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification and domestic audio tagging,” *DCASE2016 Challenge*, Tech. Rep., September 2016.
- [13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations (ICLR 2015)*, 2015, pp. 1–4.
- [14] S. Adavanne, P. Pertilä, and T. Virtanen, “Sound event detection using spatial features and convolutional recurrent neural network,” in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 771 – 775.
- [15] E. Çakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, “Convolutional recurrent neural networks for bird audio detection,” *CoRR*, 2017.
- [16] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. B. Jackson, and M. D. Plumley, “Unsupervised feature learning based on deep models for environmental audio tagging,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, June 2017.
- [17] G. Lafay, “IEEE DCASE 2016 Challenge - Task 2 - Train/Development Datasets,” https://archive.org/details/dcase2016-task2_train_dev (last visited: 07/04/2017, 2016).
- [18] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumley, “Detection and classification of acoustic scenes and events,” *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct 2015.
- [19] “SoX library,” <http://sox.sourceforge.net/sox.html> (last visited: 08/07/2017), 2017.
- [20] M. D. Zeiler, “ADADELTA: an adaptive learning rate method,” *CoRR*, vol. abs/1212.5701, 2012.
- [21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [22] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [23] “MLP based system, DCASE2017 baseline,” <https://tut-arg.github.io/DCASE2017-baseline-system/system.description.html> (last visited: 08/01/2017, 2017).
- [24] “DCASE 2017 - acoustic scene classification - task description,” <https://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification> (last visited: 17/10/2017), 2017.

SOUND EVENT DETECTION USING WEAKLY LABELED DATASET WITH STACKED CONVOLUTIONAL AND RECURRENT NEURAL NETWORK

Sharath Adavanne, Tuomas Virtanen

Department of Signal Processing , Tampere University of Technology

ABSTRACT

This paper proposes a neural network architecture and training scheme to learn the start and end time of sound events (strong labels) in an audio recording given just the list of sound events existing in the audio without time information (weak labels). We achieve this by using a stacked convolutional and recurrent neural network with two prediction layers in sequence one for the strong followed by the weak label. The network is trained using frame-wise log mel-band energy as the input audio feature, and weak labels provided in the dataset as labels for the weak label prediction layer. Strong labels are generated by replicating the weak labels as many number of times as the frames in the input audio feature, and used for strong label layer during training. We propose to control what the network learns from the weak and strong labels by different weighting for the loss computed in the two prediction layers. The proposed method is evaluated on a publicly available dataset of 155 hours with 17 sound event classes. The method achieves the best error rate of 0.84 for strong labels and F-score of 43.3% for weak labels on the unseen test split.

Index Terms— sound event detection, weak labels, deep neural network, CNN, GRU

1. INTRODUCTION

Sound event detection (SED) is the task of recognizing sound events and its respective start and end timings in an audio recording. Recognizing such sound events and its temporal information can be useful in different applications such as surveillance [1, 2], biodiversity monitoring [3, 4] and query based multimedia retrieval [5]. Traditionally, SED has been tackled with datasets that have temporal information for each of the sound event present [6, 7]. We refer to such temporal information of sound events as strong labels in this paper.

The internet has a vast collection of audio data. Many collaborative and social websites like Freesound¹ and YouTube² allow users to upload multimedia with metadata like captions and tags. We can potentially automate the collection of audio data associated with a given tag from these online sources in considerably less time and manual effort. Recently, Gemekke et al. [8] carried out this with 632 sound event tags on YouTube and collected nearly two

The research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. The authors also wish to acknowledge CSC-IT Center for Science, Finland, for computational resources.

¹<https://freesound.org/>

²<https://www.youtube.com/>

million 10 second audio recordings. While these tags indicate that the sound event is present in the audio recording, the tags do not contain the information as to how many times they occur or at what time they occur. In this paper, we call such tags without any temporal information as weak labels. The task of identifying weak labels of an audio is also referred as audio tagging in literature [9, 10].

Collecting and annotating data with strong labels to train SED methods is a time-consuming task involving a lot of manual labor. On the other hand, collecting weakly labeled data takes much less time to annotate manually, since the annotator has to mark only the active sound event classes and not its exact time boundaries. If we can build SED methods which can learn strong labels from such weakly labeled data, then the methods can learn on a large amount of data. In this paper, we propose to implement such a strong label learning SED method using weakly labeled training data.

Similar research of using weakly labeled data to learn strong labels has been done in neighboring audio domains such as music [11, 12], and bird classification [13, 14]. Liu et al. [11] used a fully convolutional neural network (FCN) to recognize instruments and tempo for each time frame of an audio clip given only the clip level information. They further extended this network to sound event detection [15] and experimented on publicly available datasets. The advantage of using an FCN is it can handle audio input of any length. On the other hand, the limitation is that the frame-wise strong labels are obtained by an upscaling layer which replicates segment-wise output to as many number of frames required. Similar FCN as [15] was proposed in [16] without the upscaling layer, thereby estimating labels for short segments of length 1.5 s instead of frame wise labels. The study compares the performance of this FCN with a VGG-like network [17] like network which outputs sound event labels in segments of 1.5 s. The FCN network is trained using the entire audio, and its respective weak label. On the other hand, the VGG network is trained on sub-segments of the entire audio, assuming that the recording level weak label annotation remains the same in all its sub-segments. The study showed that using an FCN performs better SED than using the VGG method. Kumar et al. [18] proposed a multiple instance learning (MIL) approach [19] for this task, though the results were promising the approach was claimed to be not scalable to large datasets by the same authors in [16].

Sound events in real life most often overlap each other. A SED method which can recognize such overlapping sound events is referred as polyphonic SED method. The state of the art for polyphonic SED, trained using strong labels, was proposed recently in [20], where log mel-band energy feature was used along with a stacked convolutional and recurrent neural network and evaluated on multiple datasets. Similar stacked convolutional and recurrent neural network has also been shown to outperform state of the art

methods in audio tagging tasks [9, 10]. Motivated by the performance of this method in SED and audio tagging, in this paper, we propose to extend the method to perform both SED and audio tagging together, given only the audio and its respective weak labels. In particular, we use the log mel-band audio feature extracted from the audio and extend the stacked convolutional and recurrent neural network to predict two outputs sequentially, the strong followed by the weak labels. To train the proposed network we generate dummy strong labels by replicating the weak labels as many times as the number of frames in the audio input feature. We further propose to control the information that the network learns by separately scaling the loss calculated in the weak and strong prediction layers.

Networks similar to the proposed stacked convolutional and neural network are the current state of the arts for audio tagging [9, 10]. This shows that the architecture is capable of learning the relevant information in temporal domain and mapping it to active classes. In this paper, we show that the proposed training scheme can extract this temporal information that the network is learning in the intermediate layers and can be used as strong labels. In comparison to previous works [15, 16], the proposed method supports higher time resolution for strong labels by its inherent design.

The feature extraction and the proposed network is described in Section 2. The dataset, metric and evaluation procedure is discussed in Section 3. Finally, the results and discussions of the evaluation performed are presented in Section 4.

2. METHOD

Figure 1 shows the overall block diagram of the proposed method. The log mel-band energy feature extracted from the audio is fed to a stacked convolutional and recurrent neural network, which sequentially produces the strong labels followed by the weak labels.

Audio features are calculated using overlapping windows on the input audio of length 10-seconds, resulting in T frames of the feature. The proposed neural network maps these features into strong labels first, and further, the strong labels are mapped to weak labels. For an input of T frames, and a total number of sound classes C in the dataset, the network predicts C for each of the T time frames as strong label output and just C as weak label output. The predicted outputs for each of the sound class is in the continuous range of $[0, 1]$, where one signifies the presence of the sound class and zero the absence. The details of the feature extraction and the network are presented below.

2.1. Feature extraction

Log mel-band energy (*mbe*) is extracted in 40 ms Hamming windows with 50% overlap. In total 40 mel bands are used in the 0–22050 Hz range. For a given 10 second audio input, the feature extraction block produces a 500×40 output ($T = 500$).

2.2. Neural network

The input to the proposed network is the $T \times 40$ *mbe* feature as shown in Figure 1. The local shift-invariant features of this input are learned using CNN layers in the beginning. We use a 3×3 receptive field and pad the output with zeros to keep the size same as input in all our CNN layers. The max-pooling operation is performed along the frequency axis after every layer of CNN to reduce the dimension to $T \times 1 \times N$, where N is equal to the number of filters in the last CNN layer. We do not perform max-pooling along the time axis to preserve the input time resolution. The CNN layers activation

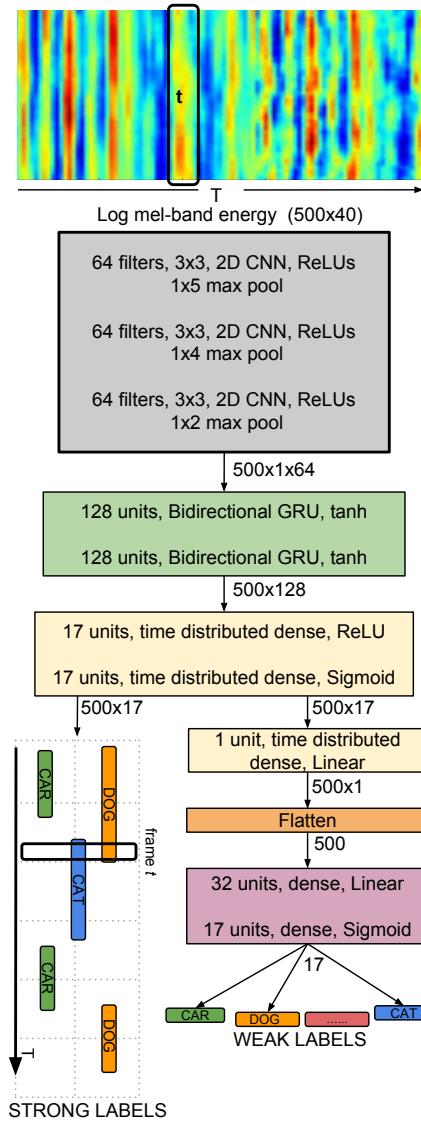


Figure 1: Stacked convolutional and recurrent neural network for learning strong labels from weak labels.

is further fed to a bi-directional gated recurrent units (GRU) with tanh activation to learn the long-term temporal structure of sound events, followed by time distributed fully-connected (dense) layers to reduce the feature-length dimensionality. The time resolution of T frames is unaltered in both the GRU and dense layers. Since we have to predict multiple labels simultaneously, we use sigmoid activation in the last dense layer. This prediction layer outputs the strong labels present in the input audio, and we refer to this as *strong output* in future. The dimensions of the strong labels are $T \times C$. We calculate the strong label loss on this output. Further, we reduce the activation dimensionality and remove the framing information using dense layers and map it to the C weak labels present in the audio. We refer to this weak label prediction layer as *weak output* in future and calculate the weak label loss on its output. The total loss of the network is then calculated as the weighted sum of strong and weak losses.

During the training, the loss at weak and strong outputs was weighed differently to facilitate learning from one output more than the other. In other words, during training, the weak labels along with the weighting scheme help control the learning of strong labels. On the other hand, during testing, the weak labels are obtained from the predicted strong labels.

Batch normalization [21] is performed on the activations of every CNN layer. We train the network for 1000 epochs using binary cross-entropy loss function for both the strong and weak outputs, and Adam [22] optimizer. Early stopping was used to reduce the overfitting of the network to training data. The training was stopped if the sum of the error rate of strong labels and F-score of weak labels (see Section 3.2) referred as the training metric in future did not improve for more than 100 epochs. We used dropout[23] after every layer of the network as a regularizer to make the training generic and work on unseen data. The implementation of the network was done using Keras [24] with Theano [25] as backend.

3. EVALUATION

3.1. Dataset

The method is evaluated using a subset of the recently released Audioset data by Google [8]. This subset was organized as part of a challenge in the Detection and Classification of Acoustic Scenes and Events (DCASE) [26].

The dataset consists of a training, testing and evaluation split. The training split consists of 51,172 recordings, and the testing split consists of 488 recordings. All recordings are of 10-second length, monochannel and sampled at 44100 Hz. All these recordings have been collected from publicly uploaded YouTube videos as explained in [8]. Different methods trained on this training and testing split were benchmarked using the unseen evaluation split of 1103 recordings at the DCASE 2017 challenge [26].

The dataset contains 17 labels in total and each recording can have more than one label. Strong labels are provided only for the testing split, while weak labels are provided for both the splits. In order to train our network, we need strong labels in the training data as well. We generate this by replicating the weak labels for every time frame of the audio and use them as strong labels.

3.2. Metric

We evaluate our method in a similar fashion as the challenge [26]. Evaluation are performed individually on the weak and strong label predictions.

The weak labels are evaluated by calculating the total number of recalls (R), its respective precision (P) and the F-score as $R = TP / (TP + FN)$, $P = TP / (TP + FP)$ and $F = 2 \cdot P \cdot R / (P + R)$ respectively. Where, true positives (TP) is the number of times the method correctly predicted the ground-truth label. False positives (FP) is the number of times the method predicted incorrectly the ground-truth labels. False negative (FN) is the number of times the method did not predict a ground-truth label.

The strong labels are evaluated using a segment based F-score and error rate (ER) as proposed in [27]. According to which the F-score is calculated as

$$F = \frac{2 \cdot \sum_{k=1}^K TP(k)}{2 \cdot \sum_{k=1}^K TP(k) + \sum_{k=1}^K FP(k) + \sum_{k=1}^K FN(k)}, \quad (1)$$

where $TP(k)$, $FP(k)$ and $FN(k)$ are the true positives, false positives and false negatives respectively calculated for each of the K segments. The ER is calculated as

$$ER = \frac{\sum_{k=1}^K S(k) + \sum_{k=1}^K D(k) + \sum_{k=1}^K I(k)}{\sum_{k=1}^K N(k)}, \quad (2)$$

where $N(k)$ is the total number of labels active in a given segment k . $S(k)$, $D(k)$ and $I(k)$ are the substitutions, deletions and insertions respectively measured for each of the K segments as $S(k) = \min(FN(k), FP(k))$, $D(k) = \max(0, FN(k) - FP(k))$ and $I(k) = \max(0, FP(k) - FN(k))$.

We use a segment length of one second for our strong label metrics. The ideal F-score is 100 and ER is zero.

3.3. Baseline

The baseline method for the dataset is provided by [26]. It is a basic method to provide a comparison point for other methods using the dataset. This baseline method uses mbe as the audio feature. The network used is a fully-connected one with two hidden layers, each with 50 units and 20% dropout, followed by a prediction layer with as many sigmoid units as the number of classes in the dataset. A context of five frames of the audio feature is used for training the network along with binary cross-entropy loss and Adam optimizer. The evaluation metric scores for the baseline method are shown in Table 1. The network is trained by replicating the weak labels as many number of times as the number of frames in the input audio feature. During testing, the weak labels are obtained outside the network by identifying the sound events active in the strong labels.

3.4. Evaluation procedure

The stacked convolutional and recurrent neural network is trained with mbe as input, the weak labels provided in the dataset as weak output and the strong labels generated by replicating weak labels for each time frame as strong output.

Given that the data is huge and the hardware has memory constraints, the training time can be long (about 1800 s/epoch on our hardware). We cannot perform an extensive hyperparameter search in the limited time, hence we start with a similar network configuration as in [7], and perform a random search [28] by varying the number of units/filters in each of the layers until no under or over-fitting is observed while having a strong training metric. Since the dataset is large and is uploaded by different users, we assume that there will be enough variability and hence do not use any regularizer. The best configuration with highest training metric is as shown in Figure 1. This configuration has around 218,000 parameters. Other configurations with higher number of parameters, up to 2,000,000, did not show any substantial improvement over the chosen configuration.

On finalizing the network, in order to be sure of our assumption that the high variability in data will not result in an over-fitting model, we experiment using dropout for each layer in our network as a regularizer and vary it in the set of {0.05, 0.15, 0.25, 0.5 and 0.75}. We use the same dropout for all layers in this study.

The weights for the two prediction layers were experimented with different combinations in the logarithmic set of {0.002, 0.02, 0.2, 1}. The number 0.002 is motivated from the ratio of the total number of time frames for the weak label (1) to the number of frames for the strong label (500).

| Dropout | Weak labels | | | Strong labels | |
|---------------|-------------|-------------|-------------|---------------|-------------|
| | P | R | F | ER | F |
| Baseline [26] | 12.2 | 14.1 | 13.1 | 1.02 | 13.8 |
| 0.05 | 44.6 | 37.8 | 40.9 | 0.86 | 38.6 |
| 0.15 | 47.5 | 39.7 | 43.3 | 0.84 | 38.8 |
| 0.25 | 43.0 | 35.0 | 38.6 | 0.86 | 33.8 |
| 0.5 | 21.5 | 17.3 | 19.2 | 0.99 | 8.6 |
| 0.75 | 12.3 | 9.9 | 11.0 | 1.15 | 8.0 |

Table 1: Evaluation metric scores for weak and strong labels for different dropout values.

4. RESULTS AND DISCUSSION

The evaluation metric scores for weak and strong labels for different dropout values tried are compared in Table 1. The best training metric of 43.3% F-score for weak labels and 0.84 ER for strong labels was achieved with 0.15 dropout for the proposed network. This study was performed by having the same weight of one for weak and strong outputs during training. In comparison with the baseline [26] score of 1.02 ER for strong labels and 13.1% F-score for weak labels, this is a significant improvement.

We used the above-estimated dropout of 0.15 and studied how the network learns when provided with different weights for weak and strong outputs and present the results in Table 2. For example, from the first row of the table, the loss at strong output was scaled with 0.002 while the loss at weak output was unscaled. Since the strong labels for training were generated by replicating the weak labels, they are not the actual true labels, hence by intuition, we assumed higher weighting for weak labels will give better training metric. From experimentation, it was seen that the best training metric was actually obtained by using the same weight of one for both the weak and strong outputs. Another interesting observation is that the ER and F-score for strong labels improve when strong output is given more weight than the weak output, even though the strong labels used while training is ‘weak’ in the sense of correctness. On the other hand, this also results in poor metric for weak labels.

We analyzed the predicted labels of our configuration with an equal weight of one for weak and strong outputs which achieved the best training metric. Among the weak labels, the vehicular sound events - train and skateboard, warning events - fire engine siren and civil defense siren were seen to have the highest F-scores of over 60%. On the other hand, sound events - ambulance siren, car alarm, car passing, reverse beeps, train horn had zero F-score. The same sound events and the trend were observed for strong labels.

In order to understand what our method is learning, we visualized the activations in the first convolutional layer of the network for a given output class. The visualizations are done using the saliency map [29] approach implementation in keras-vis [30]. The saliency map is the gradient of output class with respect to the input feature. An example of such visualization is shown in Figure 2 for the recording ‘jc0NAXK8M_30.000_40.000’ in the test dataset. The top and center sub-plots are the activations of the first convolutional layer for the strong and weak output of sound class ‘car’. The bottom subplot shows the ground truth marked in red dotted line over the input *mbe* feature. We see from the activation map of both strong and weak heat maps that the network is actually learning the sound event from a relevant time period in the *mbe* feature.

| Strong Weight | Weak Weight | Weak labels | | | | Strong labels | | | |
|------------------|----------------|-------------|-------------|-------------|-------------|---------------|-------------|-------------|-------------|
| | | P | R | F | F_{ch} | ER | F | ER_{ch} | F_{ch} |
| 0.002 | 1 | 44.9 | 37.0 | 40.5 | | 1.38 | 10.9 | | |
| 0.02 | 1 | 44.2 | 36.5 | 40.0 | | 1.13 | 17.0 | | |
| 0.2 | 1 | 47.5 | 39.6 | 43.2 | 46.6 | 0.84 | 38.1 | 0.80 | 48.3 |
| 1 | 1 | 47.5 | 39.7 | 43.3 | 45.5 | 0.84 | 38.8 | 0.81 | 47.9 |
| 1 | 0.2 | 47.3 | 39.5 | 43.0 | 44.5 | 0.84 | 38.6 | 0.82 | 48.9 |
| 1 | 0.02 | 25.5 | 20.6 | 22.8 | | 0.81 | 41.1 | | |
| 1 | 0.002 | 20.5 | 16.5 | 18.3 | 26.3 | 0.81 | 42.4 | 0.79 | 49.0 |

Table 2: Evaluation metric scores for different combinations of weights for strong and weak label loss. The scores with subscript ch represents the challenge results.

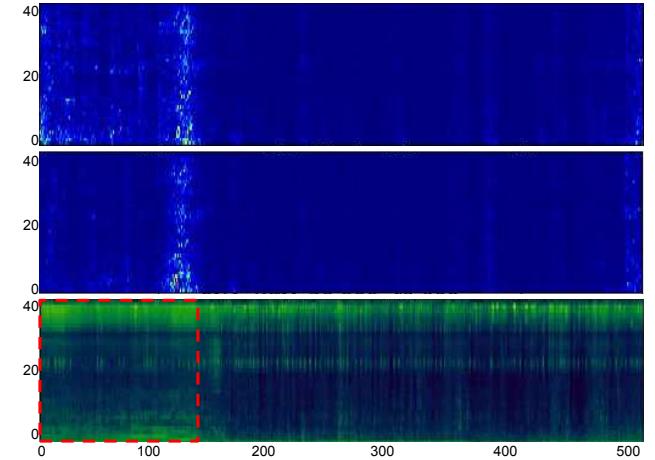


Figure 2: Visualization of the activations in the first layer of CNN for strong (top) and weak (center) prediction of sound event class ‘car’ in ‘jc0NAXK8M_30.000_40.000’ recording. The bottom plot shows the input to the network, the log mel-band feature of the recording, where the sound event class is active in the region bounded by the red dotted line.

4.1. DCASE 2017 challenge results

The results of the proposed method on the evaluation split of DCASE 2017 challenge [26] is presented in Table 2. Four systems with different strong and weak output weighting were chosen based on their performance on test split. Similar trend of better strong label score when strong output is weighed more was observed on evaluation data ($ER_{ch} = 0.79$) as well. In comparison, [31] obtained the best weak label F-score of 55.6% and [32] obtained the best strong label error rate of 0.66.

5. CONCLUSION

The task of learning temporal information of sound events in an audio recording, given only the sound events existing in the audio without the time information is tackled in this paper. A stacked convolutional and recurrent neural network architecture with two prediction layer outputs and a training scheme was proposed in this regard. This network was trained using different weights for the loss calculated in the two prediction layers. Even though the strong labels used for training were just repeated weak labels, it was observed that the network learned the relevant strong labels correctly when the weighting for the two prediction layers was equal. This evaluation was carried out on a publicly available dataset of 155 hours duration. An error rate of 0.84 for strong labels and F-score of 43.3% for weak labels was achieved on the test data.

6. REFERENCES

- [1] A. Harma, M. F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.
- [2] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: A systematic review," in *ACM Computing Surveys (CSUR)*, 2016.
- [3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2015.
- [4] S. Chu, S. Narayanan, and C. J. Kuo, "Environmental sound recognition with time-frequency audio features," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.
- [5] M. Xu, C. Xu, L. Duan, J. S. Jin, and S. Luo, "Audio keywords generation for sports video analysis," in *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2008.
- [6] "Sound event detection in real life audio." Detection and Classification of Acoustic Scenes and Events (DCASE), 2016. [Online]. Available: <http://www.cs.tut.fi/sgn/arg/dcase2016/task-sound-event-detection-in-real-life-audio>
- [7] S. Adavanne, P. Pertilä, and T. Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [8] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: an ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [9] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *International Joint Conference on Neural Networks (IJCNN)*, 2017.
- [10] S. Adavanne, K. Drossos, E. Cakir, and T. Virtanen, "Stacked convolutional and recurrent neural networks for bird audio detection," in *European Signal Processing Conference (EUSIPCO)*, 2017.
- [11] J.-Y. Liu and Y.-H. Yang, "Event localization in music auto-tagging," in *ACM on Multimedia Conference*, 2016.
- [12] M. I. Mandel and D. P. W. Ellis, "Multiple-instance learning for music information retrieval," in *International Conference of Music Information Retrieval (ISMIR)*, 2008.
- [13] J. F. Ruiz-Munoz, M. Orozco-Alzate, and G. Castellanos-Dominguez, "Multiple instance learning-based birdsong classification using unsupervised recording segmentation," in *International Joint Conference on Artificial Intelligence*, 2015.
- [14] F. Briggs, B. Lakshminarayanan, L. Neal, X. Z. Fern, R. Raich, S. J. K. Hadley, A. S. Hadley, and M. G. Betts, "Acoustic classification of multiple simultaneous bird species: a multi-instance multi-label approach," in *The Journal of the Acoustical Society of America*, 2012.
- [15] T.-W. Su, J.-Y. Liu, and Y.-H. Yang, "Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [16] A. Kumar and B. Raj, "Deep CNN framework for audio event recognition using weakly labeled web data," in *arXiv:1707.02530v2*, 2017.
- [17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
- [18] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *ACM on Multimedia Conference*, 2016.
- [19] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Prez, "Solving the multiple instance problem with axis-parallel rectangles," in *International Joint Conference on Artificial Intelligence*, 1997.
- [20] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," in *IEEE/ACM TASLP*, volume 25, issue 6, 2017.
- [21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.
- [22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980 [cs.LG]*, 2014.
- [23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *Journal of Machine Learning Research (JMLR)*, 2014.
- [24] F. Chollet, "Keras v1.1.2," <https://github.com/fchollet/keras>, 2015.
- [25] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," in *arXiv:1605.02688*, 2016.
- [26] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: tasks, datasets and baseline system," in *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE)*, 2017, submitted.
- [27] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," in *Applied Sciences*, vol. 6(6):162, 2016.
- [28] J. Bergstra and Y. Bengio, "Random search for hyperparameter optimization," in *Journal of Machine Learning Research*, 2012.
- [29] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *arXiv:1312.6034*, 2013.
- [30] R. Kotikalapudi, "Keras Visualization Toolkit," 2017. [Online]. Available: <https://github.com/raghakot/keras-vis>
- [31] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumley, "Technical report: Surrey-CVSSP system for DCASE2017 challenge task4," 2017.
- [32] K. Lee, D. Lee, S. Lee, and Y. Han, "Technical report: Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," 2017.

SEQUENCE TO SEQUENCE AUTOENCODERS FOR UNSUPERVISED REPRESENTATION LEARNING FROM AUDIO

Shahin Amriparian^{1,2,3}, Michael Freitag¹, Nicholas Cummins^{1,2}, Björn Schuller^{2,4}

¹ Chair of Complex & Intelligent Systems, Universität Passau, Germany

² Chair of Embedded Intelligence for Health Care & Wellbeing, Augsburg University, Germany

³ Machine Intelligence & Signal Processing Group, Technische Universität München, Germany

⁴ GLAM – Group on Language, Audio & Music, Imperial College London, London, UK

shahin.amriparian@tum.de

ABSTRACT

This paper describes our contribution to the Acoustic Scene Classification task of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2017). We propose a system for this task using a recurrent sequence to sequence autoencoder for unsupervised representation learning from raw audio files. First, we extract mel-spectrograms from the raw audio files. Second, we train a recurrent sequence to sequence autoencoder on these spectrograms, that are considered as time-dependent frequency vectors. Then, we extract, from a fully connected layer between the decoder and encoder units, the learnt representations of spectrograms as the feature vectors for the corresponding audio instances. Finally, we train a multilayer perceptron neural network on these feature vectors to predict the class labels. In comparison to the baseline, the accuracy is increased from 74.8 % to 88.0 % on the development set, and from 61.0 % to 67.5 % on the test set.

Index Terms— deep feature learning, sequence to sequence learning, recurrent autoencoders, audio processing acoustic scene classification

1. INTRODUCTION

Machine learning algorithms for audio processing typically operate on expert-designed feature sets extracted from the raw audio signals. Arguably among the most widely used features are Mel-band energies and features derived from them, such as *Mel Frequency Cepstral Coefficients* (MFCCs). Both feature spaces are widely used in acoustic scene classification [1–3], with the former being employed in the DCASE 2017 Challenge baseline system [4], and the latter being the low level feature space used by the winners of the DCASE 2016 acoustic scene classification challenge [5].

It takes considerable effort and human intervention to manually engineer such features for a specific purpose, which then may not perform well on unrelated tasks. Further, many feature spaces such as MFCCs are non-task specific and are equally adept in a range of audio and speech-based classification tasks [6–8]. For these reasons, among others, unsupervised representation learning has recently gained considerable popularity as a highly effective substitute for using conventional feature sets [9, 10]. Representation learning with deep neural networks (DNNs), in particular, has been shown to be superior to feature engineering for a wide variety of tasks, including speech recognition [6, 11] and music transcription [11, 12]. However, the advantages of deep representation learning are yet to be fully established for acoustic scene classification.

Audio sequences are typically varying length signals; this presents a drawback for deep representation learning using architectures such as convolutional neural networks (CNNs) which typically require inputs of fixed dimensionality. Furthermore, typical DNN architectures used for representation learning, such as stacked autoencoders or Restricted Boltzmann Machines, do not explicitly account for the inherent sequential nature of acoustic data [9]. For the learning of fixed-length representations of variable-length sequential data, sequence to sequence learning with recurrent neural networks (RNNs) has been proposed in machine translation [13, 14]. Moreover, RNNs have been successfully used in a range of audio-based classification tasks such as novelty detection [15], scene classification [16], and speech recognition [17].

In this paper, we extend the RNN encoder-decoder model proposed by Cho et al. [13] to develop a recurrent sequence to sequence autoencoder for deep unsupervised representation learning suitable for use with acoustic data. Sequence to sequence autoencoders have been employed for unsupervised pretraining of RNNs, with promising results on text classification and image recognition tasks [18], as well as machine translation [19]. Variational sequence to sequence autoencoders have been used to learn representations of sentences, and to generate new sentences from the latent space [20]. Furthermore, de-noising recurrent autoencoders have been used for reverberated speech recognition [21], moreover, in this approach, variable-length representations of audio are learnt. Despite this success in a range of applications, to the best of the authors' knowledge, there is no previous work on extracting the learnt representations from sequence to sequence autoencoders for further processing, such as audio classification.

The remainder of this paper is organised as follows: In Section 2, we describe our proposed recurrent sequence to sequence autoencoder approach in detail. Subsequently, we outline our experimental evaluation and results thereof for the DCASE 2017 Acoustic Scene Classification challenge in Section 3. Finally, concluding remarks and our future work plans are given in Section 4.

2. APPROACH

A high-level overview of our system is given in Figure 1. First, mel-spectrograms are extracted from the raw audio files (cf. Figure 1a). Subsequently, a recurrent sequence to sequence autoencoder is trained on these spectra (cf. Figure 1b), which are viewed as time-dependent sequences of frequency vectors. The learnt representations of the spectrograms are then extracted for use as feature

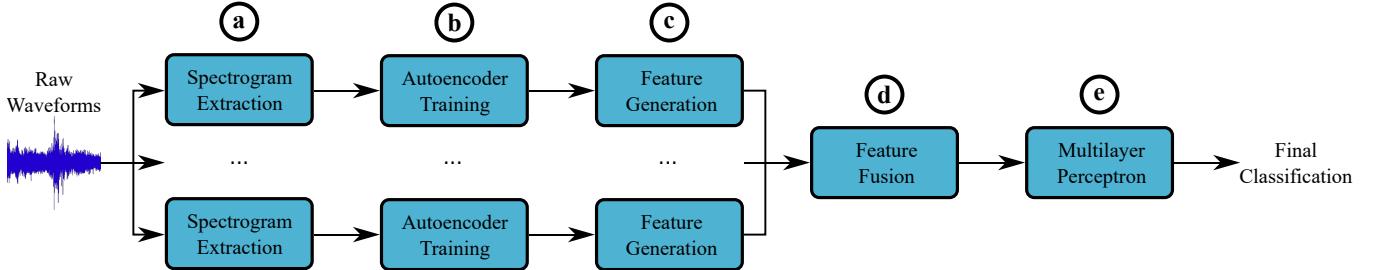


Figure 1: Illustration of the proposed representation learning and classification approach. Except for the final classification, the approach is entirely unsupervised. A detailed account of the procedure is given in Section 2.

vectors for the corresponding instances (cf. Figure 1c). This step is repeated for the different spectral representations made possible by the stereo recordings provided in the challenge dataset, with the resulting set of learnt representation – per audio instance – being concatenated together (cf. Figure 1d). Finally, we train a multilayer perceptron (MLP) (cf. Figure 1e) on the fused feature vectors to predict the labels of instances.

2.1. Spectrogram Extraction

First, the power spectra of audio samples are extracted using periodic Hann windows with width w and overlap $0.5w$. From these, we then compute a given number N_m of log-scaled Mel frequency bands. Finally, the mel-spectra are normalised to have values in $[-1; 1]$, since the outputs of the recurrent sequence to sequence autoencoder are constrained to this interval.

The challenge corpus contains audio samples which have been recorded in stereo [4]. In such data sets, there may be instances in which important information related to the class label has been captured in only one of the two channels. Following the winners of the DCASE 2016 acoustic scene classification challenge [5], we thus extract mel-spectrograms from each individual channel, as well as from the mean and difference of the two channels.

We extract separate sets of mel-spectrograms for different parameter combinations, each containing one mel-spectrogram per audio sample. As illustrated in Figure 1, representations are learnt independently on different sets of mel-spectrograms, and we investigate feature-level fusion of these representations.

2.2. Recurrent Sequence to Sequence Autoencoders

We use recurrent sequence to sequence autoencoders to learn representations of the extracted mel-spectra in an unsupervised manner [13, 14]. An illustration of the structure of these autoencoders is shown in Figure 2. Mel-spectra are viewed as a time-dependant sequence of frequency vectors in $[-1; 1]^{N_m}$, each of which describes the amplitudes of the N_m Mel frequency bands within one audio frame. This sequence is fed to a multilayered *encoder RNN*, which updates its hidden state in each time step based on the input frequency vector. Therefore, the final hidden state of the encoder RNN contains information about the whole input sequence. This final hidden state is transformed using a fully-connected layer, and another multilayered *decoder RNN* is used to reconstruct the original input sequence from the transformed representation.

The encoder RNN consists of N_l layers, each containing N_u Gated Recurrent Units (GRUs) [13]. During our initial system design phase we conducted experiments using Long Short-Term Memory cells instead of GRUs. However, we observed that this

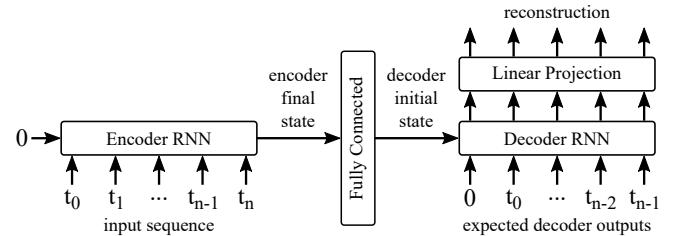


Figure 2: An overview of the implemented recurrent autoencoder.

did not lead to improvements in system performance. The hidden states of the encoder GRUs are initialised to zero for each input sequence, and their final hidden states in each layer are concatenated into a one-dimensional vector. This vector can be viewed as a fixed-length representation of a variable-length input sequence, with dimensionality $N_l \cdot N_u$, if the encoder RNN is unidirectional, and dimensionality $2 \cdot N_l \cdot N_u$ if it is bidirectional.

The representation vector is then passed through a fully connected layer with hyperbolic tangent activation. The output dimensionality of this layer is chosen in such a way that it can be used to initialise the hidden states of the decoder RNN.

The decoder RNN contains the same number of layers and units as the encoder RNN. Its task is the frame-by-frame reconstruction of the input mel-spectrogram, based on the representation which was used to initialise the hidden states of the decoder RNN. At the first time step, a zero input is fed to the decoder RNN. During subsequent time steps t , the expected decoder output at time $t - 1$ is fed as input to the decoder RNN [14]. Stronger representations could potentially be obtained by using the actual decoder output instead of the expected output, since this reduces the amount of information available to the decoder. However, during initial experiments we observed that our approach greatly accelerates model convergence with negligible effects on representation quality.

The outputs of the decoder RNN are passed through a single linear projection layer with hyperbolic tangent activation at each time step in order to map the decoder RNN output dimensionality to the target dimensionality N_m . The weights of this output projection are shared across time steps. In order to introduce greater short-term dependencies between the encoder and the decoder, our decoder RNN reconstructs the reversed input sequence [14, 22].

Autoencoder training is performed using the root mean square error (RMSE) between the decoder output and the target sequence as the objective function. Dropout is applied to the inputs and outputs of the recurrent layers, but not to the hidden states. Once training is complete, the activations of the fully connected layer are extracted as the learnt representations of spectrograms.

2.3. Multilayer Perceptron Classifier

A multilayer perceptron, similar to the one used in the challenge baseline system [4], is employed for classification. Our MLP contains two hidden fully-connected layers with rectified linear activation, and a softmax output layer. The hidden layers contain 150 units each, and the output layer contains one unit for each class label. Training is performed using cross entropy between the ground truth and the network output as the objective function, with dropout applied to all layers except the output layer. A range of different classifiers were tested during our initial experimentation. However, we observed that more sophisticated classification paradigms did not aid our overall system performance.

3. EXPERIMENTAL SETTINGS AND RESULTS

3.1. Database

The DCASE 2017 acoustic scene classification challenge is carried out on the TUT Acoustic Scenes 2017 data set [4]. This data set contains binaural audio samples of 15 acoustic scenes recorded at distinct geographic locations. For each location, between 3 and 5 minutes of audio were initially recorded and then split into 10 second segments. The development set for the challenge contains 4 680 instances, with 312 instances per class, and the evaluation set contains 1 620 instances with unknown labels.

A four-fold cross-validation setup is provided by the challenge organisers for the development set. In each fold, roughly 75 % of the samples are used as the training split, and the remaining samples are used as the evaluation split. Samples from the same original recording are always included in the same split. For further detail on the challenge data and the cross fold validation set-up, the interested reader is referred to [4].

3.2. Common Experimental Settings

We have implemented the representation learning approach outlined above as part of the AUDEEP toolkit¹ for deep representation learning from audio. AUDEEP is implemented in Python, and relies on TENSORFLOW² for the core sequence to sequence autoencoder and MLP implementations.

Both the autoencoders and MLPs are trained using the Adam optimiser with a fixed learning rate of 0.001 [23]. Autoencoders are trained for 50 epochs in batches of 64 samples, and we apply 20 % dropout to the outputs of each recurrent layer. Furthermore, we clip gradients with absolute value above 2 [14]. The MLPs used for classification are trained for 400 epochs without batching or gradient clipping, and 40 % dropout is applied to the hidden layers. Features are standardised to have zero mean and unit variance during MLP training, and the corresponding coefficients are used to transform the validation data.

3.3. Hyperparameter Optimisation

Our proposed approach contains a large number of adjustable hyperparameters, which prohibits an exhaustive exploration of the parameter space. Instead, we select suitable values for the hyperparameters in stages, using the results of our preliminary experiments to bootstrap the process. During these experiments, we observed

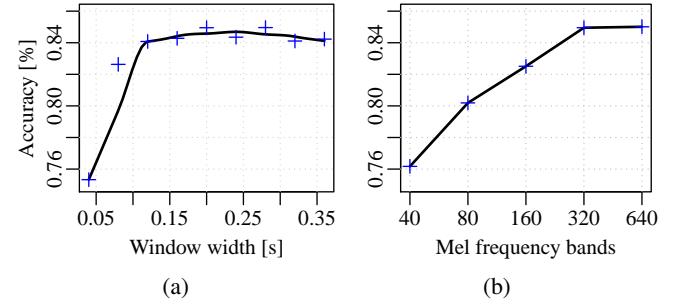


Figure 3: Classification accuracy on the development set for different FFT window widths (a), and different numbers of Mel frequency bands (b). A detailed account of the experiments leading to these results is given in Section 3.3.

that very similar parameter choices lead to comparable performance on spectrograms extracted from different combinations of the audio channels (mean, difference, left and right). We therefore performed hyperparameter optimisation on the mean-spectrograms only, and used the resulting parameters for the other spectrogram types.

In the first stage, we selected a suitable autoencoder configuration, i. e. the optimal number of recurrent layers N_l , the number of units per layer N_u , and either unidirectional or bidirectional encoder and decoder RNNs. In this stage, autoencoders are trained on mel-spectrograms extracted with window width $w = 0.16$ seconds, window overlap $0.5w = 0.08$ seconds, and $N_m = 320$ Mel frequency bands, without amplitude clipping. These choices proved to be reasonable during our preliminary evaluation. We exhaustively evaluated $N_l \in \{1, 2, 3\}$, $N_u \in \{16, 32, 64, 128, 256, 512\}$ and all combinations of uni- or bidirectional encoder and decoder RNNs. The highest classification accuracy was achieved when using $N_l = 2$ layers with $N_u = 256$ units, a unidirectional encoder RNN, and a bidirectional decoder RNN.

Our second development stage served to optimise the window width w used for spectrogram extraction. We use the autoencoder configuration determined in the previous stage, and once again set $N_m = 320$. The window width w is evaluated between 0.04 and 0.36 seconds in steps of 0.04 seconds. For each value of w , the window overlap is chosen to be $0.5w$. As shown in Figure 3a, classification accuracy quickly rises above 84 % for $w > 0.10$ seconds, and peaks at 85.0 % for $w = 0.20$ seconds and $w = 0.28$ seconds. For larger values of w , classification accuracy decreases again. As a larger window width may blur some of the short-term dynamics of the audio signals, we choose $w = 0.20$ seconds. Correspondingly, the window overlap is chosen to be $0.5w = 0.10$ seconds.

In the final optimisation stage, we evaluated different numbers of Mel frequency bands $N_m \in \{40, 80, 160, 320, 640\}$, the results of which are shown in Figure 3b. Classification accuracy rises with larger values of N_m until it reaches 85.0 % for $N_m = 320$. Increasing N_m beyond 320 does not improve performance further, so we choose $N_m = 320$ to minimise the amount of data the system has to process.

3.4. Fusion Experiments

Given the supplied stereo audio tracks [4], we extract separate sets of spectrograms from the mean and difference of channels, and from the left and right channels individually (cf. Section 2.1). On each set of spectrograms, an autoencoder is trained, and the learnt

¹<https://github.com/auDeep/auDeep>

²<http://www.tensorflow.org>

Table 1: Comparison of the classification accuracies of the different variants of our proposed system with the challenge baseline. We extract four different feature sets of spectrograms from the mean (M) and difference (D) of channels, and from the left (L) and right (R) channels separately. We obtain the highest accuracy after fusing the features generated from all channels.

| System | Features | Accuracy [%] | |
|--|-----------------|--------------|-------|
| | | Devel. | Eval. |
| Baseline | 200 (per frame) | 74.8 | 61.0 |
| Proposed: Individual Feature Sets | | | |
| Mean (M) | 1 024 | 85.0 | — |
| Left (L) | 1 024 | 84.6 | — |
| Right (R) | 1 024 | 83.8 | — |
| Difference (D) | 1 024 | 82.0 | — |
| Proposed: Fused Feature Sets | | | |
| Mean, Left | 2 048 | 86.2 | — |
| Mean, Left, Right | 3 072 | 86.9 | — |
| All (M + L + R + D) | 4 096 | 88.0 | 67.5 |

representations are extracted as features for the instances. This results in four feature sets herein identified by the spectrogram type from which they have been extracted (i. e. ‘mean’, ‘difference’, ‘left’, and ‘right’). On the development set, the ‘mean’ feature set achieves the highest classification accuracy with 85.0 %, followed by ‘left’ with 84.6 %, ‘right’ with 83.8 %, and ‘difference’ with 82.0 % (cf. Table 1). In order to determine if the different spectral representations contain complementary information, we perform feature-level fusion. We perform a weighted fusion, in which the weights are proportional to performance of the individual systems. Fusing the ‘mean’ and ‘left’ feature sets improves classification accuracy to 86.2 % on the development set. Adding the ‘right’ feature set further increases classification accuracy to 86.9 %, and fusing all four feature sets results in 88.0 % accuracy (cf. Table 1). The latter constitutes our best result on the development set, with an improvement of 13.2 % over the baseline [4]. A confusion matrix for this result is given in Figure 4.

Besides fusion between different channels, we also investigated fusion between different window sizes w and numbers of Mel frequency bands N_m . We also trialled fusion with various conventional acoustic feature sets which we extracted from the raw audio samples using the openSMILE toolkit [24]. However, we did not identify a combination of these options which resulted in increased performance on the development set.

3.5. Challenge Submission and Evaluation Set Results

For our submission to the DCASE 2017 Acoustic Scene Classification Challenge, we select the four feature sets with the best performance on the development partition, i. e. the ‘mean’ feature set and all three fused feature sets. We extract spectrograms from audio samples in the evaluation set using the same parameters that we used for the development set. Subsequently, we extract the four individual feature sets described above with the respective autoencoder that we trained on the development set. Finally, fusion of these feature sets is performed as detailed above.

For prediction on the evaluation set, the MLP classifier is trained using the entire development set as training data. As shown

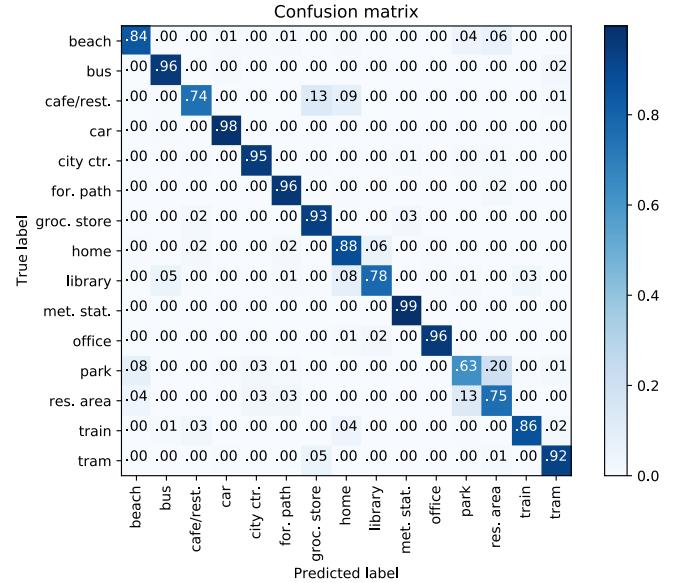


Figure 4: Confusion Matrix of our strongest performing system on the development partition of the TUT Acoustic Scenes 2017 data set which achieved a classification accuracy of 88.0 %.

in Table 1, our approach achieved classification accuracies of 00.0 %, 00.0 %, 00.0 % and 00.0 % on the evaluation set.

4. CONCLUSIONS

Despite representation learning with deep neural networks (DNNs) has shown superior performance of hand-crafted feature sets in a variety of machine learning recognition and classification tasks, such approaches have not been widely explored with the domain of acoustic scene classification. In this regard, our entry to the 2017 DCASE 2017 Acoustic Scene Classification challenge has demonstrated the feasibility of using a recurrent sequence to sequence autoencoder for the unsupervised feature representation. A major advantage of our approach is that it is able to learn a fixed length representation from variable length audio signals while taking account of their time-dependent nature. A fused combination of features learnt from our system was able to achieve an accuracy of 88.0 % on the challenge development data, an improvement of 17.65 percentage points over the official baseline.

In future work, we will be testing our system over a wide range of different acoustic classification tasks. We also want to collect further data from social multimedia using our purpose built software [25] to train the autoencoder with more real life audio recordings. Finally we plan to investigate the potential of Generative Adversarial Networks for acoustic based deep representation learning.

5. ACKNOWLEDGEMENTS

This research has received funding from the European Union’s Seventh Framework under grant agreement No. 338164 (ERC StG iHEARu) and the Innovative Medicines Initiative 2 Joint Undertaking under grant agreement No 115902. This Joint Undertaking receives support from the European Union’s Horizon 2020 research and innovation programme and EFPIA.



6. REFERENCES

- [1] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *24th European Signal Processing Conference (EUSIPCO 2016)*. Budapest, Hungary: IEEE, Aug 2016, pp. 1128–1132.
- [2] 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 *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 IEEE AASP Challenge Workshop (DCASE 2016), satellite to EUSIPCO 2016*. Budapest, Hungary: IEEE, Sep 2016, pp. 65–69.
- [3] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. Plumbley, “Detection and Classification of Acoustic Scenes and Events,” *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.
- [4] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, Nov 2017, submitted.
- [5] 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,” *Detection and Classification of Acoustic Scenes and Events 2016 IEEE AASP Challenge (DCASE 2016)*, Sep 2016, Technical Report.
- [6] G. Hinton, L. Deng, D. Yu, et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [7] B. Schuller and A. Batliner, *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing*. Chichester, United Kingdom: Wiley.
- [8] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.
- [9] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [10] M. Schmitt and B. Schuller, “openXBOW — Introducing the Passau open-source crossmodal bag-of-words toolkit,” *Journal of Machine Learning Research*, vol. 18, 2017, 5 pages.
- [11] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1096–1104.
- [12] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” in *Proceedings of the 29th International Conference on Machine Learning (ICML’12)*. Edinburgh, Scotland: Omnipress, June 2012, pp. 1881–1888.
- [13] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: ACL, Oct 2014, pp. 1724–1734.
- [14] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112.
- [15] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, “Deep Recurrent Neural Network-based Autoencoders for Acoustic Novelty Detection,” *Computational Intelligence and Neuroscience*, vol. 2017, 2017, 14 pages.
- [16] M. Zöhrer and F. Pernkopf, “Gated Recurrent Networks applied to Acoustic Scene Classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 IEEE AASP Challenge Workshop (DCASE 2016), satellite to EUSIPCO 2016*. Budapest, Hungary: IEEE, Sep 2016, pp. 115–119.
- [17] F. Weninger, J. Bergmann, and B. Schuller, “Introducing CURRENNT: the Munich Open-Source CUDA RecurREnt Neural Network Toolkit,” *Journal of Machine Learning Research*, vol. 16, pp. 547–551, 2015.
- [18] A. M. Dai and Q. V. Le, “Semi-supervised sequence learning,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3079–3087.
- [19] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, “Multi-task sequence to sequence learning,” *arXiv preprint arXiv:1511.06114*, 2015, 10 pages.
- [20] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” *arXiv preprint arXiv:1511.06349*, 2015, 12 pages.
- [21] F. Weninger, S. Watanabe, Y. Tachioka, and B. Schuller, “Deep recurrent de-noising auto-encoder and blind de-reverberation for reverberated speech recognition,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, May 2014, pp. 4623–4627.
- [22] Y.-A. Chung, C.-C. Wu, C.-H. Shen, and H.-Y. Lee, “Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks,” in *INTERSPEECH*. ISCA, 2016, pp. 765–769.
- [23] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014, 15 pages.
- [24] F. Eyben, F. Weninger, F. Gross, and B. Schuller, “Recent developments in opensmile, the munich open-source multimedia feature extractor,” in *Proceedings of the 21st ACM International Conference on Multimedia*. ACM, Nov 2013, pp. 835–838.
- [25] S. Amiriparian, S. Pugachevskiy, N. Cummins, S. Hantke, J. Pohjalainen, G. Keren, and B. Schuller, “CAST a database: Rapid targeted large-scale big data acquisition via small-world modelling of social media platforms,” in *Proc. ACII 2017*. San Antonio, TX: IEEE, October 2017, 6 pages.

NONNEGATIVE FEATURE LEARNING METHODS FOR ACOUSTIC SCENE CLASSIFICATION

Victor Bisot[†], Romain Serizel^{§†}, Slim Essid[#], Gaël Richard[#]*

[‡]LTCI, Télécom ParisTech, Université Paris Saclay, F-75013, Paris, France

^{*} Université de Lorraine, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54506, France

[§]Inria, Villers-lès-Nancy, F-54600, France

[†]CNRS, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54506, France

ABSTRACT

This paper introduces improvements to nonnegative feature learning-based methods for acoustic scene classification. We start by introducing modifications to the task-driven nonnegative matrix factorization algorithm. The proposed adapted scaling algorithm improves the generalization capability of task-driven nonnegative matrix factorization for the task. We then propose to exploit simple deep neural network architecture to classify both low level time-frequency representations and unsupervised nonnegative matrix factorization activation features independently. Moreover, we also propose a deep neural network architecture that exploits jointly unsupervised nonnegative matrix factorization activation features and low-level time frequency representations as inputs. Finally, we present a fusion of proposed systems in order to further improve performance. The resulting systems are our submission for the task 1 of the DCASE 2017 challenge.

Index Terms— Feature learning, Nonnegative Matrix Factorization, Deep Neural Networks,

1. INTRODUCTION

In this paper we deal with the acoustic scene classification (ASC) problem [1], a subtask of the more general computational auditory scene analysis area of research. The goal of ASC is to identify in which type of environment a recording has been captured. ASC is a particularly interesting machine learning application, as it has been shown that computational methods tend to outperform humans when asked to discriminate between sound scenes. Moreover, ASC was the subtask of the 2016 edition of the DCASE challenge [2] that attracted the most submissions, showing its rising increase in popularity.

Most earlier ASC works relied on simple classifiers such as support vector machines (SVM) to classify hand-crafted features inspired from other audio classification tasks. Notable feature-based systems include the use of Mel frequency or Gammatone filter bank-based cepstral coefficients [3, 4] sometimes paired with recurrent quantitative analysis to model the temporal evolution of the scene [5]. Features inspired from image processing such as histograms of oriented gradients [6] or local binary patterns [7] have been proposed as well to describe the texture of time-frequency representations of the scenes.

Nowadays, the trend is shifting towards feature learning or deep learning-based techniques. First, unsupervised feature learning techniques such as K-means or nonnegative matrix factorization (NMF) have shown to be competitive with best hand-crafted

features [8, 9]. Supervised extensions of NMF have also been proposed to adapt the decomposition to the task at hand in order to learn better features [10, 11]. The second dominant trend in ASC is to find appropriate neural network structures for the task. One way to address ASC with deep learning is to use deep neural networks (DNN) as a better classifiers than SVMs to interpret large sets of hand-crafted features [12]. Many works have also proposed more complex neural networks such as convolution neural networks (CNN) [10, 13, 14] or recurrent neural networks (RNN) [15].

In this paper, while describing our submission for the 2017 DCASE challenge [16], we present two contributions by proposing improvements to previous successful ASC techniques. Our main contribution is an efficient modification to previous task-driven nonnegative matrix factorization (TNMF) algorithm for ASC [8]. TNMF is a supervised variant of NMF which jointly learns a non-negative dictionary and a classifier that has obtained very good results for the task [11]. The modified algorithm introduces a way to take the scaling of NMF projections into account in the TNMF framework. The proposed adaptive scaling strategy allows for balancing the contribution of each dictionary component when jointly learning the dictionary and classifier. The system that we propose also includes DNN trained by using unsupervised NMF features as an input. While the most common approach is to train networks on low-level time-frequency representations it has been shown that NMF features can often be a better choice of representation to train DNNs [17]. In this work, we intend to take advantage of both representations by proposing a DNN architecture where two branches are trained in parallel on the NMF features and time-frequency representations separately before being merged by concatenation, deeper in the network. Finally, we propose a simple fusion of both our TNMF and DNN systems trained on both channels of the scene stereo recordings in order to further improve the performance of our systems.

The paper is organized as follows. The modified TNMF algorithm is described in Section 2. The chosen DNN architectures are introduced in Section 3. Experimental results and our final system are presented in Section 4. Finally, conclusions and directions for future work are exposed in Section 5.

2. IMPROVEMENTS TO SUPERVISED NMF

2.1. NMF and TNMF models

The ASC systems we propose in this paper all rely on variants of NMF [18] to learn features from time-frequency representations. Suppose we have a nonnegative data matrix $\mathbf{V} \in \mathbb{R}_+^{F \times N}$ such as the

time-frequency representation of an audio recording, where F is the number of frequency bands and N is the number of time frames. The goal of NMF [18] is to find a decomposition that approximates the data matrix \mathbf{V} such as: $\mathbf{V} \approx \mathbf{WH}$, with $\mathbf{W} \in \mathbb{R}_+^{F \times K}$ and $\mathbf{H} \in \mathbb{R}_+^{K \times N}$. NMF is obtained by solving the following optimization problem:

$$\min D_\beta(\mathbf{V}|\mathbf{WH}) \text{ s.t. } \mathbf{W}, \mathbf{H} \geq 0; \quad (1)$$

where D_β represents the β -divergence (Euclidean distance with $\beta = 2$, generalized Kullback-Leibler divergence with $\beta = 1$).

In order to adapt our NMF decompositions to the task at hand, we focus on TNMF [8], a supervised variant of NMF. TNMF approaches are nonnegative instantiations of the more general task-driven dictionary learning framework [19] that were successfully applied to speech enhancement and acoustic scene classification [8, 20]. The motivation behind using TNMF is to learn discriminative nonnegative dictionaries of spectral templates with the objective of minimizing a classification cost. In our case, the TNMF model jointly learns a multinomial logistic regression and a nonnegative dictionary.

Let each data frame \mathbf{v} be associated with a label y in a fixed set of labels \mathcal{Y} . The TNMF problem is expressed as a nested optimization problem as follows:

$$\begin{cases} \mathbf{h}^*(\mathbf{v}, \mathbf{W}) = \min_{\mathbf{h} \in \mathbb{R}_+^K} D_\beta(\mathbf{v}|\mathbf{Wh}) + \lambda_1 \|\mathbf{h}\|_1 + \frac{\lambda_2}{2} \|\mathbf{h}\|_2^2 \\ \min_{\mathbf{W} \in \mathcal{W}, \mathbf{A}} \mathbb{E}_{y, \mathbf{v}} [\ell_s(y, \mathbf{A}, \mathbf{h}^*(\mathbf{v}, \mathbf{W}))] + \frac{\nu}{2} \|\mathbf{A}\|_2^2. \end{cases} \quad (2)$$

Thus, the features for classification are computed as the output of a function $\mathbf{h}^*(\mathbf{v}, \mathbf{W})$ of the data point \mathbf{v} and the dictionary \mathbf{W} , solution of a nonnegative sparse coding problem with ℓ_1 and ℓ_2 -norm penalties. The objective is to minimize the expectation of a classification loss $\ell_s(y, \mathbf{A}, \mathbf{h}^*(\mathbf{v}, \mathbf{W}))$, a function of the optimal projection $\mathbf{h}^*(\mathbf{v}, \mathbf{W})$, where \mathbf{A} are the parameters of the classifier. Therefore, the TNMF problem is a joint minimization of the expected classification cost over \mathbf{W} and \mathbf{A} . Here, \mathcal{W} is defined as the set of nonnegative dictionaries containing unit ℓ_2 -norm basis vectors and ν is a regularization parameter on the classifier parameters to prevent over-fitting.

In deep learning terminology the TNMF model can be seen as a one hidden layer network, where \mathbf{W} are the parameters of the layer and the output of the layer is computed by applying the function $\mathbf{h}^*(\mathbf{v}, \mathbf{W})$ to the input \mathbf{v} and parameters \mathbf{W} . Then, the classification layer is a fully connected layer with softmax activations acting as multinomial regression. The parameters of both layers (\mathbf{W} and \mathbf{A}) are jointly trained to minimize a categorical cross-entropy loss.

2.2. Adapted scaling algorithm for TNMF

As with any other features in general, when performing NMF for feature learning it is often advised to scale the resulting activation matrix \mathbf{H}^* before classification in order for each feature dimension (K in that case) to have unit variance. Especially in ASC, some components of the dictionary might represent lower energy or distant background events that could be useful to discriminate between different environments. By scaling the activation matrix, each basis event will have a more balanced contribution during the classifier training phase. This problem has also been addressed for deep neural networks with batch normalization [21]. Similarly to training DNN, during TNMF training, changes in the dictionary result in changes in the distribution of the activations which can make it more challenging to train the classifier.

We introduce a simple but efficient variant of our previous TNMF algorithm [11] which takes the activation matrix scaling operation into account for both the classifier and dictionary updates. Let N be the number of examples in the training data, we divide the data randomly into B different mini-batch. The mini-batch are subsets of the data of equal size where \mathbf{V}_b is the mini-batch of index b . We obtain the mini-batch of activations \mathbf{H}_b by applying the function \mathbf{h}^* to each data point in \mathbf{V}_b . We also define m_b and σ_b^2 the mean and variance of the mini-batch projection matrix \mathbf{H}_b . In the proposed algorithm, we first compute the mean m and variance σ over the optimal projection matrix \mathbf{H}^* of the training data on the dictionary \mathbf{W} . These statistics are used to scale projections to have zero mean and unit variance before updating the classifier parameters.

The next step is to update the dictionary with mini-batch stochastic gradient descent as in [8]. The statistics of the activation matrix indirectly depend on the dictionary so they would change after the dictionary update on each mini-batch. To take these changes into account we slightly update the global statistics in the direction of the statistics of the mini batch's projection. We modify the mean in the direction of the batch mean proportionally to the size of the mini-batch $m = m - \frac{1}{B}(m - m_b)$ and repeat the process for updating the global variance. We then use the updated statistics to scale the activation features to zero mean and unit variance. The update of the statistics adds an additional operation between the projections and the classifier which needs to be taken into account when computing the gradient of the loss $\nabla_{\mathbf{W}} l_s(y, \mathbf{A}, \mathbf{H}_b)$ with respect to \mathbf{W} . Here, we make the approximation that the statistics are constant which makes the modifications to the expressions of the gradients [19] straightforward. In practice, this approximation seems justified as it does not prevent the loss from decreasing and our adapted scaling algorithm shows improved performance over the previous TNMF algorithms for ASC. The proposed modifications are presented in Algorithm 1, we will refer to TNMF trained with the adaptive scaling algorithm as TNMF-AS. Here, the operation $\Pi_{\mathcal{W}}$ used in the update step for \mathbf{W} corresponds to the projection on the set \mathcal{W} . In practice, this projection is done by thresholding the coefficients to 0 and normalizing each dictionary component to have unit ℓ_2 norm.

Algorithm 1 Adaptive scaling TNMF algorithm

Require: $\mathbf{V}, \mathbf{W} \in \mathcal{W}, \mathbf{A} \in \mathcal{A}, \lambda_1, \lambda_2, \nu, I, \rho$

for $i = 1$ to I **do**

//Update classifier

Compute $\mathbf{H}^* = \mathbf{h}^*(\mathbf{V}, \mathbf{W})$

Compute mean m and variance σ of \mathbf{H}^*

$\mathbf{H}' = \frac{1}{\sigma}(\mathbf{H}^* - m)$

Update classifier parameters \mathbf{A} with LBFGS

//Update dictionary

for $b = 1$ to B **do**

Draw a batch \mathbf{V}_b and its labels y_b

Compute $\mathbf{H}_b^* = \mathbf{h}^*(\mathbf{V}_b, \mathbf{W})$

Compute mean m_b and variance σ_b^2 of \mathbf{H}_b^*

$m = m - \frac{1}{B}(m - m_b)$ and $\sigma^2 = \sigma^2 - \frac{1}{B}(\sigma^2 - \sigma_b^2)$

$\mathbf{H}'_b = \frac{1}{\sigma}(\mathbf{H}_b - m)$

$\mathbf{W} \leftarrow \Pi_{\mathcal{W}}[\mathbf{W} - \rho \nabla_{\mathbf{W}} l_s(y, \mathbf{A}, \mathbf{H}'_b)]$

end for

end for

return \mathbf{W}, \mathbf{A}

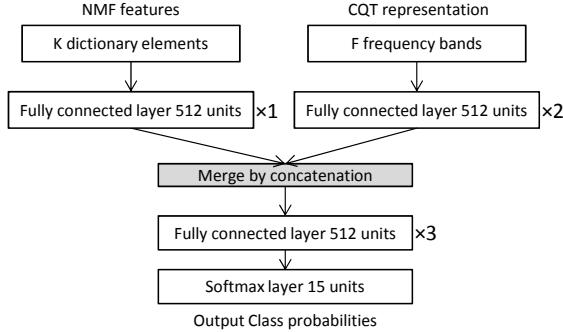


Figure 1: Architecture of the DNN-M model learned on both representations. All hidden layers have ReLU activations and a dropout of 0.2.

3. DNN MODELS

3.1. Learning from different representations

The recent rise in popularity of deep learning approaches for ASC allows for insightful comparison of the effectiveness of different network configurations, especially after the 2016 edition of the DCASE challenge [2]. The most successful neural network-based approaches in ASC usually involve training simple DNN on large sets of hand-crafted features [12] or CNN directly on time-frequency representations [13, 15]. Instead, in our recent work [17], we have shown the usefulness of training simple DNN on features learned from NMF decompositions. In fact, the unsupervised NMF decomposition plays a better role at extracting suitable mid-level representations from low-level time-frequency representations than the first layer of DNN. While training DNN on NMF-based features outperformed those trained on time-frequency representations, we believe they can be complementary as the two approaches do not have the same properties. Therefore we propose a simple architecture where the network is trained jointly in the NMF features and the time-frequency representations. The architecture of the model is shown in Figure 1. The network is first composed of two branches that are later merged by concatenation before getting into the final layers of the network. The first branch has the activation matrix \mathbf{H} as input while the time-frequency data matrix \mathbf{V} is the input of the second branch. The first branch has less hidden layers, as the unsupervised NMF plays the role of a pre-trained hidden layer [17]. The proposed DNN model using both representations as inputs will be denoted as DNN-M in the remainder of the paper. For DNN-M, as well as the other alternative DNN that are compared to ours, all layers are simple fully-connected layers with rectified linear unit activations and dropout between each layer.

3.2. Fusion of systems

Many of the better performing ASC methods include some form a fusion between different systems [11, 13, 14]. In a similar way, we propose a simple fusion of TNMF-AS introduced in Section 2.2 with the DNN-M architecture. Moreover, combining different occurrences of the different models can help mitigating the uncertainty inherent to the training of those models, both owing to the data samples used and the non-uniqueness of the solutions obtained for the non-convex problems solved during the training. Both the TNMF and the DNN models rely on multinomial logistic regres-

| | TNMF methods | | |
|-------------------------|--------------|-------------|-------------|
| | $K = 256$ | $K = 512$ | $K = 1024$ |
| NMF (unsupervised) | 82.3 | 84.3 | 84.0 |
| TNMF [8] | 86.6 | 86.8 | 86.3 |
| TNMF-AS | 87.6 | 88.2 | 87.7 |
| DNN methods | | | |
| Baseline system [16] | | 77.2 | |
| DNN-CQT | | 85.5 | |
| DNN-NMF | 86.5 | 87.0 | 87.2 |
| DNN-M | 87.3 | 87.8 | 87.9 |
| Fusion of final systems | | | |
| TNMF-AS stereo | 87.5 | 88.4 | - |
| DNN-M stereo | 88.0 | 88.8 | 89.2 |
| TNMF-AS fusion | | 89.2 | |
| TNMF-AS + DNN-M | | 90.1 | |

Table 1: Comparing performance of the proposed systems on the development set of the DCASE 2017 dataset for different dictionary sizes K .

sion for classification (as the output layer of the DNN is a softmax layer), which can give us access to class-wise probability of each example. Therefore we perform a simple fusion by averaging the log-probabilities of different occurrences of each model in order to take the final decision.

4. EXPERIMENTAL EVALUATION

4.1. Dataset

We use the 2017 version of the DCASE challenge dataset for acoustic scene classification [16]. It contains 13 hours of urban audio scenes recorded with binaural microphones in 15 different environments split into 4680 10-s recordings. We use the same 4 training-test splits provided by the challenge, where 25% of the examples are kept for testing. The baseline for this dataset is DNN trained on shingled Mel energy representations [16].

4.2. Time-frequency representation

In order to build the data matrix \mathbf{V} , we average both channels of the audio and rescale the resulting mono signals to $[-1, 1]$. Next, we extract Constant-Q transforms with 24 bands per octave from 5 to 22050 Hz and with 30-ms non-overlapping windows using YAAFE [22], resulting in 291 dimensional feature vectors. The time frequency representations are then averaged by slices of 0.5 second resulting in 20 vectors per example. The length of the slices have been selected during preliminary experiments by choosing the value that maximized the performance of a simple unsupervised NMF classification system. After concatenating all the averaged slices for each example to build the data matrix, we apply a square root compression to the data and scale each feature dimension to unit variance.

4.3. Comparing the TNMF algorithm

The dictionary in the TNMF model is initialized with unsupervised NMF using the GPU implementation [23]. The classifier is updated using the LBFGS solver from logistic regression implementation of scikit-learn [24]. We set the regularization parameter to $\nu = 10$

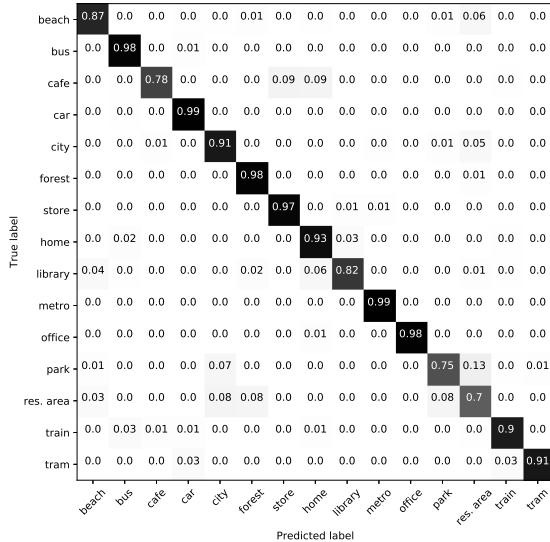


Figure 2: Normalized confusion matrix for the system fusion.

and the ℓ_1 and ℓ_2 regularization terms to $\lambda_1 = 0.2$ and $\lambda_2 = 0$. After preliminary experiments, the gradient step is set to $\rho = 0.001$ for the original algorithm [11] and to $\rho = 0.0005$ for the modified algorithm presented in Section 2.2. We follow the same decaying strategy as described in [19]. Moreover, contrary to our previous application of TNMF [8], we classify each slice individually and average the log-probabilities to take a decision on the full examples.

The performance of our adapted scaling algorithm is compared to the previously used TNMF algorithm for ASC in the first part of Table 1. The results are shown for different values of dictionary size K and are averaged over 10 initializations of the model. As a baseline we also include the results for an unsupervised NMF-based feature learning system classified with logistic regression. For both dictionary sizes, the proposed modifications for the algorithm allow for a notable increase in performance. This confirms that the introduced adaptive scaling strategy in the algorithm improves the model generalization capacity. Moreover the proposed TNMF system largely outperforms the dataset baseline as well as the unsupervised NMF-based systems.

4.4. Comparing DNN architectures

When used to train DNNS, the unsupervised NMF features are extracted using the Kullback-Leibler divergence with a ℓ_1 sparsity constraint $\lambda_1 = 0.2$. The DNNs are trained separately on the NMF activations or the time-frequency representation, we keep the same DNN architectures as proposed in our previous work on the same dataset [17]. The networks have 2 hidden layers of 256 units when having NMF features as an input and 3 hidden layers with 512 units when trained on the CQT representations. All layers have rectified linear unit (ReLU) [25] activations and dropout probability of 0.2 [26]. The models are trained with keras [27] using the stochastic gradient descent algorithm with default settings on 50 epochs without early stopping. The architecture of the merged CQT-NMF DNN model is described in Figure 2 and is trained with the same settings as just described.

The results of the compared DNN systems are reported in the second part of Table 2. First, as it was found in [17], training DNN directly on the NMF largely outperforms those trained directly on time-frequency representation as well as unsupervised NMF trained with a logistic regression. The proposed merged DNN architecture helps slightly increasing the performance compared to DNN trained only on NMF activations. Moreover it allows us to reach accuracies similar to the best TNMF system. However, as the networks are trained from unsupervised NMF features they require more components to attain the best results. We can also note that the TNMF-AS model still provides better results compared to the best DNN-M model which further confirms its usefulness for the task.

4.5. Fusion for final system

In order to further improve the results of our final system we augment the data by using both channels of the stereo signal instead of the mono mix used previously and apply late fusion to combine the output of the systems described above. The systems trained on both channels will be denoted as "TNMF-AS stereo" and "DNN-M stereo" and are reported in the fourth part of Table 2. The results are averaged over 10 different initializations of the models. Our final system fusion is computed with the following process similar to the one proposed in [11]:

- Train 4 initializations DNN-M-Stereo for $K = 1024$ on the 4 training sets and store the resulting 16 output log-probabilities
- Train 4 initializations TNMF-AS stereo $K = 512$ on the 4 training sets and store the resulting 16 output log-probabilities
- Average all log-probabilities in order to make the final predictions

The log-probabilities of the models trained on each of the 4 available training sets in the development set. We also report and submitted the system using only the log-probabilities computed with TNMF-AS which is denoted as TNMF-AS fusion. The accuracy obtained for the separate systems with both channels and the fusion systems are reported in the last row of Table 1. The accuracy scores on the development set go from 88.4 and 89.2% for TNMF-AS and DNN-M systems respectively to 90.1% with the fusion of both systems. The normalized confusion matrix for the last fusion system is shown in Figure 2. The majority of confusions comes from labels corresponding to similar scenes such as *park and residential area*, *library and office* or *restaurant and grocery store*. In fact, such pairs of acoustic environments tend to contain many event queues in common that could make it difficult for the systems to discriminate between them.

5. CONCLUSION

In this paper we have described the system we submitted to the 2017 DCASE challenge for acoustic scene classification. We use two established methods to learn nonnegative representations of acoustic scenes, a supervised NMF method and DNN trained on unsupervised NMF activation features. Moreover, we propose improvements to both methods in order to improve classification performance. We introduce a new adaptive scaling algorithm for TNMF and DNN architecture that learns from both the time-frequency representation and the NMF features. Finally we use a fusion of both methods as our final system. For future work, we could investigate in what respect NMF representation can be jointly learned with the networks in the TNMF.

6. REFERENCES

- [1] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, “Acoustic scene classification: Classifying environments from the sounds they produce,” *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.
- [2] A. Mesaros, T. Heittola, and T. Virtanen, “Tut database for acoustic scene classification and sound event detection,” in *Proc. of European Signal Processing Conference*, 2016.
- [3] 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,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.
- [4] X. Valero and F. Alías, “Gammatone cepstral coefficients: biologically inspired features for non-speech audio classification,” *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1684–1689, 2012.
- [5] G. Roma, W. Nogueira, and P. Herrera, “Recurrence quantification analysis features for environmental sound recognition,” in *Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.
- [6] A. Rakotomamonjy and G. Gasso, “Histogram of gradients of time-frequency representations for audio scene classification,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 1, pp. 142–153, 2015.
- [7] D. Battaglino, L. Lepauloux, L. Pilati, and N. Evansi, “Acoustic context recognition using local binary pattern codebooks,” *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–5, 2015.
- [8] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Feature learning with matrix factorization applied to acoustic scene classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1216–1229, June 2017.
- [9] J. Salamon and J. P. Bello, “Unsupervised feature learning for urban sound classification,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.
- [10] A. Rakotomamonjy, “Supervised representation learning for audio scene classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1253–1265, 2017.
- [11] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Supervised nonnegative matrix factorization for acoustic scene classification,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [12] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, “A comparison of deep learning methods for environmental sound,” *arXiv preprint arXiv:1703.06902*, 2017.
- [13] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, “DCASE 2016 acoustic scene classification using convolutional neural networks,” DCASE2016 Challenge, Tech. Rep., 2016.
- [14] 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,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [15] H. Phan, P. Koch, F. Katzberg, M. Maass, R. Mazur, and A. Mertins, “Audio scene classification with deep recurrent neural networks,” *arXiv preprint arXiv:1703.04770*, 2017.
- [16] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.
- [17] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Leveraging deep neural networks with nonnegative representations for improved environmental sound classification,” in *Proc. of Workshop on Machine Learning for Signal Processing*, 2017.
- [18] 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.
- [19] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.
- [20] P. Sprechmann, A. M. Bronstein, and G. Sapiro, “Supervised non-euclidean sparse nmf via bilevel optimization with applications to speech enhancement,” in *Proc. of Joint Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA)*. IEEE, 2014, pp. 11–15.
- [21] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [22] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, “Yaafe, an easy to use and efficient audio feature extraction software,” in *Proc. of International Society for Music Information Retrieval*, 2010, pp. 441–446.
- [23] R. Serizel, S. Essid, and G. Richard, “Mini-batch stochastic approaches for accelerated multiplicative updates in nonnegative matrix factorisation with beta-divergence,” in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.
- [24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., “Scikit-learn: Machine learning in python,” *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [25] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?” in *Proc. IEEE International Conference on Computer Vision*, 2009, pp. 2146–2153.
- [26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [27] F. Chollet, “keras,” *GitHub repository*, 2015.

CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR RARE SOUND EVENT DETECTION

Emre Çakir and Tuomas Virtanen

Tampere University of Technology
Finland
emre.cakir@tut.fi

ABSTRACT

Sound events possess certain temporal and spectral structure in their time-frequency representations. The spectral content for the samples of the same sound event class may exhibit small shifts due to intra-class acoustic variability. Convolutional layers can be used to learn high-level, shift invariant features from time-frequency representations of acoustic samples, while recurrent layers can be used to learn the longer term temporal context from the extracted high-level features. In this paper, we propose combining these two in a convolutional recurrent neural network (CRNN) for rare sound event detection. The proposed method is evaluated over DCASE 2017 challenge dataset of individual sound event samples mixed with everyday acoustic scene samples. CRNN provides significant performance improvement over two other deep learning based methods mainly due to its capability of longer term temporal modeling.

Index Terms— Sound Event Detection, Convolutional Neural Network, Recurrent Neural Network, Machine learning

1. INTRODUCTION

The aim of sound event detection (SED) is to temporally locate and label the sound event class(es) present in an acoustic signal. For an SED task, a set of target sound event classes should be determined. For instance, an SED task can be defined as the detection of dog barking, door bell, and baby crying sounds for any given acoustic signal. Recently, SED has been utilized in application areas such as wildlife bird audio monitoring [1, 2], audio surveillance [3], and multimedia event detection [4].

Recently, the research on SED has been mainly shifted from traditional classifier approaches such as Gaussian mixture models (GMM) - hidden Markov models (HMM) to deep learning based methods such as feed-forward neural networks (FNN) [5, 6], convolutional neural networks (CNN) [7], recurrent neural networks (RNN) [8], and convolutional recurrent neural networks (CRNN) [2, 9]. Feed-forward neural networks have the benefit of higher expressional capability over nonlinear functions compared to GMM-HMMs. However, their drawback is the fixed connections (each weight is connected to a fixed pair of neurons) which makes them less robust to slight spectral shifts in the acoustic features of the same sound event class. These slight shifts are a major factor in the inherent acoustic variability of sound event classes. This problem has mainly been overcome with the introduction of CNNs for SED, however the temporal context that can be modeled with CNNs is rather short. CRNN combines the long-term modeling capabilities of gated recurrent unit (GRU) [10] layers and the robustness of CNN to small spectral shift variations.

There are several difficulties on developing SED systems to be utilized in real-life environments. Some of these can be listed as the inherent acoustic variability of the sounds belonging to the same event class, overlapping (simultaneously occurring) sound events, environmental noise, variability in the acoustic characteristics of the background acoustic scene, and rarely occurring sound events.

The main problem encountered with the detection of the rare sound events using neural networks is the data imbalance. To elaborate, in an SED task, the classifier is trained to learn the relationship between the target class and its input representation, which is composed of acoustic features extracted in short time frames of an acoustic signal. During training, the classifier makes an estimation for the class presence probabilities for each frame, and calculates the error in the estimation through a loss function (which will be used to update the classifier parameters). In a rare SED task, the target class is not present in a significantly higher portion of time frames of each signal. Unless the training procedure of the classifier is adjusted correspondingly, the classifier will be biased on predicting "non-present" for all the frames, because it will reach low error even if it fails to detect the frames where the target class is present. Data imbalance is a very common problem in machine learning and methods such as data augmentation using time stretching and block mixing [8], oversampling [11] and synthesizing new samples through generative methods [12] have been previously proposed to limit the negative effect of data imbalance.

In this work, we propose to utilize CRNNs for combined single-class, rare SED in the presence of a real-life acoustic scene in the background. The convolutional layers of CRNN are used to extract shift invariant features from the input time-frequency representation. The gated recurrent layers are especially effective in detecting rare sound events, because they can reset and update their hidden/cell state to distinguish the features from a small number of consecutive time frames (corresponding to a rare target event) which are noticeably different from the features from the rest of the acoustic signal (corresponding to the background). The proposed CRNN method has been previously shown to provide state-of-the-art accuracy in both real-life and synthetic SED datasets [9] and QMUL bird audio detection challenge 2017 [2]. We follow the similar CRNN architecture and procedure as in [9], with the exception that we train separate CRNNs for each class due to the combined single-class approach. In addition, we slightly adjust the training procedure according to the evaluation metric of the given SED task (see Section 3.2). This work has a companion website at¹.

The rest of the paper is organized as follows. The acoustic features and the proposed CRNN method is explained in Section 2. In Section 3, the acoustic material, evaluation metric and the eval-

¹www.cs.tut.fi/~cakir/DCASE2017

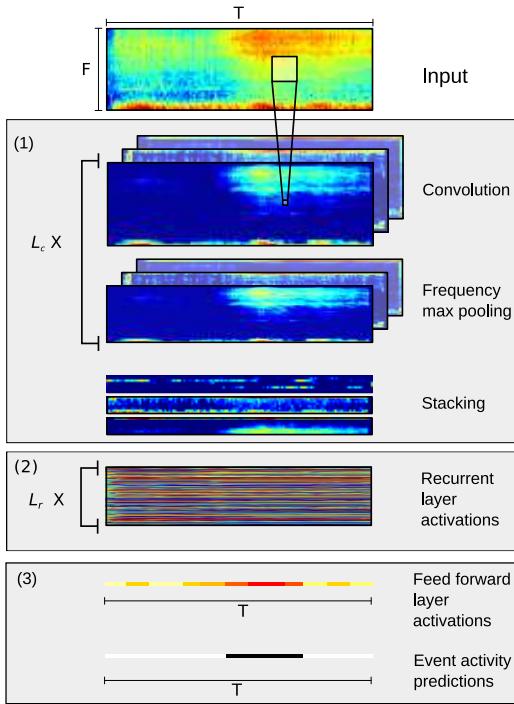


Figure 1: Overview of the proposed CRNN. (1): Multiple convolutional layers with max pooling in frequency axis, and stacking of the features over frequency axis (2): Gated recurrent layers, (3): feed-forward layer produces the event activity probabilities which are then binarized in evaluation/usage case.

uation results of the proposed method compared with the baseline methods is presented. Finally, our conclusions on this work are presented in Section 4.

2. METHOD

2.1. System Overview

The used SED approach consists of sound representation and frame-wise classification stages. In the sound representation stage, frame-level acoustic features are extracted for each time frame in the acoustic signal to obtain a feature matrix $\mathbf{X} \in \mathbb{R}^{F \times T}$, where $F \in \mathbb{N}$ is the number of features per frame and $T \in \mathbb{N}$ is the number of frames in the acoustic signal. In the classification stage, the task is to estimate the probabilities $p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta})$ for target output vector $\mathbf{y} \in \mathbb{R}^T$, where \mathbf{y} denotes the probability of the target event in each frame and $\boldsymbol{\theta}$ denotes the parameters of the classifier. Once the method is to be evaluated or utilized in a usage case, the event activity probabilities are typically binarized by thresholding, e.g. over a constant, to obtain binary event activity predictions $\hat{\mathbf{y}} \in \mathbb{R}^T$.

The classifier parameters $\boldsymbol{\theta}$ are trained by supervised learning, and the target outputs \mathbf{y} are obtained from the onset-offset annotations of the sound event class. If the sound event class is present during frame t , then y_t will be set to 1, and 0 otherwise.

In this work, SED is conducted in combined single-class manner, so the stages below are repeated separately for each class.

2.2. Acoustic Features

The acoustic features used in this work are log mel-band energies, as they have been shown to provide good performance on SED with deep neural networks [2, 6, 9]. Each audio sample is divided into 40 ms frames with 50% overlap and 40 log mel-band energy features are extracted from the magnitude spectrum of each frame. Each feature is then normalized independently to zero mean and unit standard deviation by using statistics calculated from the training data.

2.3. CRNN Architecture

The CRNN architecture used in this work consists of three main blocks: (1) convolution block, (2) recurrent block, and (3) classification block. The illustration of the architecture is given in Figure 1. The input for the CRNN are the acoustic features (log mel-band energies). In the convolution block, the input is fed to L_c consecutive convolutional layers with linear activation functions. Each convolutional layer is followed by batch normalization per feature map [13], a rectified linear unit (ReLU) activation function, a dropout layer [14], and a frequency domain max-pooling layer. At the end of the convolutional block, the extracted features over the CNN feature maps are stacked along the frequency axis.

Convolutional layers provide robustness to frequency shifts in the input features due to shared weight connections and max-pooling operation, and this is crucial to overcome the problem of intra-class acoustic variability for SED. However, as it has been shown previously in other works [9, 15], convolutional layers perform the best when the filter size is small, and this means the temporal context used in these layers is very short (typically less than two hundred milliseconds).

In the recurrent block, these stacked features are fed to L_r GRU layers where tanh and hard sigmoid activation functions are used for update and reset gates, respectively. Each recurrent layer produces outputs for each frame by using both the features extracted by the convolutional layers (or the previous recurrent layers) and the previous frame activations as input. Dropout is applied on both the inputs and the hidden state outputs of the recurrent layer [16].

GRU layers control the information flow through a gated unit structure. For frame t , the total activation of GRU layer is a linear interpolation of previous activation h_{t-1} and the candidate activation \hat{h}_t as

$$h_t = u_t \cdot h_{t-1} + (1 - u_t) \cdot \hat{h}_t \quad (1)$$

where u_t denotes the update gate. Candidate activation \hat{h}_t is a function of h_{t-1} , the GRU layer's input x_t and the reset gate r_t . GRU activation is mainly controlled by reset gate when the GRU layer's input x_t is significantly different than in previous frames. When reset gate is closed ($r_t = 0$), the candidate activation does not include any contribution from h_{t-1} . Fast response to the changes in the input and the previous activation information is crucial for high performance in rare SED, where the task is to detect a small of consecutive time frames where target event is present.

In the classification block, a feed-forward layer of single unit with sigmoid activation function is used as the classification layer. While computing the output of the classification layer, the same weight and bias values are used over the recurrent layer outputs for each frame. The contributions of GRU's previous and candidate activations to the classification output, namely c_{t-1} and \hat{c}_t , can be computed as

$$\begin{aligned} c_{t-1} &= w \odot (u_t \cdot h_{t-1}) \\ \hat{c}_t &= w \odot ((1 - u_t) \cdot \hat{h}_t) \end{aligned} \quad (2)$$

Table 1: CRNN hyper-parameters for each target class.

| Hyper-parameters | Baby cry | Glass break | Gun shot |
|------------------|----------|-------------|----------|
| L_c | 3 | 3 | 3 |
| pool size | (5,4,2) | (5,4,2) | (5,4,2) |
| L_r | 1 | 3 | 1 |
| # filters/units | 96 | 160 | 32 |
| # Parameters | 520K | 1750K | 59K |

where w is the weight vector that connects GRU layer and the classification layer, and \odot denotes element-wise multiplication. The outputs of the classification layer are regarded as the presence probabilities of the target class in each frame of the audio sample.

If the model is to be evaluated or utilized in a usage case, the presence probabilities are binarized with a constant threshold of 0.5 to get the binary presence predictions. These predictions are further post-processed with a median filter of length 540 ms.

3. EVALUATION

3.1. Acoustic Material

For the acoustic material, DCASE2017 challenge dataset has been used and detailed information on the dataset can be found in Section 4 of [17]. The dataset consists of samples from 15 different everyday acoustic scenes (park, home, street, cafe, train etc.), some of which are mixed with isolated recordings from at most one of the three different target sound event classes: baby crying, glass breaking and gun shot. The isolated recordings are divided into segments based on the signal energy levels, and the segments relevant to the target class are selected by a human annotator. Mixing is done by adding a segment to the 30-second long background acoustic scene sample with a random time offset. The mean duration of the isolated target sound event recordings is below 2.25 seconds for all three classes and each isolated event is present at most once for each mixed sample, making them active for only a short period of time (hence the task name rare sound event detection).

For the development set, 2973 training, 298 validation and 1496 test samples (4767 total) are generated through the code repository provided as a part of the DCASE challenge [18]. Although the probability of including isolated recordings in each mixed sample is set to 0.5 as default in the code provided by the challenge, we increase the probability of including target events from default 0.5 to 0.99 for training and validation samples. This change increases the percentage of the frames labeled as including a target event from 5% to 8% in the training data, which helps to ease the problem of data imbalance. This probability is kept at 0.5 for the test samples, as suggested by the challenge organizers, to be able to compare the development set results over the same conditions with other participants. In the evaluation set, the training and validation samples of the development set are combined into a single training set, test samples are used as the validation set, and the system is evaluated against an unseen set of 1500 samples (500 for each target class).

3.2. Procedure and Final Configuration

The CRNN is trained using Adam method for gradient based optimization [19]. Cross-entropy is used as the loss function. The network is trained for a maximum of 200 epochs. After each epoch

of training, validation set is evaluated for the event-based error rate (see Section 3.4) and the model at the epoch with the lowest error rate is saved in the memory. This way, we aim to align the training procedure with the evaluation metric of this work. If the error rate does not decrease for 25 consecutive epochs, the training is stopped and the last saved model is selected as the final model.

In order to decide the architecture to be used in the evaluation, we run a hyper-parameter grid search and pick the architecture with the lowest event-based error rate on the test set of the development data. The fixed hyper-parameters for each experiment is as follows. We use 5-by-5 size feature maps in convolutional layers, and dropout with probability 0.25 for both convolutional and recurrent layers. The grid search covers the number of convolutional feature maps (filters) / RNN hidden units (both are set to the same value) {32, 96, 160}; the number of recurrent layers {1, 2, 3, 4}; and the number of CNN layers {1, 2, 3, 4} with the following frequency max-pool sizes after each convolutional layer {(8), (4, 2), (2, 2, 2), (5, 2, 2), (5, 4, 2), (5, 2, 2, 1), (5, 2, 2, 2)}. The best performing CRNN hyper-parameters for each target class are listed in Table 1.

3.3. Baseline

In this work, we compare the performance of CRNN with two baseline methods using deep learning with the same input features. The first baseline method is a deep FNN with two hidden layers of 50 units, which is also the official baseline method for the challenge. The input features differ slightly in the sense that the extracted 40 log mel-band energy features are concatenated for five consecutive frames to gather temporal context, creating a feature vector with 200 entries. The second baseline method is the CNN. While selecting the CNN architectures to be used in evaluation, a very similar grid search procedure has been applied as explained in Section 3.2, the only difference being that the recurrent layers of the CRNN are replaced with the feed-forward layers to obtain CNN architecture.

3.4. Evaluation Metric

The official evaluation metric used in DCASE2017 challenge task 2 is the event-based error rate (ER) with onset tolerance of 500 ms. ER is the sum of insertion, deletion and substitution rates. ER is calculated as explained in detail in [20].

3.5. Results

The models used in the evaluation (hence the challenge submission) have been selected as the following. As a part of hyper-parameter grid search, 84 experiments have been run on development data for CNN and CRNN each. The evaluation models are then selected based on ER on test set of development data. We present four different CRNN methods for the rare SED challenge which are labeled as:

- CRNN-1: the architecture with the lowest ER on average over three classes. This model also happens to have the lowest ER on the "baby cry" class, and its parameters are given in the corresponding column of Table 1.
- CRNN-2: the ensemble of the seven best architectures with the lowest ER on average over three classes.
- CRNN-3: the architecture with the lowest ER for each of the three classes. The parameters for each of the architectures are given in Table 1.

Table 2: Event-based error rate for the baseline FNN and CNN methods and the proposed CRNN on the test set of evaluation data. Method indices are explained in Section 3.5.

| Method | Evaluation | | | |
|--------|-------------|-------------|-------------|-------------|
| | Baby cry | Glass break | Gun shot | Average |
| FNN | 0.80 | 0.38 | 0.73 | 0.64 |
| CNN-1 | 0.46 | 0.13 | 0.58 | 0.39 |
| CNN-2 | 0.38 | 0.15 | 0.53 | 0.35 |
| CNN-3 | 0.46 | 0.14 | 0.55 | 0.39 |
| CNN-4 | 0.42 | 0.14 | 0.53 | 0.36 |
| CRNN-1 | 0.27 | 0.07 | 0.20 | 0.18 |
| CRNN-2 | 0.18 | 0.10 | 0.23 | 0.17 |
| CRNN-3 | 0.27 | 0.14 | 0.47 | 0.29 |
| CRNN-4 | 0.21 | 0.11 | 0.24 | 0.19 |

- CRNN-4: the ensemble of seven best architectures with the lowest ER for each class.
- CNN methods have been obtained in the same fashion to CRNN methods as explained above.

The ensemble method is conducted as follows. Among the seven selected architectures, if four or more predict that the target class is not present in a given sample, then the final decision on the sample is that the target class is not present. Otherwise, the onset and offset values of the target class are selected as the median of the predicted onset and offset values for the sample. This ensemble method is used in order to get more reliable predictions over the onset and offset values and to filter the outlier predictions among the architectures with lowest ER.

The event-based ER results for the proposed and baseline methods have been presented in Table 2. CRNNs clearly provide better performance compared to both baseline methods for all target classes. In addition, by utilizing ensemble methods for both CNN and CRNN, the performance can be further improved, however this comes with an increased computational cost due to running several architectures in parallel. With the experiments for CRNN-1 method, we aimed to show if it is possible to find a single architecture that performs well for all three classes. For the development data, CRNN-1 provides comparable performance (0.16 vs 0.14 ER) with CRNN-3, where the best architecture is selected for each target class. For the evaluation data, as presented in Table 2, CRNN-1 performs even better than CRNN-3.

Regardless of the method, highest performance is obtained for glass break. Although the best performing architectures for each class differ significantly in the number of parameters (see Table 1), the median number of parameters among the seven best architectures are 687K, 806K and 774K for baby cry, glass break and gun shot, respectively. Therefore it is not possible to draw any direct conclusions on the relationship between the target class and the best performing architecture size.

3.6. An insight on GRU layer of CRNN

A case-study demonstration of the effect of GRU layers on the CRNN outputs is given in Figure 2. The CRNN architecture used to create this illustration consists of three convolutional layers and one GRU layer with 32 filters/units each, followed by a single unit classification layer. In panel (a), we can see that multiple GRU units respond to the change of input features around 4.5 second

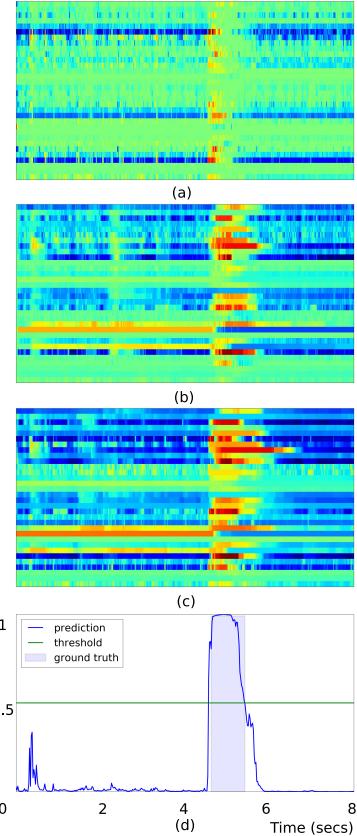


Figure 2: (a): contribution of current (candidate) activation \hat{c}_t , (b): contribution of previous activation c_{t-1} , (c): total contribution of GRU layer to the classification activation; (d): event activity probabilities vs. time for the first eight seconds of sample `devtest_babycry_001_1128b63726e9ed59ddc1bb944b3f22ce.wav`.

mark and trigger the candidate activation, as the target event starts to appear in the audio signal. After that, the GRU contribution is mainly controlled by the previous activations while the target event is still present, as shown in panels (b) and (c). Finally, the CRNN produces almost perfect detection of onset and offset for the given target event, as shown in panel (d).

4. CONCLUSIONS

In this paper, CRNN has been proposed for rare SED. CRNN has provided significantly improved performance over FNNs and CNNs for every target sound event class in DCASE 2017 challenge dataset. It is shown that the performance can further be improved using ensemble methods. For future work, improved ways to incorporate the evaluation metric into training procedure as the objective function can be considered. For instance, instead of aiming to directly match the target output and the predicted output for each frame, the objective function can be calculated over a window of frames, especially for the case when the onset and offset times can be tolerated to a certain degree.

5. REFERENCES

- [1] D. Stowell and D. Clayton, “Acoustic event detection for multiple overlapping similar sources,” in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2015, pp. 1–5.
- [2] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, “Convolutional recurrent neural networks for bird audio detection,” in *European Signal Processing Conference (EUSIPCO)*, 2017, accepted.
- [3] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, “Reliable detection of audio events in highly noisy environments,” *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.
- [4] Y. Wang, L. Neves, and F. Metze, “Audio-based multimedia event detection using deep recurrent neural networks,” in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2742–2746.
- [5] I. Choi, K. Kwon, S. H. Bae, and N. S. Kim, “DNN-based sound event detection with exemplar-based approach for noise reduction,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [6] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi-label deep neural networks,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.
- [7] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification and domestic audio tagging,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [8] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6440–6444.
- [9] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.
- [10] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.
- [11] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-smote: a new over-sampling method in imbalanced data sets learning,” *Advances in intelligent computing*, pp. 878–887, 2005.
- [12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [13] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448–456.
- [14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [15] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [16] Y. Gal, “A theoretically grounded application of dropout in recurrent neural networks,” *Advances in neural information processing systems*, 2016.
- [17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.
- [18] T. Heittola. (2016) Dcase2017 baseline system. [Online]. Available: github.com/TUT-ARG/DCASE2017-baseline-system
- [19] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [20] A. Mesaros, T. Heittola, and T. Virtanen, “Metrics for polyphonic sound event detection,” *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

THE SINS DATABASE FOR DETECTION OF DAILY ACTIVITIES IN A HOME ENVIRONMENT USING AN ACOUSTIC SENSOR NETWORK

Gert Dekkers^{1,2,}, Steven Lauwereins^{2,*}, Bart Thoen^{1,*}, Mulu Weldegebreal Adhana^{1,*}, Henk Brouckx^{3,*}, Bertold Van den Bergh^{2,*}, Toon van Waterschoot^{1,2}, Bart Vanrumste^{1,2,4}, Marian Verhelst², Peter Karsmakers¹*

¹ KU Leuven, Department of Electrical Engineering, Engineering Technology Cluster, Geel, Belgium.

² KU Leuven, Department of Electrical Engineering, Leuven, Belgium.

³ Vrije Universiteit Brussel, Department ETRO-DSSP, Brussels, Belgium.

⁴ IMEC, Leuven, Belgium.

ABSTRACT

There is a rising interest in monitoring and improving human well-being at home using different types of sensors including microphones. In the context of Ambient Assisted Living (AAL) persons are monitored, e.g. to support patients with a chronic illness and older persons, by tracking their activities being performed at home. When considering an acoustic sensing modality, a performed activity can be seen as an acoustic scene. Recently, acoustic detection and classification of scenes and events has gained interest in the scientific community and led to numerous public databases for a wide range of applications. However, no public databases exist which a) focus on daily activities in a home environment, b) contain activities being performed in a spontaneous manner, c) make use of an acoustic sensor network, and d) are recorded as a continuous stream. In this paper we introduce a database recorded in one living home, over a period of one week. The recording setup is an acoustic sensor network containing thirteen sensor nodes, with four low-cost microphones each, distributed over five rooms. Annotation is available on an activity level. In this paper we present the recording and annotation procedure, the database content and a discussion on a baseline detection benchmark. The baseline consists of Mel-Frequency Cepstral Coefficients, Support Vector Machine and a majority vote late-fusion scheme. The database is publicly released to provide a common ground for future research.

Index Terms— Database, Acoustic Scene Classification, Acoustic Event Detection, Acoustic Sensor Networks

1. INTRODUCTION

There is a rising interest in smart environments to enhance the human experience and/or quality of life of its inhabitant. Such a system aims to understand the home scene to provide smart functionality, e.g. security, health monitoring [2] and entertainment using different types of sensors including microphones. In the context of Ambient Assisted Living (AAL) persons are monitored, e.g. to support patients with a chronic illness and older persons, by tracking their activities being performed at home [3, 4, 5, 6].

In order to make a smart home capable to automatically anticipate to forthcoming scenarios some form of sensing capabilities need to be available. Numerous sensor modalities have been investigated ranging from wearable [7] to contact-less sensors [8]. Existing research

has been focussed either on a single modality or on the fusion of multiple modalities [6]. Compared to other modalities, microphone sensors are rarely used but contain highly informative data which can be exploited for multiple tasks [5]. Over the past decades, integrated components containing wireless radios and sensors are getting smaller in size, while maintaining computational power. This has led to using a network of sensors, which increases spatial sampling resolution. Therefore, this work focusses on using an Acoustic Sensor Network (ASN).

Another vital part of a smart home are the models that translate the data stream, acquired by the sensor(s), to information which can be used for a certain task. The task considered here is to detect an activity being performed, similar to the work in [3, 4, 5, 6]. When considering an acoustic sensing modality, an activity can be seen as an acoustic scene. The acoustic sensing literature has mainly covered the problems of Acoustic Event detection (AED) and Acoustic Scene Classification (ASC). An acoustic event is defined as a single consecutive event originated from a single sound source, e.g. a hand clap or a door knock. The ensemble of multiple events create a acoustic scenes describing a certain environment (e.g. a park or a living room) or, relevant to this paper, an activity being performed by a person (e.g. cooking or watching TV). Both the AED and ASC problems target the interpretation of the acoustic data. The rising interest in these problems has led to numerous public databases for a wide range of applications. The NAR dataset contains 41 sound events recorded by a humanoid robot Nao in a home environment [9]. The data used for the CLEAR 2006 and 2007 evaluations contain meeting room events collected by multiple microphones [10]. The DARES-G1 database contains annotations of sound events in different sound scenes, e.g. street and home [11]. The LITIS Rouen Audio Scene dataset [12], DCASE 2013 and 2016 databases [13, 14] consist of (binaural) recordings of events and/or scenes in public areas, e.g. office and park. The Multimodal subset of the SWEET-HOME database consists of recordings of daily activities performed by 21 different users leading to 26 hours of data. The recording setup consists of 7 microphone sensors, along with other sensor modalities, deployed in a smart home [6]. However, current databases do not possess all characteristics needed for our purposes: a) data collected in a home environment, b) activities being performed in a spontaneous manner, c) acquisition system based on an ASN, d) continuously recorded, and e) containing the activity when no person is present. Besides needing large databases to obtain accurate models and for algorithm validation, reference databases are important in algorithm development and comparison between algorithms.

*Thanks to VLAIO-SBO project Sound INterfacing through the Swarm (SINS) [1] for funding (contract 130006).

The main contributions of this paper are a) introducing a database, named "SINS", of real-life recordings in a home environment using an ASN and b) providing a baseline detection benchmark as a reference to future work using this database. The paper is organized as follows: Section 2 introduces the recording environment and sensing hardware. Sections 3 presents the database content and recording procedure. This includes statistics about its content and how the annotation was performed. Section 4 describes the baseline detection benchmark and evaluation procedure. Section 5 shows the performance of the baseline along with an analysis. Finally, Section 6 presents conclusions and future work.

2. RECORDING ENVIRONMENT AND SETUP

The database was collected in a vacation home with a floor area of 50 m^2 . The home consisted of five different rooms: a combined living room and kitchen, bathroom, toilet, bedroom and hall. Thirteen sensor nodes, each containing four microphones were distributed uniformly over the five rooms as indicated by Fig. 1. Details of the sensor's exact locations and height can be found in [15]. The sensor node has a modular design equipped with low-power audio sensing, audio processing and wireless capabilities [16]. The sensor node configuration used in this setup is a control board together with a linear microphone array. The control board contains an EFM32 ARM cortex M4 microcontroller from Silicon Labs (EFM32WG980) used for sampling the analog audio. The microphone array contains four Sonion N8AC03 MEMS low-power ($\pm 17\mu\text{W}$) microphones with an inter-microphone distance of 5 cm. Although not used in this work, the setup can be used for sound source localization [16]. The sampling for each audio channel is done sequentially at a rate of 16 kHz with a bit depth of 12. The acquired data is sent to a Raspberry Pi 3 for data storage. The data is stored in chunks of one minute and timestamped. Timestamps were obtained based on an NTP protocol for rough synchronization, between the sensor nodes, with a sample accuracy of ~ 500 ms. For algorithms demanding a more precise synchronization, an internal counter value of the control board is stored. The value was resetted every second by a GPS/Clock module. Using these counter values, a more precise synchronization (sample accuracy approximately $\sim 25 \mu\text{s}$) could be obtained using interpolation techniques.

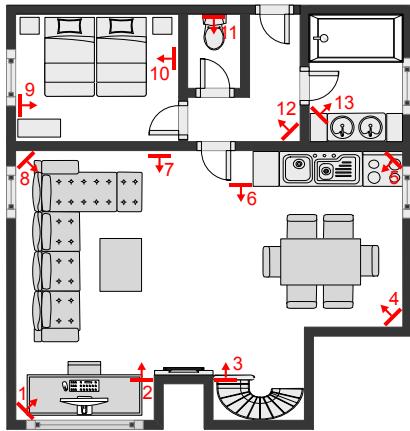


Figure 1: Floor map of the recording environment.

| Room | Activity | Nr. ex. | duration (min.) |
|-------------|-------------------|---------|---------------------|
| Living room | Phone call | 22 | 8.17 ± 13.73 |
| | Cooking | 19 | 16.62 ± 9.49 |
| | Dishwashing | 15 | 6.37 ± 1.49 |
| | Eating | 19 | 7.78 ± 4.27 |
| | Visit | 9 | 13.3 ± 12.11 |
| | Watching TV | 13 | 155.38 ± 93.28 |
| | Working | 49 | 31.24 ± 39.33 |
| | Vacuum cleaning | 13 | 4.79 ± 2.14 |
| | Other | 200 | 0.75 ± 0.95 |
| | Absence | 72 | 66.37 ± 130.30 |
| Bathroom | Drying with towel | 10 | 1.67 ± 0.28 |
| | Shaving | 13 | 1.91 ± 1.46 |
| | Showering | 10 | 6.11 ± 2.38 |
| | Toothbrushing | 19 | 1.41 ± 0.25 |
| | Vacuum cleaning | 9 | 0.87 ± 0.59 |
| | Other | 75 | 0.42 ± 0.4 |
| | Absence | 35 | 248.56 ± 263.62 |
| Hall | Vacuum cleaning | 9 | 3.31 ± 1.11 |
| | Other | 164 | 0.36 ± 0.22 |
| | Absence | 175 | 50.17 ± 102.52 |
| Toilet | Toilet visit | 21 | 4.74 ± 3.24 |
| | Vacuum cleaning | 7 | 0.53 ± 0.07 |
| | Absence | 31 | 282.75 ± 263.19 |
| Bedroom | Dressing | 28 | 1.53 ± 1.10 |
| | Sleeping | 7 | 348.43 ± 130.73 |
| | Vacuum cleaning | 7 | 1.04 ± 0.27 |
| | Other | 22 | 0.27 ± 0.23 |
| | Absence | 22 | 122.28 ± 157.43 |

Table 1: Recorded activities for each room.

3. DATABASE CONTENT AND RECORDING PROCEDURE

One person lived in the environment for a continuous duration of one week. In order to have an as realistic as possible data recording there was no predefined set of scenarios that were simulated by some actor. Consequently, the recorded scenarios included being absent (e.g. getting groceries and going for a walk) or even receiving visitors. Although there was no restriction on the activities being performed, the number of activities that were labeled was limited as indicated in Table 1. In total 16 different activities were annotated in five different rooms. Table 1 lists the different activities along with the amount of examples and the mean and standard deviation of the duration of all examples for each room. Most of the activities are self-explanatory, except for "Working" and "Other". "Working" contains recordings of the person doing work on a computer. The activity "Other" represents the presence of a person when not doing any activity of the ones listed in Table 1. Examples of recordings that are included in the "Other" activity are transitions between activities or the time between entering the room and starting an activity. In case of the *Living room* also sitting on the sofa or any other activity not listed in Table 1 was assigned to the "Other" category. In the case of the *Hall* this refers to crossing between rooms. Overall the database is strongly unbalanced, which indeed reflects the imbalance of different activities in daily life. In the case of the *Living room*, "Absence" and "Watching TV" are a factor 10 to 30 times larger in terms of total duration than the short-

est activities "*Vacuum cleaning*" and "*Other*". This ratio is even larger for the other rooms.

The annotation was performed in two phases. First, during the data collection a smartphone application was used to let the monitored person annotate the activities while being recorded. The person could only select activities listed in Table 1. The application was easy to use and did not significantly influence the transition between activities. Secondly, the start and stop timestamps of each activity were refined by using our own annotation software. In [15] more details are available on how these boundaries were chosen. During data collection we noticed occasional sensor node failure on two nodes. These were carefully annotated as well.

Postprocessing and sharing the database involves privacy-related aspects. Besides the person living there, multiple people visited the home. Also during the activity "*Phone call*", one can partially hear the person on the other end. A written informed consent was obtained from all participants. The database and annotation are publicly available [15].

4. DETECTION BENCHMARK

The provided baseline is adopted from earlier work on a similar problem [4]. It consists of a Mel-Frequency Cepstral Coefficients (MFCC) feature extraction and a Support Vector Machine (SVM) based classifier. Each sensor node performs feature extraction and detection locally on the first out of four microphone signals. The obtained class label is fused centrally using majority vote to obtain a final class label. A decision is obtained for each room separately. In case of the *Living room* and *Bedroom*, decisions from respectively eight and two different sensor nodes are combined. In the other rooms no fusion is needed.

First, in each sensor node, the audio stream is transformed by a Short-Time Fourier Transform with a 30 ms hamming window and a 10 ms step size. Then, a mel-scale filterbank is used of length 26 with a frequency range of 500 to 8000 Hz. The mel-features are transformed to a lower dimension using Discrete Cosine Transform. The first 14 coefficients were kept, including the 0th order coefficient. Delta (Δ) and acceleration ($\Delta\Delta$) coefficients were also computed, based on a window length of 9 MFCC frames. Subsequently, the MFCC $\Delta/\Delta\Delta$ feature vector stream was segmented using a sliding window of 15 s and a step size of 10 s. The window size is chosen based on the shortest average duration in Table 1. The mean and standard deviation are calculated for each feature dimension in the entire segment of 15 s which results in a total feature vector of length 84.

Finally, these features serve as an input to the model training and prediction phase of a SVM. SVM is a binary classifier that constructs a separating hyperplane such that the margin between two classes is maximized. For problems that are not linearly separable, a kernel maps the original space to a higher-dimensional space to make the separation easier. The kernel used here is the well-known radial basis function (RBF) kernel. To expand SVM to multi-class classification a 1-vs-1 coding scheme is used. The SVM hyper-parameters, kernel-bandwidth of the RBF and regularization parameter were tuned based on the training set [17]. Due to class imbalance, the contribution of each example in the model training phase is weighted based on class size.

The label assigned to the final feature vector is the active class at the middle of the segment window. Estimates of the current class therefore are based on non-causal information which introduces a delay of 7.5s in a practical setup. The feature vectors were grouped

based on which example (Table 1) it belongs to. These groups were randomly assigned to a fold to be used for 4-fold cross-validation. F_1 -score is used as a metric which does not take into account class imbalance. F_1 -score's are obtained for each class separately and averaged to obtain the overall F_1 -score.

5. RESULTS AND DISCUSSION

Results are analysed using normalized confusion matrices (nCM). For each room a confusion matrix is normalized by the marginals of either the column or the row. This provides insights into the precision and recall scores for each class and how the confusion is distributed. The precision is interpreted as how often the system is correct *when it estimates* a certain class, while recall provides insight into how often the system is correct *with respect to the ground truth*. In Fig. 2a classes are given on the y-axis. The precision (%) of each class is shown on the diagonal. The off-diagonals show the confusion with respect to a class on the diagonal in the same column. Therefore, the columns sum to 100%. For practical considerations, the values are rounded to the nearest integer. For example, in Fig. 2a, the class "*Working*" has a precision of 59% and is mostly confused with the class "*Absence*". The analysis is the same for the recall, where confusion should also be looked up vertically in the same column.

| | | | | | | | |
|-----------------|-----|-----|-----|-----|-----|-----|-----|
| Phone call | 87% | 1% | 1% | 10% | 1% | 1% | 3% |
| Cooking | 82% | 8% | 1% | 3% | 1% | 1% | 5% |
| Dishwashing | 7% | 71% | 4% | 1% | 1% | 2% | 8% |
| Eating | 2% | 2% | 76% | | 1% | | 2% |
| Visit | 6% | 1% | 1% | 53% | 1% | | 1% |
| Watching TV | | | | 1% | 98% | | 1% |
| Working | 1% | 1% | 2% | 10% | 1% | 59% | 15% |
| Vacuum cleaning | | | | | | 95% | 1% |
| Other | 2% | 7% | 14% | 6% | 6% | 5% | 2% |
| Absence | 4% | | | 2% | | 34% | 24% |

(a)

| | | | | | | | | |
|-----------------|-----|-----|-----|-----|------|-----|-----|-----|
| Phone call | 64% | | | | 12% | | | 1% |
| Cooking | 88% | 17% | 4% | 2% | | | 1% | 11% |
| Dishwashing | 1% | 3% | 55% | 2% | 1% | | 1% | 7% |
| Eating | 1% | 5% | 82% | 1% | | 2% | 1% | 4% |
| Visit | 9% | 1% | 1% | 56% | | | 4% | |
| Watching TV | 13% | | 2% | 25% | 100% | | 1% | 3% |
| Working | 4% | 4% | 10% | 7% | 1% | 90% | | 34% |
| Vacuum cleaning | | | | 1% | | | 96% | |
| Other | 2% | 3% | 9% | 2% | 1% | 2% | 1% | 29% |
| Absence | 6% | | 1% | 2% | | 5% | 1% | 6% |

(b)

Figure 2: Living room - (a) precision nCM - (b) recall nCM

Fig. 2 shows the output for the *Living room*. The averaged F_1 -score over all classes is $82.3 \pm 2.2\%$. The worst performing class is "*Other*" which gets confused with "*Working*" and "*Absence*". This seems logical because these three classes contain a great amount of silence. The class "*Absence*" contains audio when "*Vacuum cleaning*" is active in other rooms. The rooms in the vacation home were poorly acoustically isolated from each other and the door between *Hall* and *Living room* was partially opened due to the electricity cable of the vacuum cleaner. This, however, did not lead to high confusion between the two classes. The best performing classes are "*Absence*", "*Watching TV*", "*Vacuum cleaning*" and "*Cooking*" with F_1 -scores around 95%.

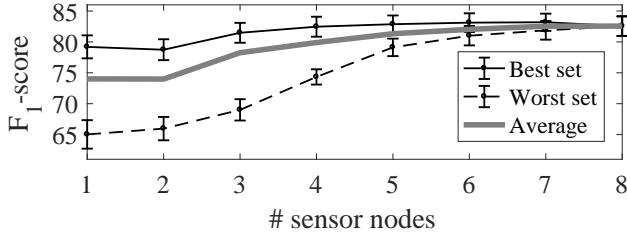
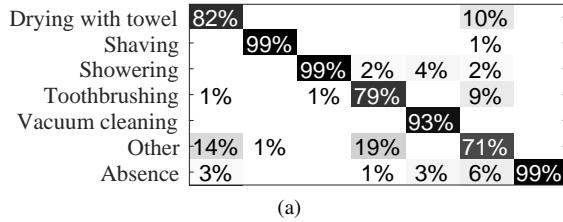
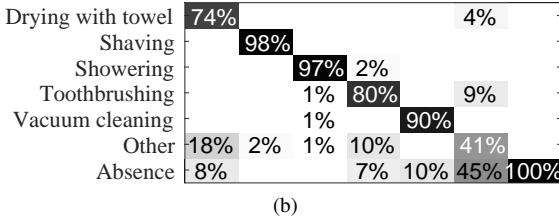
Figure 3: F₁-score versus amount of nodes used in the *Living room*

Fig. 3 shows the F₁-score with respect to the amount of nodes used. All possible sets of node combinations are tested. The best and worst set is shown together with the averaged performance over all sets. The gain in F₁-score, between using a single sensor or eight sensors, ranges between 3.4% and 17.3% depending on which sensor node is selected. On average the gain is 8.5%.



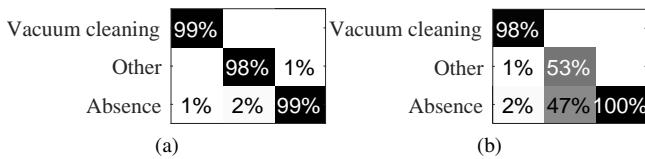
(a)



(b)

Figure 4: Bathroom - (a) precision nCM - (b) recall nCM

Fig. 4 shows the output for the *Bathroom*. averaged F₁-score over all classes is 84.8±2.0%. Similar trends are noticeable here compared to the *Living room*. The classes "Shaving", "Showering" and "Absence" perform above 95%. The worst performing classes are "Drying with towel" and again "Other". Most of these classes are, as expected, confused with "Absence" and "Other".



(a)

(b)

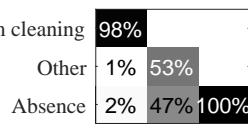


Figure 5: Hall - (a) precision nCM - (b) recall nCM

The output for *Hall* is shown in Fig. 5. The averaged F₁-score over all classes is 89.1±1.9%. The recall of class "Other" (53%) is considerably lower than the precision (98%). This shows that in case the system detects the class "Other" is active it is 98% correct, while when it should be "Other" is often confused with the class "Absence". This could be due to the relatively short duration of

the class "Other". The average duration is 21.6 s (Table 1), while detections are based on segments of 15 s. In the case of *Hall*, the class "Other" only contains transitions between rooms.

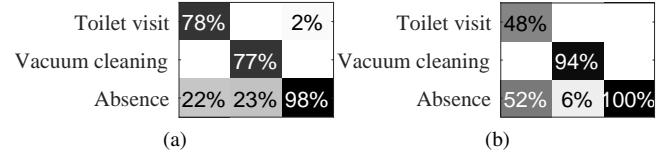


Figure 6: Toilet - (a) precision nCM - (b) recall nCM

Similar trends are observable for the results of *Toilet* in Fig. 6. The score of "Vacuum cleaning" is lower compared to other rooms due the shorter duration (31.8 s on average). The precision of "Toilet visit" (78%) is much higher than the recall (48%). A toilet visit is often without making any audible audio, causing the confusing with "Absence" for the recall nCM. When an event occurs however, it shows that these events often are recognized correctly leading to relatively high score. F₁-score over all classes is 79.0±8.6%.

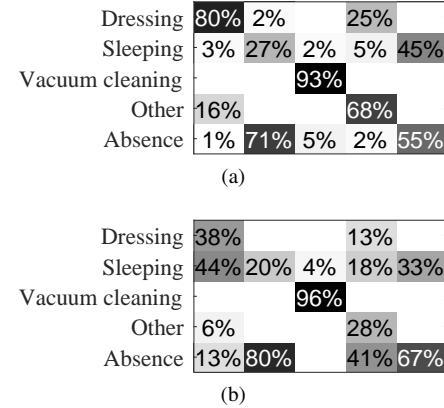


Figure 7: Bedroom - precision and recall CM

The output for *Bedroom* is shown in Fig. 7. The overall F₁-score over all classes is 52.5±3.5%. This is the lowest overall F₁-score over all rooms, as expected due to the class "Sleeping". The classes "Dressing", "Sleeping", "Other" and "Absence" are often confused between eachother. The only class performing above 90% is "Vacuum cleaning".

6. CONCLUSIONS

In this paper we a) introduced the "SINS" database, a real-world database for detection of daily activities in a multi-room home environment using an acoustic sensor network and b) provided a first analysis on the detection performance using a benchmark system. The best performing room was the *Hall* with an F₁-score of 89.1%. The worst performing room is the *Bedroom* with an F₁-score of 52.5%. Both the database and annotation is available for download [15]. Future work will focus on a) improving the benchmark system and b) extending this database with isolated acoustic events and annotations on a sound event level using the acoustic scene database. For both subsets it is foreseen to also provide a benchmark and make the data public.

7. REFERENCES

- [1] SINS. Sound INterfacing through the Swarm. [Online]. Available: <http://www.esat.kuleuven.be/sins/>
- [2] F. Erden, S. Velipasalar, A. Z. Alkar, and A. E. Cetin, "Sensors in assisted living: A survey of signal and image processing methods," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 36–44, March 2016.
- [3] L. Vuegen, B. Van Den Broeck, P. Karsmakers, H. Van hamme, and B. Vanrumste, "Automatic monitoring of activities of daily living based on real-life acoustic sensor data: A preliminary study," in *Proc. Fourth workshop on speech and language processing for assistive technologies (SLPAT)*, 2013, pp. 113–118.
- [4] ———, "Energy efficient monitoring of activities of daily living using wireless acoustic sensor networks in clean and noisy conditions," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, Aug 2015, pp. 449–453.
- [5] M. Vacher, F. Portet, A. Fleury, and N. Noury, "Development of audio sensing technology for ambient assisted living: Applications and challenges," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 2, no. 1, pp. 35–54, January 2011.
- [6] M. Vacher, B. Lecouteux, P. Chahuara, F. Portet, B. Meillon, and N. Bonnefond, "The Sweet-Home speech and multimodal corpus for home automation interaction," in *The 9th edition of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland, 2014, pp. 4499–4506. [Online]. Available: <http://hal.archives-ouvertes.fr/hal-00953006>
- [7] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, Third 2013.
- [8] G. Laput, Y. Zhang, and C. Harrison, "Synthetic sensors: Towards general-purpose sensing," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 3986–3999.
- [9] M. Janvier, X. Alameda-Pineda, L. Girin, and R. Horaud, "Sound representation and classification benchmark for domestic robots," in *Proc. IEEE International Conference on Robotics and Automation (ICRA14)*, May 2013.
- [10] A. Temko, C. Nadeu, D. Macho, R. G. Malkin, C. Zieger, and M. Omologo, "Acoustic event detection and classification," in *Computers in the Human Interaction Loop*, ser. Human-Computer Interaction Series. Springer, 2009, pp. 61–73.
- [11] M. Grootel, T. Andringa, and J. Krijnders, "DARES-G1: Database of Annotated Real-world Everyday Sounds," in *Proceedings of the NAG/DAGA Meeting 2009*, Rotterdam, 2009.
- [12] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 1, pp. 142–153, Jan. 2015.
- [13] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct 2015.
- [14] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*, Aug 2016, pp. 1128–1132.
- [15] KU Leuven, AdvISe research group. (2017) Datasets. [Online]. Available: <http://iiw.kuleuven.be/onderzoek/advise/datasets>
- [16] B. Thoen, G. Ottoy, F. Rosas, S. Lauwereins, S. Rajendran, L. De Strycker, S. Pollin, and M. Verhelst, "Saving energy in WSNs for acoustic surveillance applications while maintaining QoS," in *Proc. Sensors Applications Symposium (SAS)*, March 2017, pp. 1128–1132.
- [17] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

ACOUSTIC SCENE CLASSIFICATION BY ENSEMBLING GRADIENT BOOSTING MACHINE AND CONVOLUTIONAL NEURAL NETWORKS

Eduardo Fonseca, Rong Gong, Dmitry Bogdanov, Olga Slizovskaia, Emilia Gomez, Xavier Serra

Music Technology Group, Universitat Pompeu Fabra, Barcelona

name.surname@upf.edu

ABSTRACT

This work describes our contribution to the acoustic scene classification task of the DCASE 2017 challenge. We propose a system that consists of the ensemble of two methods of different nature: a feature engineering approach, where a collection of hand-crafted features is input to a Gradient Boosting Machine, and another approach based on learning representations from data, where log-scaled mel-spectrograms are input to a Convolutional Neural Network. This CNN is designed with multiple filter shapes in the first layer. We use a simple late fusion strategy to combine both methods. We report classification accuracy of each method alone and the ensemble system on the provided cross-validation setup of TUT Acoustic Scenes 2017 dataset. The proposed system outperforms each of its component methods and improves the provided baseline system by 8.2%.

Index Terms— acoustic scene classification, gradient boosting machine, convolutional neural networks, ensembling

1. INTRODUCTION

Humans have the ability to identify the environment where they are (e.g., park or beach) and recognize acoustic events that occur around them (e.g., a baby crying or a car passing by) from the incoming sounds they perceive. However, these tasks are not trivial for machine listening systems that attempt to accomplish them. The computational analysis of environmental sounds to automate tasks like the ones mentioned has recently received growing attention from the research community. The consecutive editions of the IEEE AASP Challenges Detection and Classification of Acoustic Scenes and Events“ (DCASE) provide the scenario where to evaluate and benchmark different approaches for acoustic scene classification and acoustic event detection [1]. In particular, DCASE 2017 challenge comprises four tasks: acoustic scene classification (task 1), detection of rare sound events (task 2), sound event detection in real life audio (task 3), and large-scale weakly supervised sound event detection for smart cars (task 4) [2]. This work concerns task 1, i.e., Acoustic Scene Classification (ASC), that can be defined as the task of associating a label to an audio stream thereby identifying the particular context or environment where the audio stream was generated [1]. The acoustic scene hence consists of all the acoustic information that is typically present in a given context, including background noises and specific acoustic events. ASC can trigger applications that range from audio collections management [3] and intelligent wearable interfaces [4] to the development of context-aware applications [5].

Traditionally, ASC systems have been based on a two-stage approach where *i*) pre-designed features or descriptors are extracted from the audio signal and *ii*) they are utilized as input to a classifier.

This *feature engineering* based approach relies heavily on the capacity of the features to capture relevant information from the audio signal for the task under consideration, which may require substantial expertise and effort. One of the most popular hand-crafted features in ASC are cepstral features, e.g., MFCCs, which have been taken from the speech recognition field and have been widely utilized for ASC [6, 7, 8]. Also, a number of low-level features computed either from the time or frequency domain (e.g., zero-crossing rate or spectral centroid) have been utilized [7]. Some typical examples of classifiers used for this task are GMM [6] and SVM [8], the latter being used in the winning system for DCASE 2013 challenge.

As opposed to the previous approach that relies on hand-crafting features, other techniques are based on *learning representations* from data. In particular, deep learning has recently become a widespread approach among the audio research community. In this case, the system is able to learn an internal representation from a simpler one at the input (typically, a time-frequency representation, e.g., spectrogram), and hence the two stages described before (feature engineering and classifier) are optimized jointly. Among the various deep learning approaches available, Convolutional Neural Networks (CNNs) have proved to be effective for several audio related tasks, e.g., speech recognition [9], automatic music tagging [10] or environmental sound classification [11]. In the specific case of ASC, several well-ranked submissions in the DCASE 2016 challenge were CNN-based, e.g., [12, 13]. Also in the context of DCASE 2016 challenge, a number of highly ranked submissions were based on the ensemble of different models, including the winning system [14], where the scores of a feature engineering based method (MFCC & i-vectors) were fused with those of a feature learning based method (CNN).

In this paper, we present a system for ASC that leverages both of the approaches presented above. On the one hand, a number of low-level time- and frequency-based audio features are extracted and input to a classifier. We decided to use Gradient Boosting Machine (GBM) due to its high performance as the winning solution in Kaggle challenges.¹ On the other hand, a CNN learns features from a log-scaled mel-spectrogram representation of the audio signal. By combining two methods of different nature, our intention is to obtain a system that takes advantage of the complementary information that they provide. The remainder of this work is organized as follows. Section 2 describes the methods (GBM and CNN) that compose our proposed system, as well as the late fusion strategy utilized. In Section 3 we present the dataset used and the evaluation setup that we follow. Results for the different methods (GBM, CNN and ensemble) are presented in Section 4 and we end this work with the conclusions in Section 5.

¹<https://www.kaggle.com/>

2. SYSTEM DESCRIPTION

2.1. Gradient Boosting Machine

Gradient boosting machine [15] is a powerful technique for building predictive models. It selects a loss as the objective function, and uses the additive model of many weak learners—typically regression trees—to minimize the loss. The parameters of added trees are tuned by a gradient descent algorithm. There are two GBM frameworks which are used widely in the data science community: XGBoost [16] and LightGBM.² The former is very popular among Kaggle community where it has been used for many competitions. The latter is a newcomer, which includes several improved features:

- It uses histogram based algorithms, which aggregate continuous features into discrete bins, to speed up training and reduce memory usage.
- It grows the tree by leaf-wise, which can reduce more loss than the level-wise algorithm.

In our experiments, we also found that LightGBM is faster than XGboost on training and achieves a slightly better overall classification accuracy. In consequence, we choose LightGBM as the GBM framework for the experiment.

2.1.1. Feature Extraction and Pre-processing

To consider the temporal characteristics, we segment each recording of 10s into 10 equal length non-overlapped sequences. We then extract features on each sequence using *FreesoundExtractor*,³ a feature extractor from Essentia open-source library for audio analysis [17]. This extractor is originally used by Freesound⁴ in order to provide sound analysis API and search by similar sounds functionality. It allows calculating hundreds of sound and music features. However, we discard some music-related features in rhythm, key, chords and tonal categories since we do not observe much musical trait in the development dataset. We further discard some feature statistics such as histogram and covariance matrix due to their high dimensionality and sparsity. The selected features and their dimensionality are listed in Table 1. The features are calculated on frame-level by using a 4096 samples frame size and a 2048 samples hop size. All other parameters are set to FreesoundExtractor default values. We then perform four statistical aggregations—mean, variance, mean of the derivative and variance of the derivative—to the frame-level feature vector of each sequence. Finally, a $\mathbb{R}^{820 \times 1}$ (205×4) feature vector is output for each sequence. In the cross-validation experiment (Section 3), we fit a mean and variance standardization scaler for each fold by using the features of the training dataset, which is then used for scaling the training and test set. In the final prediction step, we fit a standardization scaler for the whole development dataset and then apply it to the evaluation dataset.

2.1.2. LightGBM Parameters

Since ASC is a multiclass classification problem, we use logarithmic loss as the objective function, which yields a $\mathbb{R}^{15 \times 1}$ prediction probability for each sequence (considering 15 acoustic scenes). The three most important parameters are set as *i*) Learning rate: 0.05, *ii*)

Table 1: Selected features extracted by *FreesoundExtractor*. Dim: dimensionality.

| Feature name | Dim | Feature name | Dim |
|-------------------|-----|-------------------|-----|
| Bark bands energy | 32 | Tonal features | 3 |
| ERB bands energy | 23 | Pitch features | 3 |
| Mel bands energy | 45 | Silence rate | 3 |
| MFCC | 13 | Spectral features | 32 |
| HPCP | 38 | GFCC | 13 |

Number of trees: 500, and *iii*) Number of leaves: 255. All other parameters are default values. All parameters are held unchanged through the 4-fold cross-validation experiment and the model for the prediction of the evaluation dataset.

We keep all 820 dimensions features because according to our pilot experiment, removing irrelevant features only affected the training speed rather than improving the prediction accuracy. The parameter tuning process has also been simplified because several techniques in LightGBM such as *weak learner*, *Taylor approximation of the loss function* and *bagging*, make the system robust to over-fitting [18].

2.2. Convolutional Neural Networks

CNNs appear to be a reasonable choice for this task for various reasons. First, if they are presented with a time-frequency representation of audio, they are able, in theory, to capture spectro-temporal modulation patterns that can be relevant to identify the different acoustic scenes. Furthermore, when the input to the CNN is a time-frequency representation, the width and height dimensions of the convolutional filters can be related to the time and frequency axes, respectively.

2.2.1. Input Representation and Pre-processing

We use log-scaled mel-spectrogram as the input representation to the CNN. To compute it, first, the 2-channel wav files are down-mixed to mono, and short-time Fourier transform (STFT) is applied using Hamming windows of 40 ms with 50% overlap. After calculating its power, a mel filter bank is applied consisting of 128 bands ranging from 0 to 22050 Hz (the sampling rate being 44.1 kHz) according to Slaney’s formula [19]. Following results reported in [20], we use a filter bank with triangular filters in the frequency domain presenting a peak value of one. Finally, the resulting mel energy values are logarithmically scaled. The whole procedure was carried out using the Librosa library (v0.5.1) [21].

Resulting log-scaled mel-spectrograms are normalized to zero mean and unit standard deviation for the training set of every fold (see Section 3). Later on, the corresponding test set for every fold is standardized with the values from the training set normalization. Then, the spectrogram corresponding to every full 10s recording (consisting of 501 frames) is split into non-overlapping time-frequency patches (T-F patches) or *sequences* of 1.5s (i.e., 75 frames⁵). In this way, for every recording we obtain a total of 7 sequences (the last one being padded with the last original frame until reaching the desired duration). Every sequence, i.e., a T-F patch of $\mathbb{R}^{75 \times 128}$, is the input to the CNN and the minimum classification unit that will be aggregated to make decisions at the recording level.

²<https://github.com/Microsoft/LightGBM>
³http://essentia.upf.edu/documentation/extractors_out_of_box.html
⁴<https://freesound.org/>

⁵This duration is selected as the result of preliminary experiments, among durations ranging from 1 to 3s in steps of 0.5s.

2.2.2. Network Architecture

The proposed CNN architecture is depicted in Table 2.

Table 2: Proposed CNN architecture.

| |
|---|
| Input: 1 x (75,128) |
| <i>Conv1</i> : 48x (3,8) 32x (3,32) 16x (3,64) 16x (3,90) + BN + ReLU |
| Max-Pooling: (5,5) |
| <i>Conv2</i> : 224x (5,5) + BN + ReLU |
| Max-Pooling: (11,4) |
| Dense: 15 units + softmax |

The architecture is comprised of two convolutional layers alternated with max-pooling operations and a final fully connected dense layer. For the design of the convolutional filters, we hypothesize that for the acoustic material of many of the scenes under consideration, the spectro-temporal patterns are more relevant along the frequency domain (e.g., spectral envelope shapes and background noises) rather than in the time domain (e.g., onsets/offsets and attack-decay patterns of specific acoustic events). Most CNN architectures proposed in the literature use squared filters and only one filter shape in the first convolutional layer [11, 12, 14]. In contrast, some recent works suggest to employ, in the first layer, *i*) filter shapes that are not squared but rectangular, and *ii*) different co-existing filter shapes. This has shown to be effective for learning timbre representations in music audio classification tasks [22], and for learning features at multiple time resolutions in acoustic event recognition [23]. Motivated by those works and based on our assumption above, we decided to experiment with several configurations of filters with multiple *vertical shapes*⁶ in the first layer (results are reported in Section 4.2). By doing this, we intend to aid the learning process towards what we intuitively assume as more important for our task. To implement this, the first convolutional layer is, in turn, an ensemble of several convolutional layers, each one with filters of one shape, that are eventually merged. In order to obtain feature maps of the same size, zero-padding is applied to the network’s input.

Filter shapes are specified in Table 2 as *number of filters x (time, frequency)*. The number of filters are 112 and 224 for the first and second convolutional layers, respectively. The proposed final architecture presents, in *Conv1*, four different sets of filters, each of them presenting one different shape. For simplicity, and based on initial experiments, it was decided to use a fixed time dimension of 3 for all filters in this layer. In *Conv2* filters are squared. All filters have unitary stride in both dimensions. In both convolutional layers L2 regularization is applied with a parameter of 10^{-5} .

After every convolutional layer, batch normalization (BN) is applied [24] and the activation function is Rectified Linear Unit (ReLU) [25]. We use max-pooling after the two convolutional layers, which provides downsampling of the feature maps while adding some invariance along the time-frequency dimensions. More specifically, after *Conv1*, max-pooling is applied over squares of dimension 5. After *Conv2*, the pooling operation is designed to be global in the time domain so as to select only the most prominent feature, and with a dimension of 4 in the frequency domain, following previous work [12]. After the last max-pooling operation, resulting feature maps are flattened. Finally, the output layer is a dense layer,

⁶We denote vertical filters as those whose frequency dimension is much larger than its time dimension.

followed by a softmax activation function with 15 output units corresponding to the 15 acoustic scenes.

Network weights are initialized with a uniform distribution. The loss function is categorical cross-entropy and the optimizer is Adam with a learning rate of 0.001. The training is stopped through early stopping if the validation accuracy is not improved during 15 epochs, up to a maximum of 200 epochs. Training samples are shuffled between epochs, so that the batches (of size 64) are formed differently in order to increase data variability. The system is implemented using the Keras library (v2.0.2) [26].

2.3. Late Fusion

We use *arithmetic mean* as the late fusion method, which combines prediction probabilities from GBM and CNN systems by taking the arithmetic mean of the probabilities for each recording:

$$\text{pred}^i = \text{argmax}\left(\frac{\text{proba}_{\text{GBM}}^i + \text{proba}_{\text{CNN}}^i}{2}\right) \quad (1)$$

where $\text{proba}_{\text{GBM}}^i$ and $\text{proba}_{\text{CNN}}^i$ are respectively the prediction probabilities for the recording i from GBM and CNN systems; pred^i is the predicted label. This strategy led to better results than geometric mean.

3. EXPERIMENTS

3.1. Dataset

We use the *TUT Acoustic Scenes 2017* dataset, which is split into a development dataset and an evaluation dataset, of 4680 and 1620 audio recordings respectively. The development dataset is provided at the beginning of the challenge, together with ground truth. It includes 15 acoustic scenes⁷ each of them containing 312 recordings of 10s. A four-fold cross-validation setup is provided so as to make results reported strictly comparable. Along with the dataset, the challenge provides a two-layers multilayer perceptron (MLP) baseline system. Its prediction accuracy is reported in Table 3.

3.2. Evaluation Setup

We use the development dataset for training and testing both GBM and CNN models, according to the suggested four-fold cross-validation setup. Since no parameter tuning is performed for GBM, we use the entire training set in each fold to train the model. For the CNN, a 15% validation set is randomly split from the training data of every class in each fold to early-stop the training process. As explained in Sections 2.1.1 and 2.2.1, the output of the GBM and CNN models for every input sequence is a $\mathbb{R}^{15 \times 1}$ vector with the probabilities of the sequence belonging to every label. The class prediction at the recording level is computed by averaging class-wise scores across sequences and finding the class with the maximum average score. For the final proposed ensemble system, we carry out the late fusion as explained in Section 2.3. Figure 1 shows the evaluation process. The metric used is classification accuracy, i.e., the number of correctly classified audio recordings divided by the total amount of recordings and we report the average accuracy across the four folds. The evaluation dataset is used to predict acoustic scenes with our final proposed system for the challenge submission.

⁷A list of the scenes together with more details about the dataset can be found in <http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification>.

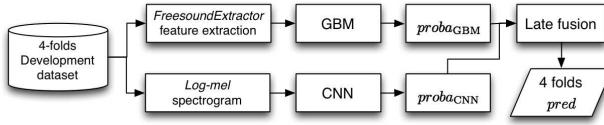


Figure 1: Evaluation diagram.

4. RESULTS AND DISCUSSION

4.1. Gradient Boosting Machine

The four-fold mean prediction accuracy of our LightGBM system is 80.8%, which improves the MLP baseline by 6%. The simplicity of this method—using an out-of-box feature extractor and no extensive parameter tuning—shows the suitableness of the Essentia extracted features for this task and the robustness of the LightGBM system.

4.2. Convolutional Neural Networks

We experimented with different filter configurations in the first convolutional layer. Accuracy results for the architectures using those configurations are listed in Table 3, together with the accuracy of the MLP-based baseline.

Table 3: ASC performance using the proposed CNN with various filter configurations in the first layer.

| System | Filter configuration #filters x (t, f) | #params | acc (%) |
|---------|---|---------|-------------|
| MLP | - | - | 74.8 |
| CNN Q=1 | 112x (5,5) | 648k | 77.8 |
| CNN Q=1 | 112x (3,40) | 659k | 78.1 |
| CNN Q=2 | 64x (3,20) 48x (3,70) | 660k | 78.7 |
| CNN Q=3 | 48x (3,10) 32x (3,30) 32x (3,60) | 656k | 79.6 |
| CNN Q=4 | 48x (3,8) 32x (3,32) 16x (3,64) 16x (3,90) | 657k | 79.9 |

We design filter configurations with Q sets of filters in the first layer, every set presenting one filter shape. Therefore, Q refers to the number of different filter shapes. Every set of filters can have a different number of filters, as seen in the second column of Table 3, but the total amount of filters is always 112. For every configuration, the filters’ frequency dimensions were defined of different lengths in order to cover spectral signatures of different nature, i.e., ranging from narrow patterns to patterns that are more spread in frequency. By adjusting the number of filters for each set and the filters’ dimensions, it was intended to keep the amount of network parameters approximately constant (except for the case of squared filters), so that results are comparable. Accuracies reported are the outcome of averaging results of three runs of every experiment. Although a more thorough study would be required to draw strong conclusions on the specific effect of varying *i*) the number of filter shapes, *ii*) the number of filters per shape, and *iii*) the dimensions of the filters, it seems that the combination of vertical filters and different filter shapes in the first layer is beneficial for the task. Since the case $Q=4$ provides the best results (a 5.1% improvement with respect to the MLP baseline), we use it in the proposed architecture.

4.3. Late Fusion

As explained in Section 2.3, the scores from GBM and CNN for every recording are averaged to produce the final system scores, from

which the acoustic scene label is predicted. After this late fusion, the system provides a classification accuracy of 83.0% on the development set, which means an improvement of 8.2% with respect to the MLP baseline. Further, it implies an improvement of 2.2% and 3.1% when compared to the GBM and CNN approaches, respectively.⁸ This demonstrates that both models provide complementary information and their fusion is able to increase performance substantially, even with a very simple fusion method. We believe the performance can be further improved by employing more sophisticated fusion strategies, e.g., logistic regression. Figure 2 shows the confusion matrix for the proposed ensemble system, where it can be seen which acoustic scenes are misclassified the most. The worst case occurs clearly between ‘residential area’ and ‘park’, which are perceptually very similar. Also, the system often confuses ‘tram’ and ‘train’, ‘grocery store’ and ‘cafe/restaurant’, and ‘library’ and ‘home’. Confusion matrixes for the GBM and CNN along with additional discussion and materials can be found in⁹.

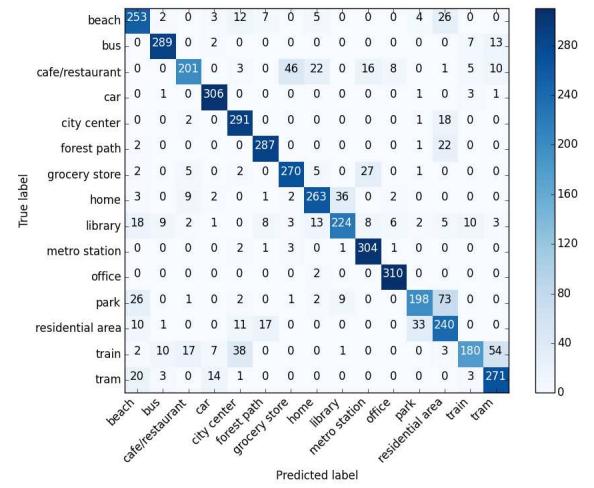


Figure 2: Confusion matrix for the proposed ensemble system evaluated on the development dataset.

5. CONCLUSION

This work proposes a system for ASC that consists of the ensemble of two methods of different nature: one that inputs a collection of hand-crafted features to a GBM, and a CNN that learns representations from log-scaled mel-spectrograms. We have shown how a simple late fusion of them already brings substantial performance improvement, which demonstrates that they provide complementary information beneficial for ASC. The proposed system achieves a classification accuracy of 83.0% on the TUT Acoustic Scenes 2017 development dataset. We believe that the proposed approach of combining two methods of different nature can be generalizable to other audio processing tasks, and we intend to test its effectiveness beyond ASC.

⁸Although the accuracy obtained by the GBM is higher than that of the CNN, their comparison is not totally fair. The latter uses 15% of the training data as validation set while the former uses the entire training set for training.

⁹<https://edufonseca.github.io/DCASE2017-Task1-ASC/>

6. ACKNOWLEDGMENT

This work is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688382 "AudioCommons", and the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583), and the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). We are grateful for the GPUs donated by NVidia.

7. REFERENCES

- [1] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.
- [2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.
- [3] C. Landone, J. Harrop, and J. Reiss, "Enabling access to sound archives through integration, enrichment and retrieval: The EASAIE project." in *ISMIR*, 2007, pp. 159–160.
- [4] Y. Xu, W. J. Li, and K. K. Lee, *Intelligent wearable interfaces*. John Wiley & Sons, 2008.
- [5] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications*. IEEE, 1994, pp. 85–90.
- [6] 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," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.
- [7] 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," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.
- [8] G. Roma, W. Nogueira, and P. Herrera, "Recurrence quantification analysis features for auditory scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [9] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.
- [10] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.
- [11] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.
- [12] 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*, 2016, pp. 95–99.
- [13] Y. Han and K. Lee, "Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification," DCASE2016 Challenge, Tech. Rep., 2016.
- [14] 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," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.
- [15] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
- [16] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.
- [17] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, X. Serra, *et al.*, "Essentia: An audio analysis library for music information retrieval," in *ISMIR*, 2013, pp. 493–498.
- [18] H. Zhang, S. Si, and C.-J. Hsieh, "GPU-acceleration for Large-scale Tree Boosting," *ArXiv e-prints*, June 2017.
- [19] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, p. 1998, 1998.
- [20] Q. Kong, I. Sobieraj, W. Wang, and M. D. Plumley, "Deep neural network baseline for DCASE challenge 2016," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*.
- [21] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloscheck, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017. [Online]. Available: <https://doi.org/10.5281/zenodo.293021>
- [22] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," *arXiv preprint arXiv:1703.06697*, 2017.
- [23] 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.
- [24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [25] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.
- [26] F. Chollet, "Keras," <https://github.com/fchollet/keras>, 2017.

ACOUSTIC SCENE CLASSIFICATION USING SPATIAL FEATURES

Marc C. Green and Damian Murphy

Audio Lab
Department of Electronic Engineering
University of York
York, UK

ABSTRACT

Due to various factors, the vast majority of the research in the field of Acoustic Scene Classification has used monaural or binaural datasets. This paper introduces EigenScape - a new dataset of 4th-order Ambisonic acoustic scene recordings - and presents preliminary analysis of this dataset. The data is classified using a standard Mel-Frequency Cepstral Coefficient - Gaussian Mixture Model system, and the performance of this system is compared to that of a new system using spatial features extracted using Directional Audio Coding (DirAC) techniques. The DirAC features are shown to perform well in scene classification, with some subsets of these features outperforming the MFCC classification. The differences in label confusion between the two systems are especially interesting, as these suggest that certain scenes that are spectrally similar might not necessarily be spatially similar.

Index Terms— Acoustic scene classification, MFCC, gaussian mixture model, ambisonics, directional audio coding, multichannel, eigenmike, soundscape

1. INTRODUCTION

Since the recent increase in research into Acoustic Scene Classification (ASC) sparked by the DCASE challenges [1], the vast majority of work has focused on identifying scenes based upon mono or, at most, stereo recordings. The potential for utilising more detailed spatial properties of acoustic scenes extracted from microphone array recordings remains largely unexplored. This is partly due to inheritance of techniques from the more mature fields of Automatic Speech Recognition (ASR) and Music Information Retrieval (MIR), which often have a “perceptually motivated” approach [2] (particularly with ASR), and partly due to the common focus of ASC research on applications including use in wearable technology, smartphones and robotics, [3] where utilisation of large microphone arrays would not be practical.

Another potential application of ASC is in environmental sound research, where the focus is not on human perception or portability *per se*, but rather on obtaining a detailed understanding of the acoustic environment itself. Such detailed analysis could assist in urban planning and legislation surrounding environmental sound. The L_{Aeq} metric currently in widespread use measures the average Sound Pressure Level (SPL) over a given period of time [4], not taking into account the content of the sound. Advanced machine listening techniques could be used to provide more nuanced measures of sound to better inform acoustic surveyors and so augment the L_{Aeq} measure. This content-focused approach to acoustic assessment has been called the “soundscape approach”, as opposed to

the “environmental noise” approach of the majority of legislation [5].

Given this application, the limitation to low-channel-count audio is not necessary. This paper investigates the possibility of classifying acoustic scenes based upon spatial features, comparing this with the use of standard Mel-Frequency Cepstral Coefficient (MFCC) features, and is organised as follows: Section 2.1 briefly introduces the EigenScape dataset, including justification of its necessity in context with previously-released acoustic scene recordings. Section 2.2 details the methods used to extract features from the recordings, whilst Section 2.3 describes the system used to classify the data. Section 3 presents results from this study, with Section 4 providing brief additional discussion. Section 5 concludes the paper by summarising the findings of this experiment.

2. METHOD

2.1. Dataset

In order to undertake this research, a large database of spatially-recorded acoustic scenes was required, including many examples of the same kinds of locations. This allows for separation of the dataset into separate training and testing sets where there is no crossover in recording locations between the two sets, avoiding the situation that gave rise to artificially inflated results in [6] where training and testing sets included segments from the same longer original recordings.

A set of 1st-order Ambisonic recordings was made as part of the DCASE 2013 challenge [1], however these were recordings of staged office environments only, and so did not provide the variety of recording environments needed for this research. The DEMAND dataset [7] contains sets of three multichannel recordings each of six different acoustic scene classes, however this is still too small a corpus for this project and their use of a nonstandard microphone grid layout could potentially make calculation of spatial features more difficult. The TUT database [8] used in DCASE challenges since 2016 features an appropriately broad range of examples of multiple acoustic scene classes, but features two-channel recordings only. A new set of recordings, the EigenScape dataset, was therefore created for this project.

The EigenScape dataset was recorded using the mh Acoustics EigenMike [9], a 32-channel spherical microphone array capable of making recordings in 4th-order Ambisonic format. Eight 24-bit/48 kHz ten-minute recordings each of eight different classes of location - Beach, Busy Street, Park, Pedestrian Zone, Quiet Street, Shopping Centre, Train Station and Woodland - were made at locations across the north of England, giving a total of 64 recordings. The location

classes were inspired by the selections in the TUT dataset, but with small indoor locations discarded, reflecting the focus of this work on the acoustic scenes of public places. Only the 1st-order channels from the 4th-order recordings were used in the present work.

Detailed information on the recording process for this dataset will be published in a future paper and the data will be made publicly available in due course. It is hoped that this data will prove useful to research in both acoustic scene and event detection.

2.2. Feature Extraction

The librosa library [10] was used to extract MFCC values from the omni channel (W) of the recordings. The audio was first resampled to half the original sampling rate before 20 MFCC values were extracted. These therefore covered the frequency range up to 12 kHz. The librosa standard frame length of 2048 samples with 25% overlap was retained.

To extract spatial features, the audio was resampled as before and filtered using a bank of FIR filters into 20 mel-spaced frequency bands in order to maintain parity in terms of frequency bands with the MFCC values. This enabled use of combined MFCC and spatial audio features for each band. Directional Audio Coding (DirAC) analysis [11, 12] was used in order to gain Direction of Arrival (DOA) estimates \mathbf{D} for each frequency band as follows:

$$\mathbf{D} = -\mathbf{P}\mathbf{U} \quad (1)$$

where \mathbf{P} is a matrix containing the 20 mel-filtered versions of the W-channel of each audio file and \mathbf{U} is a three-dimensional matrix containing the 20 filtered versions of the X, Y and Z-channels. The resultant matrix \mathbf{D} was split into time-frames corresponding to the frames used in the MFCC calculations, and mean values of \mathbf{D} were calculated for each frame. Angular values for azimuth and elevation in degrees for each frame were calculated based on this and used as features.

Secondly, a figure for diffuseness ψ in each frequency band was calculated as follows [11]:

$$\psi = 1 - \frac{\|\mathbf{D}\|}{c\{\mathbf{E}\}} \quad (2)$$

where c is the speed of sound, $\{\cdot\}$ represents the mean-per-frame values as described previously, and:

$$\mathbf{E} = \frac{1}{2}\rho_0 \left(\frac{\mathbf{P}^2}{Z_0^2} + \|\mathbf{U}\|^2 \right) \quad (3)$$

where ρ_0 is the mean density of air and Z_0 is the characteristic acoustic impedance of air. Combining all of these features results in a 60-dimensional feature vector output from the DirAC analysis.

2.3. Classification

For classification, each ten-minute recording was split into 30-second segments. In order to facilitate cross-validation, the data was split into four folds, whereby for each fold, six examples of each location would be used for training, with the remaining two examples used for testing. In this way, segments from the same recording location could not feature in both training and testing sets.

A Gaussian Mixture Model (GMM) system was used as classifier. Each scene class was assigned a ten-component GMM, which was trained using features extracted from the training fold audio using the expectation-maximisation algorithm [13]. GMMs with more components were tested but found to not substantially

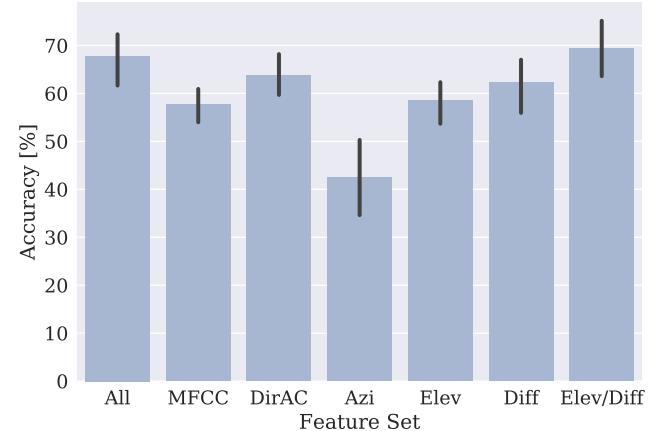


Figure 1: Mean and standard deviation classification accuracy using various feature subsets across all folds.

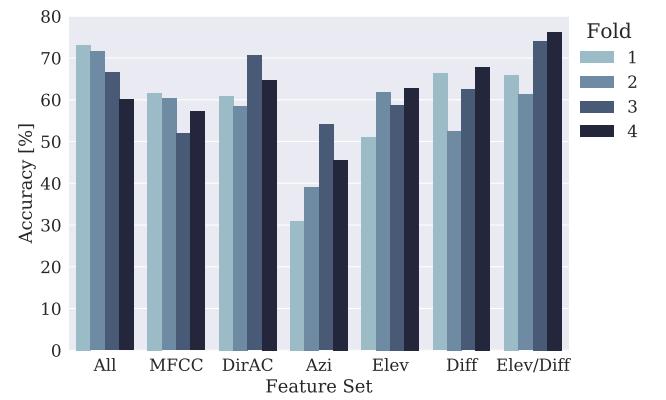


Figure 2: Accuracy of classifiers using various feature subsets across each data fold.

outperform the ten-component versions. Models were trained using all of the extracted data - an 80-dimensional concatenation of MFCC and DirAC features - and subsets thereof, including MFCCs alone (20-dimensional feature vector), individual DirAC features (20-dimension), and a combination of Elevation and Diffuseness (40-dimension).

To classify the testing data, features from test fold frames were given probability scores by each GMM and these scores were totalled across each 30-second segment. The segments were classified based on the GMM that had given its features the highest total probability score across all frames. This is essentially identical to the simple-minded audio classifier (smacpy) [14] system used as the baseline in the DCASE 2013 challenge [1].

3. RESULTS

3.1. Overall Accuracies

Figure 1 shows the mean performance accuracy of the classifiers using MFCC features, DirAC features, a combination of all features and subsets of the DirAC features. Using the MFCC features, the classifier has an average accuracy of 58%. This is consistent with

the performance of this type of classifier as reported in the literature, with Lagrange *et al* reporting 48% accuracy [15], and the DCASE 2013 baseline system giving 55% accuracy [1]. The DCASE 2016 baseline performs markedly better, though this system uses additional delta and acceleration MFCC features [8].

The DirAC spatial features outperform the MFCC features on average, at 64% accuracy as opposed to 58%. Figure 2 shows the classification accuracy in individual folds of the data. The MFCCs perform marginally better than DirAC in folds 1 and 2, but DirAC outperforms MFCCs by a larger margin in folds 3 and 4. Combining both sets of features leads to markedly improved performance relative to either alone in folds 1 and 2, with accuracies greater than 70% where each feature set alone gives accuracies closer to 60%. In the 3rd and 4th folds, however, adding the MFCCs to the DirAC features causes a decrease in accuracy of around 5% relative to using DirAC features alone.

Looking at the three sets of DirAC features individually, the elevation values alone give similar performance to the MFCCs, whereas diffuseness alone performs somewhat better than the MFCCs, except in fold 2. Using the Azimuth values alone gives the worst performance of any of the feature sets used here, averaging just 42% accuracy across all folds, and performing as badly as 31% accuracy in fold 1, though in fold 3 the accuracy is comparable to the MFCC performance. This is probably due to the fact that, whilst there should be some consistency of azimuth DOA values in similar acoustic scenes (indeed this is borne out by the fact that this classifier still performs better than chance), these azimuth values will be much more sensitive to the specific orientation of the microphone array when the recordings of the sound scenes were made. If one street scene, for instance, was recorded with the front of the microphone array facing the road, whereas another was recorded with the front parallel to the road, this will result in inconsistent azimuth values between the two scenes.

By contrast, elevation and diffuseness values should theoretically be independent of microphone rotation. This could account for the relatively high accuracy results when using these features. Because of the low accuracy results from the azimuth data, a new DirAC classifier was trained excluding this data. The results from this classifier are plotted in Figures 1 and 2 as ‘Elev/Diff’. This combination of elevation and diffuseness data was the best performing feature set on average, at 69% accuracy. With fold 4, these features give an accuracy of 76%, which was the maximum accuracy achieved in this test. There is, however, once again a marked difference in accuracy - around 10% - between the performance of the classifier in folds 1 and 2 (66%, 61%) relative to folds 3 and 4 (74%, 76%).

3.2. Classifier Confusion

Figures 3 and 4 show confusion matrices describing the classifications made by the MFCC and Elevation/Diffuseness (E/D) classifiers across all folds. Rows represent the correct classification, where columns are the labels returned by the classifiers. The values shown represent the percentages of 30-second segments classified.

From these plots, it can be seen that the E/D classifier exceeds the accuracy of the MFCC classifier for all acoustic scene classes except BusyStreet and Beach. The differences in accuracy between the two classifiers for the BusyStreet class is small. For Beach, however, the MFCC classifier classifies 36% of the samples correctly, whilst the E/D classifier classifies only 8% correctly. In fact, the Beach class is the source of the majority of the incorrect classifications from the E/D classifier. If the Beach class is excluded, the overall classifier accuracy increases from 69% to 78%.



Figure 3: Confusion matrix of classifier trained using Elevation and Diffuseness features. Figures indicate classification percentages across all folds.

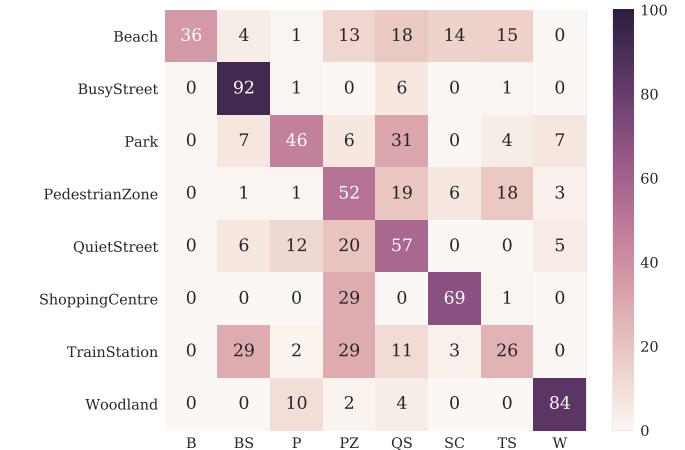


Figure 4: Confusion matrix of classifier trained using MFCC features. Figures indicate classification percentages across all folds.

ifications from the E/D classifier. If the Beach class is excluded, the overall classifier accuracy increases from 69% to 78%.

This poor performance is perhaps due to the fact that in a seafront beach acoustic environment the dominant source of sound is wave motion from the sea. This will appear to DirAC analysis as a large spread of broadband noise. Looking at Figure 3, it can be seen that the E/D classifier mislabels most of the Beach clips as either BusyStreet or QuietStreet. Since one of the dominant sounds of street scenes is the broadband noise from passing cars, it is conceivable that the spatial features extracted from street environment recordings could mirror those from a beach.

Further interesting observations may be made by comparing the specific accuracies and differences in scene confusion between the two classifiers. PedestrianZone, for instance, is classified with only 52% accuracy by the MFCC classifier, with many samples classified as either TrainStation or QuietStreet, whilst the E/D classifier is 97% accurate for this class. This suggests that the spatial information present in the acoustic scene of a pedestrian zone is more

unique to that scene than the spectral information, which evidently can be quite similar to that of a train station or quiet street.

It is also interesting to compare the two classifiers where there is significant confusion in both for a certain class, as often the confusion does not correspond. With ShoppingCentre, for instance, the majority of the confusion in results in the MFCC classifier is with PedestrianZone, perhaps owing to the prominent human sound (speech and footsteps) common to both locations. The E/D classifier, on the other hand, does not confuse ShoppingCentre with PedestrianZone at all, instead confusing it with TrainStation. This could be due to the nature of the recorded train stations and shopping centres as large reverberant indoor spaces, which could potentially influence the calculated values for elevation and particularly diffuseness.

4. DISCUSSION

The results of this experiment show that it is possible to classify acoustic scenes with reasonable accuracy using information on the spatial properties of the scenes as features with a basic GMM classifier. This is an important initial result as it confirms that spatial information could be a valuable feature to utilise in future acoustic scene analysis systems and is worthy of further study.

The results shown in Figure 2 indicate that the addition of MFCC features to DirAC features improves classification accuracy when the individual performance of the two feature sets is similar, but is a hinderance when the individual DirAC feature performance is better than the MFCCs alone. There is some indication of an inverse relationship between the MFCC performance and the performance of the DirAC and E/D classifiers, with the spatial classifiers performing much better when the MFCC performance is worse, though there is not enough data here to establish a trend.

Looking at the classification confusion of the MFCC classifier against the E/D classifier, it seems that in most cases spatial features more uniquely characterise acoustic scenes than spectral features. The differences in specific scene confusions between the two classifiers indicates that *spatial* similarity and *spectral* similarity between scenes are not necessarily the same.

5. CONCLUSIONS

This paper has presented a new system for the classification of acoustic scenes using spatial features extracted with Directional Audio Coding techniques. An extensive new dataset of Ambisonic acoustic scene recordings - the EigenScape dataset - was created for this research and introduced here. DirAC features extracted from EigenScape were used to train GMM classifiers and the accuracy of these classifiers was tested against classifiers trained using standard MFCC features. The DirAC-trained classifiers were shown to have comparable classification accuracy to the MFCC-trained classifiers and a subset of the DirAC features excluding azimuth estimates was shown to substantially outperform the MFCCs by over 10% on average.

Comparison of confusion matrices for the outputs of MFCC and DirAC-trained classifiers reveal many differences in specific scene confusions between the two. This indicates that acoustic scenes that have similar spatial features might not necessarily also have similar spectral features.

6. REFERENCES

- [1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, October 2015.
- [2] D. Wang and G. J. Brown, *Computation Auditory Scene Analysis: Principles, Algorithms and Applications*. Wiley, 2006.
- [3] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, May 2015.
- [4] European Environmental Agency, "Good practice guide on noise exposure and potential health effects," European Union, Tech. Rep., 2010.
- [5] A. L. Brown, "Soundscapes and environmental noise management," *Noise Control Engineering Journal*, vol. 58, no. 5, pp. 493 – 500, 2010.
- [6] 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," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, 2007.
- [7] N. I. Joachim Thiemann and E. Vincent, "The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings," in *Proceedings of Meetings on Acoustics*, vol. 19, June 2013.
- [8] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference (EUSIPCO)*, August 2016.
- [9] mh Acoustics, *em32 Eigenmike® microphone array release notes*, mh acoustics, 25 Summit Ave, Summit, NJ 07901, April 2013. [Online]. Available: <https://mhacoustics.com/sites/default/files/EigenmikeReleaseNotesV15.pdf>
- [10] B. McFee, C. Raffel, D. Liang, D. P. W. 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 (SciPy 2015)*, 2015.
- [11] V. Pulkki, "Spatial sound reproduction with directional audio coding," *Journal of the Audio Engineering Society*, vol. 55, no. 6, pp. 503–516, June 2007.
- [12] V. Pulkki, M.-V. Laitinen, J. Vilkamo, J. Ahonen, T. Lokki, and T. Pihlajamäki, "Directional audio coding - perception-based reproduction of spatial sound," in *International Workshop on the Principle and Applications of Spatial Hearing*, 2009.
- [13] S. Marsland, *Machine Learning: An Algorithmic Perspective*. CRC Press, 2015.
- [14] D. Stowell. (2012) simple-minded audio classifier in python (using mfcc and gmm). [Online]. Available: <https://github.com/danstowell/smacpy>
- [15] M. Lagrange and G. Lafay, "The bag-of-frames approach: A not so sufficient model for urban soundscapes," *Journal of the Acoustical Society of America*, vol. 128, no. 5, November 2015.

CONVOLUTIONAL NEURAL NETWORKS WITH BINAURAL REPRESENTATIONS AND BACKGROUND SUBTRACTION FOR ACOUSTIC SCENE CLASSIFICATION

Yoonchang Han¹, Jeongsoo Park^{1,2} Kyogu Lee²

¹ Cochlear.ai, Seoul, Korea

² Music and Audio Research Group, Seoul National University, Seoul, Korea
{ychan, jspark}@cochlear.ai, kglee@snu.ac.kr

ABSTRACT

In this paper, we demonstrate how we applied convolutional neural network for DCASE 2017 task 1, acoustic scene classification. We propose a variety of preprocessing methods that emphasise different acoustic characteristics such as binaural representations, harmonic-percussive source separation, and background subtraction. We also present a network structure designed for paired input to make the most of the spatial information contained in the stereo. The experimental results show that the proposed network structures and the preprocessing methods effectively learn acoustic characteristics from the audio recordings, and their ensemble model significantly reduces the error rate further, exhibiting an accuracy of 0.917 for 4-fold cross-validation on the development. The proposed system achieved second place in DCASE 2017 task 1 with an accuracy of 0.804 on the evaluation set.

Index Terms— DCASE 2017, acoustic scene classification, convolutional neural network, binaural representations, harmonic-percussive source separation, background subtraction

1. INTRODUCTION

Sounds contain a variety of information that humans use to understand the surroundings, and our behaviours and thoughts are heavily based on this auditory information along with information gathered from different sensory registers. Even if visual information is not given, humans can easily recognise the scene from the surrounding sounds because our expectations are well trained from experience. For instance, we know that bird chirping sound is likely recorded in the park, and cutlery sound is recorded in the restaurant. In addition, it is also possible to guess the size of the space from the sound, because cave-like environment such as metro station produce a lot of reverberations while outdoor scenes do not. However, creating an automated system that understands acoustic scenes is difficult, because it is a fairly high level of information.

Although acoustic scene classification (ASC) is one of the main objectives of machine listening research [1], the research community has lacked benchmark dataset so far [2]. Arguably, Detection and Classification of Acoustic Scenes and Events (DCASE) challenge organised by IEEE Audio and Acoustic Signal Processing (AASP) Technical Committee is one of the first large-scale challenges for ASC research. A number of novel approaches have been proposed in DCASE 2013 [3] and DCASE 2016 [4], and performances of submitted systems are evaluated under the same experimental conditions. In DCASE 2013, most of the submissions are based on hand-made acoustic features along with classifier such as in [5, 6]. Some techniques that widely used for image processing

such as a histogram of gradients (HOG) [7] and recurrence quantification analysis (RQA) [8] features also achieved top places. There was also an approach that utilises deep learning such as [9] using restricted Boltzmann machine, but it showed moderate classification accuracy, presumably due to small amounts of data.

DCASE 2016 task 1 is essentially an extended version of the previous DCASE 2013 ASC task, providing a larger amount of data for an increased number of scenes. Many of participants applied a deep learning approach such as a convolutional neural network (ConvNet) [10, 11, 12] and recurrent neural network (RNN) [13, 14]. Although deep learning approach has been successful, top ranks were achieved by i-Vector [15] and non-negative matrix factorization (NMF) [16], which are rather conventional dictionary learning methods. Also, about half of submitted algorithms in this challenge used mel-frequency cepstral coefficients (MFCCs), one of the most popular hand-made features. As can be seen from the results of the DCASE task in the past, the deep learning approach has shown promising results but clearly no better than the existing methods.

Deep learning technology is rapidly evolving everyday. Although DCASE 2017 [17] provides an increased amount of audio data compare to 2013, it is still not sufficient to take full advantage of the potential of deep learning approach. However, we believe that finding an appropriate way to utilise deep learning is one of the most important research topics in the audio processing field at the moment. This paper demonstrates our approach on ASC task using ConvNets and propose various audio domain specific preprocessing methods that emphasise the different aspects of the acoustic scene. The following sections describe the details of the proposed system and the experimental results and conclusions.

2. SYSTEM ARCHITECTURE

This section introduces the proposed audio preprocessing methods. It also describes the details of the proposed ConvNet architecture and how we have configured the ensemble model from them.

2.1. Audio Preprocessing

In general, we used a full 44.1 kHz without downsampling and amplitude of audio clips was normalised first. Then, we extracted the spectrograms with 128 bin mel-scale following [10] which is a sufficient size to keep spectral characteristics while greatly reduce feature dimensions. The window size for short-time Fourier transform was 2,048 samples (46 ms) with a hop size of 1,024 samples (23 ms). The resulting mel-spectrogram was converted into logarithmic scale, and standardised by subtracting the mean value and dividing

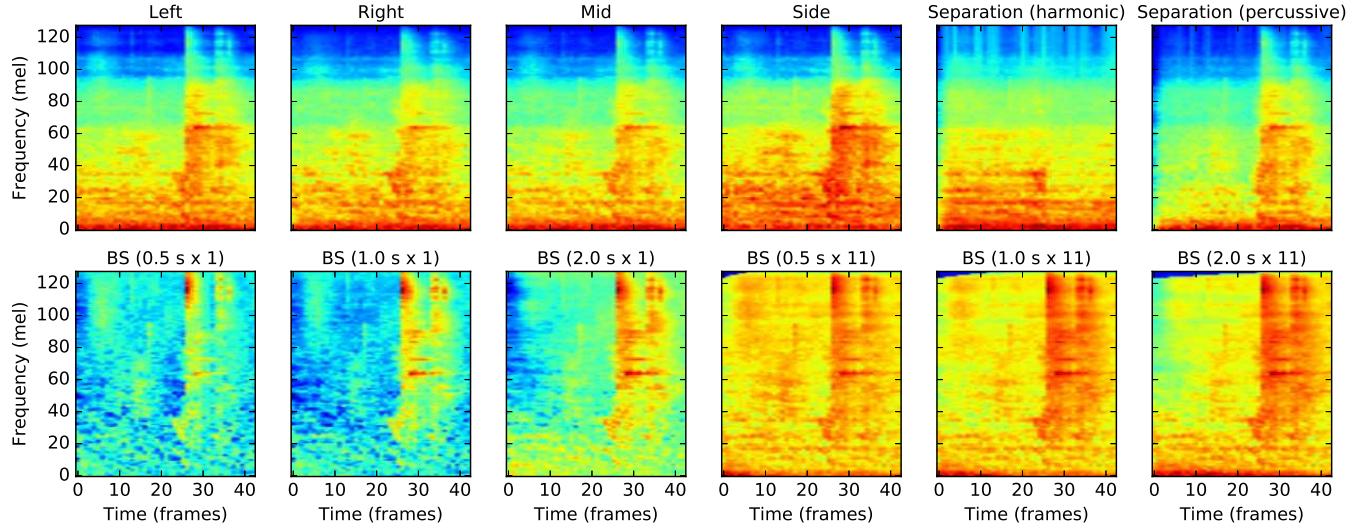


Figure 1: Extracted mel-spectrogram examples of proposed preprocessing methods applied to an audio clip for “café/restaurant” scene. “BS” is background subtraction method, and the numbers in the brackets are the median filtering kernel sizes for time and frequency axes.

by the standard deviation. Standardisation is performed feature-wise and parameters are obtained only from training data to scale both of training and testing data. Finally, we split 10 s audio clip into 1 s audio chunks without overlap for both of training and testing. We used multiple versions of mel-spectrogram which can be largely divided into three methods which are binaural representations, source separation, and background subtraction (BS). A detailed explanation of each method is presented below, and examples of extracted mel-spectrograms are illustrated in Fig.1.

2.1.1. Binaural representations

Although it is common to record audios in stereo, it is usual to make it monaural first by averaging signals prior to processing, as in our previous work [10]. However, we decided to use left-right (LR) and mid-side (MS) pairs in this work, because these contain richer spatial information than mono. For instance, if a car passes in front of a microphone, the sound moves from L to R or R to L, while it is just amplitude change in mono. In addition, the MS representation emphasises the time difference between the sounds reaching each side of the stereo microphone. Use of binaural information have shown superior results in the previous DCASE challenge as in [15] as well. The Mid channel is defined as $L + R$ and the side channel is defined as $L - R$ which is a difference between two channels. For LR and MS, we used 2-conv. model for the analysis explained in the Section 2.2.

2.1.2. Harmonic-percussive source separation

Sound can be generally be divided into two types: harmonic and percussive. In conventional research efforts, harmonic-percussive sound separation (HPSS) algorithms were presented in the context of music signal processing aimed to separate drum sounds from the mixture as in [18]. Here, we separated the audio clips in the dataset into two using the NMF-based HPSS algorithm [19] which enables to separately exploit harmonic and percussive aspects of a sound. Prior to the separation, the stereo sounds are converted to mono.

The experimental parameters used for the separation are 0.7, 1.05, 1.05, and 0.95 for α , β , γ , and δ , respectively, and frame size and hop size are 4,096 and 1,024 samples, respectively. The total number of bases is set as 200, consisting of 100 flat-initialised percussive bases and 100 randomly-initialized harmonic bases. Wiener filtering was not used for the post-processing of NMF, however, we have made the last 30 iterations out of 100 total iterations not to include prior imposition to reduce any artifacts that may be generated in the separation process.

2.1.3. Background subtraction

Typically, median filtering is used for removal of noise in scanned images. Moore Jr. and Jorgenson [20] used this technique for object extraction by subtracting median filtered data from original data. Although this technique is more commonly used in the image processing fields, we think that it can be useful to eliminate the “steady” noise from the environment or recording devices. By doing so, we expect the spectral characteristics of acoustic events in the mel-spectrogram to be emphasised and to be more robust against overfitting. Similar to object detection technology, we applied median filtering on the mel-spectrogram and subtracted it from the original version. We first converted stereo audio into mono prior to the process. The filter sizes used for median filtering are 21, 43, 87 for the time axis (approximately 0.5 s, 1.0 s, and 2.0 s), and 1, 11 for the frequency axis, which are chosen empirically by the experiment. Note that using a kernel size of 1 for on the frequency axis is virtually 1-D median filtering over time. As shown in the bottom row of Fig.1, the background subtraction process emphasizes different spectral characteristics from neighboring regions, which makes it easier to detect acoustic events.

2.2. Network Architecture

We used ConvNet consisting of 8 convolution layers using 3×3 receptive fields inspired by VGGNet [21]. In recent years, it has become common to use extremely deep (100 <) network and residual

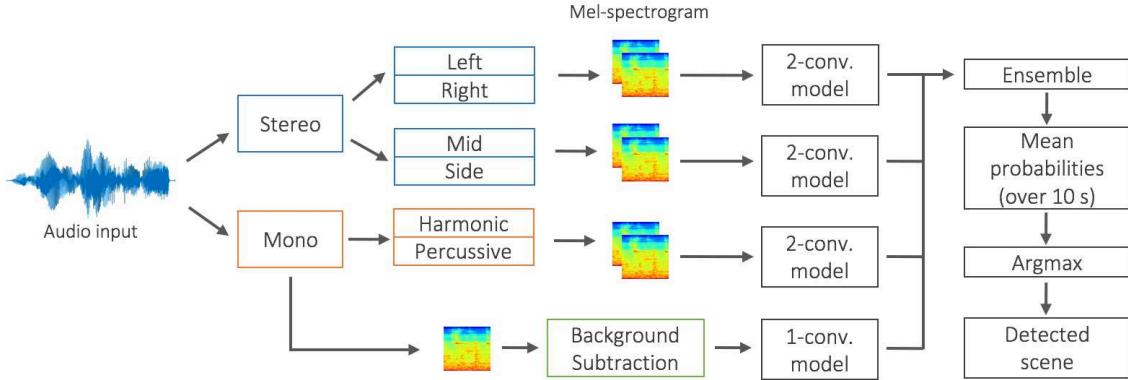


Figure 2: The overall architecture of the proposed system. Multiple ConvNet models are individually trained using a various preprocessing methods and combined into an ensemble model. It then calculates the average probabilities for entire audio clip to detect the scene.

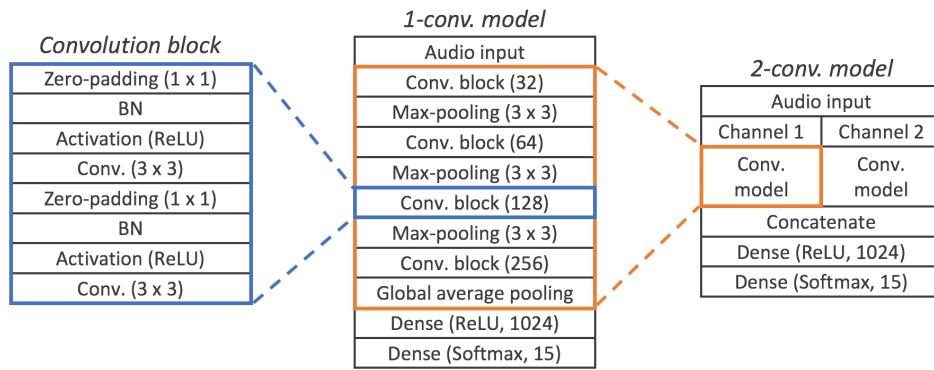


Figure 3: Details of the proposed convolution block, 1-conv. model, and 2-conv. model. The numbers in the brackets are the kernel size for padding/convolution/pooling layers, the number of filters for convolution blocks, and the number of hidden units for dense layer.

connections such as in [22, 23] in the computer vision field. However, we found that it is not highly effective to increase the number of layers or to use a residual connection, at least in our framework, likely due to insufficient amount of data to extract full advantage out of it. The overall architecture of the proposed system is illustrated in Fig.2 which uses two different network architectures: 1-conv. model and 2-conv. model. The former is used for single mel-spectrogram input such as BS, and the latter was used for paired input such as LR, MS, and HPSS. 2-conv. model is similar to 1-conv. model, but processes two channels individually and concatenated before the last fully-connected layer. For both models, we used the same convolution block as illustrated in Fig.3. We employed batch normalization (BN) [24] and rectified linear unit (ReLU) which are de facto standard for modern ConvNets. However, BN and activation function are located before the convolution layer in our proposed network, unlike other networks, because we can get a steady improvement in accuracy. This kind of pre-activation concept can be found in recent residual network papers [22, 23]. We consider that the improvement mainly comes from BN applied for the input data and after max-pooling layers, prior to convolution process.

2.3. Network Ensemble

The results generated by using the same network may be slightly different and model ensemble can generalize this problem [25].

Therefore, we repeated all experiment 3 times using the validation set extracted with different random seeds for each model and took an average probability for each class. In the final decision process, we used two strategies the mean ensemble and ensemble selection method proposed by Caruana et al. [26]. Ensemble selection algorithm aims to find the optimal combination weights by iteratively adding models that maximize the performance of the combination set. We used the mean of test accuracy of all folds as a target value and initialized the ensemble model using LR, MS, and HPSS prior to adding other models. We used 200 iteration and optimal weight we found was 36, 25, 21, 23, 29, 33, 17, 12, and 7 for LR, MS, HPSS, and BS following the order listed in Table1, respectively. Note that this ensemble selection makes use of label from test fold of cross-validation as a hill-climbing set, thus it should not be directly compared to other results of cross-validation.

3. EXPERIMENTS

3.1. DCASE 2017 ASC Dataset

The DCASE 2017 task 1 includes 15 scenes which are bus, café/restaurant, car, city center, forest path, grocery store, home, lakeside beach, library, metro station, office, residential area, train, tram, and urban park. A total 312 segments (52 minutes of audio), recorded at 44.1 kHz with 24-bit resolution in stereo, were provided

| Algorithms | Mean Acc. | Algorithms | Mean Acc. |
|--------------|-----------|----------------|-----------|
| Baseline | 0.748 | BS (2.0 s, 1) | 0.816 |
| Mono | 0.844 | BS (0.5 s, 11) | 0.861 |
| LR | 0.871 | BS (1.0 s, 11) | 0.856 |
| MS | 0.879 | BS (2.0 s, 11) | 0.843 |
| HPSS | 0.869 | Mean ensemble | 0.917 |
| BS (0.5s, 1) | 0.801 | Ensemble sel.* | 0.919 |
| BS (1.0s, 1) | 0.805 | | |

Table 1: Mean accuracy for 4-fold cross-validation using proposed ConvNet with various preprocessing and ensemble methods. Baseline and mono are not used for ensemble models, but illustrated for comparison purpose. Note that the result with * used test label, hence it should not be directly compared to other results.

per scene and the length of the audio segments were 10 seconds. The dataset size is increased compare to 2016, but the length of each audio segment was shortened to 10 s from 30 s, so each audio clip contains less information.

3.2. Experiment Settings

The experiment was carried out using 4-fold cross-validation setting provided by the organizer. Network training was performed by optimizing the categorical cross-entropy and stochastic gradient descent (SGD) with Nesterov momentum [27]. The learning rate, decay, and mini-batch size were set to 0.02, 0.0001, and 128, respectively. We trained the network with the NVIDIA GTX 970 and the experiment took about 2 h per model. We used a randomly selected 15% of the training data for validation and the network training was early-stopped if the validation loss did not decrease by more than 20 epochs. The number of examples for training was about 29,800. Baseline system provided by the organizer used mel-spectrogram with 40 mel with a frame size of 40 ms as an input feature for 2 layers x 50 hidden units multilayer perceptron (MLP).

4. RESULTS

4.1. Cross-validation Results

Accuracy is used as a performance metric and the 4-fold mean accuracy of each preprocessing method and ensemble models are presented in Table 1. As a result, the accuracy of the 2.conv-models was 0.87, and BS with various settings (1-conv. models) was generally not as good as 2-conv. models. By combining the results from all the models, it was possible to improve the mean accuracy to 0.917, and ensemble selection slightly pushed it up to 0.919. Because of page limitations, we could not present all class-specific results. However, BS results showed quite different confusion between classes, depending on median filtering size, which is the main reason for the performance improvement of the ensemble. For instance, although the result of BS (0.5 s, 1) are relatively poor compared to other methods, it showed about 16% higher accuracy than the LR for “bus” scene. The confusion matrix of ensemble selection model result is presented in Fig. 4, and it can be observed that the confusion is relatively focused in the home and office, park, and residential area.

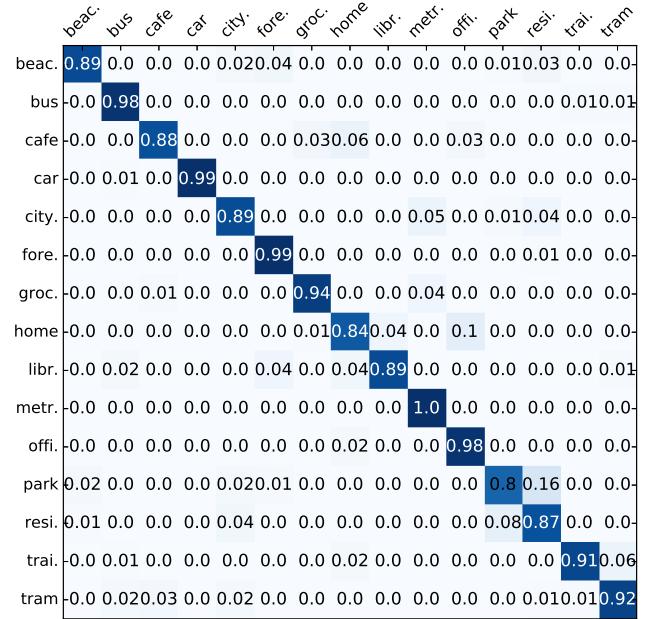


Figure 4: 4-fold mean confusion matrix of the proposed ConvNet with ensemble selection. X-axis indicates the predicted label and Y-axis indicates the true label.

4.2. DCASE 2017 Submission

We used the same experiment settings from development set for the evaluation set. For the final submission, we submitted four slightly different results following the challenge rule. We used a mean probability of 4-fold cross-validation models for submission 1 and 2, and used a newly trained model using a full development set for submission 3 and 4. Regarding ensemble method, we used ensemble selection for submission 1 and 3, and mean ensemble for submission 2 and 4.

4.3. Acknowledgement

This research was supported by Korean government, MSIP provided financial support in the form of Bio&Medical Technology Development Program (2015M3A9D7066980).

5. CONCLUSION

In this paper, we illustrated how we applied ConvNet for identifying the acoustic scene. The main contribution of this paper is presenting a various preprocessing methods that are useful for ConvNet, also having a great synergy when combined together in an ensemble model. As a result, we could obtain an accuracy of 0.917 for 4-fold cross validation on the development set and 0.804 on the evaluation set. In the future, we plan to investigate the optimal kernel size for BS and pre-activation convolution block further, which are currently selected heuristically.

6. REFERENCES

- [1] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-

- IEEE press, 2006.
- [2] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.
 - [3] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.
 - [4] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.
 - [5] D. Li, J. Tam, and D. Toub, "Auditory scene classification using machine learning techniques," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
 - [6] J. T. Geiger, B. Schuller, and G. Rigoll, "Recognising acoustic scenes with large-scale audio feature extraction and svm," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
 - [7] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 1, pp. 142–153, 2015.
 - [8] G. Roma, W. Nogueira, P. Herrera, and R. de Boronat, "Recurrence quantification analysis features for auditory scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, vol. 2, 2013.
 - [9] J. Nam, Z. Hyung, and K. Lee, "Acoustic scene classification using sparse feature learning and selective max-pooling by event detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
 - [10] Y. Han and K. Lee, "Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [11] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [12] J. Kim and K. Lee, "Empirical study on ensemble method of deep neural networks for acoustic scene classification," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [13] E. Marchi, D. Tonelli, X. Xu, F. Rinneval, J. Deng, and B. Schuller, "The up system for the 2016 DCASE challenge using deep recurrent neural network and multiscale kernel subspace learning," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [14] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [15] 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," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [16] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Supervised nonnegative matrix factorization for acoustic scene classification," DCASE2016 Challenge, Tech. Rep., September 2016.
 - [17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system."
 - [18] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in *Signal Processing Conference, 2008 16th European*. IEEE, 2008, pp. 1–4.
 - [19] J. Park, J. Shin, and K. Lee, "Exploiting continuity/discontinuity of basis vectors in spectrogram decomposition for harmonic-percussive sound separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 5, pp. 1061–1074, 2017.
 - [20] A. W. Moore Jr and J. W. Jorgenson, "Median filtering for removal of low-frequency background drift," *Analytical chemistry*, vol. 65, no. 2, pp. 188–191, 1993.
 - [21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
 - [22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.
 - [23] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.
 - [24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
 - [25] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in neural information processing systems*, 1995, pp. 231–238.
 - [26] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 18.
 - [27] Y. Nesterov *et al.*, "Gradient methods for minimizing composite objective function," 2007.

AUDIO EVENT DETECTION USING MULTIPLE-INPUT CONVOLUTIONAL NEURAL NETWORK

Il-Young Jeong^{1,2}, Subin Lee^{1,2}, Yoonchang Han², Kyogu Lee¹

¹ Music and Audio Research Group, Seoul National University, Korea,

² Cochlear.ai, Seoul, Korea

{iyjeong, sblee, ychan}@cochlear.ai, kglee@snu.ac.kr

ABSTRACT

This paper describes the model and training framework from our submission for DCASE 2017 task 3: *sound event detection in real life audio*. Extending the basic convolutional neural network architecture, we use both short- and long-term audio signal simultaneously as input data. In the training stage, we calculated validation errors more frequently than one epoch with adaptive thresholds. We also used class-wise early-stopping strategy to find the best model for each class. The proposed model showed meaningful improvements in cross-validation experiments compared to the baseline system.

Index Terms— DCASE 2017, Sound event detection, Convolutional neural networks

1. INTRODUCTION

Sound event detection (SED) is a research field that aims to detect and identify events in audio signals. In addition to playing a significant role in understanding the audio of real-life sensing, it also has a wide range of applications such as automatic driving, surveillance systems, health care, and humanoid robots.

Detection and classification of acoustic scenes and events (DCASE) 2017 is a challenge for a variety of audio recognition tasks, and task 3: *sound event detection in real life audio* focuses explicitly on SED [1]. The organizer chose six event classes related to human presence and transportation, and participants were asked to develop algorithms that automatically detect these events from real-world recordings. It also has various sub-objectives which have been dealt with conventional machine learning issues such as multi-label classification, data imbalance, insufficient data, and unreliable annotation problems.

There was a similar task about SED in DCASE 2016, and some participants had tried various approaches [2]. Most of the submitted models used deep neural networks (including recurrent neural networks [3] and convolutional neural networks (ConvNet) [4]), Gaussian mixture model [5], or random forests [6]. Regarding feature extraction method, mel-spectrogram [3, 4] or mel-frequency cepstral coefficients [5, 6] were most frequently used methods. Although it is not easy to determine the optimal approach for DCASE 2017 from these results, the ‘deep learning’ approaches seem to have better performance than traditional approaches.

From the past approaches, our proposed model also follows deep learning approach, especially the ConvNet architecture. We also designed several techniques to overcome the existing limits and maximize detection performance by taking two type input information (short, long-term) and optimizing the learning process.

The rest of this paper is organized as follows. Section 2 provides a brief description of the data sets used in DCASE 2017 and how we parsed it. Section 3 describes the proposed model architecture, and Section 4 explains how we trained the model, including short- and long-term analysis, optimization strategy, and class-wise early-stopping. Experimental results are presented in Section 5. Then directions for future work are discussed in Section 6, followed by a conclusion in Section 7.

2. DATASET AND PARSING

2.1. TUT dataset

TUT sound events 2017 dataset was provided in task 3 of DCASE 2017 [7]. It consists of 24 stereo audio recordings with 3-5 minutes length and 44.1kHz sampling rate. Each recording has corresponding annotations for six different sound event classes: breaks squeaking, car, children, large vehicle, people speaking, and people walking. Each annotation consists of an onset time, on offset time, and an event class.

We parsed both audio recordings and annotations in a matrix format. For each audio recording, it is represented as $\hat{x} \in \mathbb{R}^{H \times N}$, where H and N denote the number of channels (2 for stereo) and samples, respectively. On the other hand, the annotations are in the form of $\hat{t} = [\hat{t}_1, \dots, \hat{t}_C]^T$ and $\hat{t}_c \in \mathbb{R}^N$, where C denotes the number of classes (6 for this task). Here, $\hat{t}_c(n)$ is 1 if c -th event is active in n -th sample, and 0 if inactive.

2.2. Target setting

Our model was designed to detect events with 1 s time resolution, that is, when an event occurs in ± 0.5 s range from the present point, the expected output will be ‘active’ even if it does not in the target point. In order to consider this in the training process, we set the target output $t_c(n)$ as follows:

$$t_c(n) = \max\{\hat{t}_c(n - 22050), \dots, \hat{t}_c(n + 22050)\} \quad (1)$$

where $\hat{t}_c(n)$ is the binary annotation of c -th event in the n -th sample. It is noted that $\pm 22,050$ sample range is about ± 0.5 s time range in the 44.1kHz sample rate.

2.3. Input data setting

To obtain the sufficient information from audio recordings, our model takes two inputs with different time length as Fig. 1. For the first one, which we call the short-term data, we took the samples in the range of $\pm 88,573$ samples from the present point thus the size of

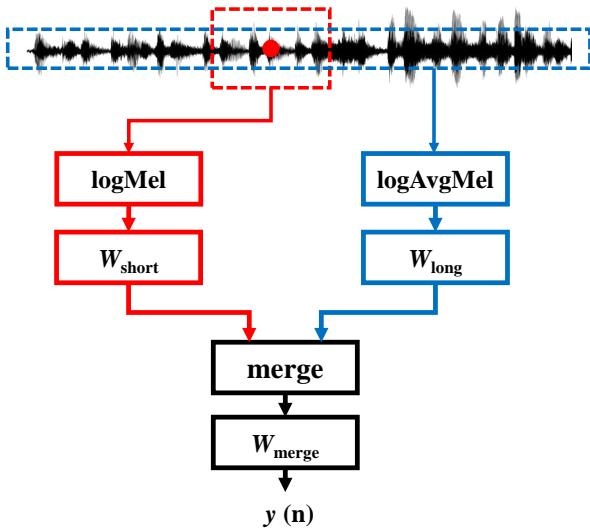


Figure 1: Framework for detecting events from the red dot point using short- and long-term data. W s denote the respective concatenations of computational layers, including convolution, pooling, and fully-connected layer.

a short-term signal is $X_{short} \in \mathbb{R}^{2 \times 177147}$, which is approximately 4 s stereo. We set this number of samples to be 3^m with integer m to be suitable for ConvNet architecture with 3 samples pooling. Another input is the long-term data, which is the entire audio recording with 3-5 minutes. As described in Section 3, each input data is first analyzed individually then merged for the deeper analysis.

3. MODEL

3.1. Mel-spectrogram

For the short-term data, we first extracted the logarithm of mel-spectrogram (logMel) as the first layer. 40 mel bins were extracted for each channel from 1,024 samples with a 729 sample shift. Here, $729 = 3^6$ shift was chosen to be suitable for ConvNet architecture with 3 sample pooling. This mel-spectrogram is followed by a logarithm operation with small offset as

$$x = \log(x + 10^{-5}). \quad (2)$$

The output of logMel layer has the size of $80 \text{ (mel} \times \text{channel}) \times 243$ (frame). Instead of preparing it for every offset samples in the training data, we computed it in real-time using *kapre* [8].

A similar approach was used for long-term data. A mel-spectrogram is extracted from the whole audio file in mono, and averaged over the time frame dimension, and taken a logarithm operation as (2) (logAvgMel). The output has the size of 40×1 . Instead of using *kapre*, we extracted it for every recording before training to avoid redundant computations in every batch iterations.

3.2. ConvNet architecture

After the logMel or logAvgMel layer, we used layers that are widely used in ConvNet-related models: convolution, pooling, dropout, and fully-connected layer. Details are described in Table 1.

Table 1: Detailed model description. conv: convolution layer, pool: max-pooling layer, repeat: layer repetition over frame dimension, add: layer adding, fc: fully-connected layer. All conv and fc layer is followed by ReLU activation, except the last fc layer, which has sigmoid activation.

| layer (short-term) | output size (filter \times frame) | layer long-term | output size (filter \times frame) |
|-----------------------|--|--------------------|--|
| logMel | 80×243 | logAvgMel | 40×1 |
| conv | 64×243 | conv | 64×1 |
| pool | 64×81 | conv | 64×1 |
| conv | 64×81 | repeat | 64×27 |
| pool | 64×27 | | |
| | | ↙ | ↙ |
| layer merged | output size (filter \times frame) | | |
| add | 64×27 | | |
| conv | 64×27 | | |
| pool | 64×9 | | |
| conv | 64×9 | | |
| pool | 64×3 | | |
| fc | 64×1 | | |
| dropout | 64×1 | | |
| fc | 6×1 | | |

We used the 1-dimensional convolution layer (conv) with 64 filters for both logMel and logAvgMel inputs. In case of logMel, we set kernel size of 3 and stride to 1, while these parameters are not required for logAvgMel since it has the length of 1. Rectified linear unit (ReLU) is used for the non-linearity of conv. On the other hand, the max-pooling layers (pool) have a pooling size of 3, and the dropout layer has a probability of 0.5. Our model also used two fully-connected layers (fc). The first fc layer has 64 filters with ReLU activation, and the second one has 6 filters and sigmoid activation to indicate event activity possibilities.

Features from short- and long-term data are merged after two convolution layers. Because of the different output sizes, we used a repeat layer for logAvgMel which reproduces the inputs to have the specific size. Several merging methods were tried, and we empirically found that taking a sum of two layers (add) performs the best detection performance.

4. LEARNING FRAMEWORKS

4.1. Optimization

We used an Adam optimizer [9] with 8 mini-batch size. For every batch generation, we randomly picked a random audio recording and an offset, and took $\pm 88,573$ stereo samples from the offset. As data augmentation, we shuffled the left and right channel randomly for every batch generation. The pre-computed logAvgMel of the selected recording is also taken. Optimization procedure is done with *keras* [10].

4.2. Validation and adaptive threshold

Since our model uses sample-level batches, one epoch means that all the sample in training data is used as the offset. In our works, we

Table 2: Results of 4-fold cross-validation. Each metric denotes error rate (ER) and F-score (F), respectively. ‘average’ denotes an arithmetic mean of 4-fold results. ‘total’ denotes a micro-averaging over all classes and it might be different with an average of class-wise results due to the different class distribution and substitutions (in case of ER). Also, it is noted that the results in CV may have some overfitting due to the adaptive thresholding.

| fold | 1 | | 2 | | 3 | | 4 | | average | | baseline | |
|------------------|------|------|------|------|------|------|------|------|---------|------|----------|------|
| | ER | F | ER | F | ER | F | ER | F | ER | F | ER | F |
| brakes squeaking | 1.00 | 0.0 | 1.00 | 0.0 | 0.97 | 11.8 | 0.93 | 32.4 | 0.98 | 11.1 | | |
| car | 0.66 | 63.8 | 0.45 | 80.0 | 0.37 | 79.4 | 0.54 | 69.9 | 0.51 | 73.3 | | |
| children | 1.00 | 0.0 | 1.00 | 0.0 | 0.99 | 2.7 | 0.25 | 85.7 | 0.81 | 22.1 | | |
| large vehicle | 0.78 | 56.8 | 0.71 | 65.7 | 0.27 | 85.4 | 0.65 | 61.0 | 0.60 | 67.2 | | |
| people speaking | 0.91 | 28.7 | 0.97 | 16.8 | 0.81 | 32.9 | 0.54 | 72.4 | 0.80 | 37.7 | | |
| people walking | 0.92 | 19.3 | 0.25 | 86.0 | 0.46 | 76.8 | 0.53 | 68.6 | 0.54 | 62.7 | | |
| total | 0.72 | 48.7 | 0.43 | 75.6 | 0.42 | 73.3 | 0.46 | 70.2 | 0.51 | 67.0 | 0.69 | 56.7 |

found that one epoch for validation might be too long. Therefore, we validated the model after every fixed number of mini-batch iteration which is much shorter than an epoch. In our works, we checked in every 20 mini-batches learning, and the model was trained until 1,000 iterations at most.

Also, we empirically found that the default threshold of 0.5 is not always the optimal setting. Two reasons can be given for this. First, because the model is validated before an epoch is completed, this portion of the data (20 mini-batches) would not have the same class distribution as the entire data set. Second, when the class imbalance is severe (e.g., 1% active and 99% inactive), then we found that the predicted results tend to have more imbalanced distribution (0% active and 100% inactive). To avoid this problem, the optimal threshold was searched from 0 to 1 at 0.001 resolution for each class in every validation stage. Although these selected thresholds could cause the overfitting for the validation dataset, we believe that it could be ignored if the class distribution in the validation and test dataset is same or similar, especially when compared to the problems as mentioned above caused by not using it.

4.3. Class-wise early-stopping

Because each event class is individually detected using the sigmoid activation in this model, it can be thought as a multi-tasked learning that consists of a single event class detection model, but simultaneously learned. In this case, each class model would be trained at different speeds, and some classes converge too much while others do not. To solve this problem, we used the class-wise early-stopping strategies for each class by calculating the class-specific validation errors. In the test phase, each class event is detected using its respective model.

5. EXPERIMENTS AND DCASE 2017 SUBMISSION

5.1. Experimental results

We evaluated our model following the settings and the metrics of DCASE 2017 [11]. It first counts the number of false negatives (FN), false positives (FP), and true positives (TP) in the prediction for every 1 s of data, and calculate substitutions (S), insertions (I), and deletions (D) as follows:

$$\begin{aligned} S(k) &= \min(FN(k), FP(k)), \\ D(k) &= \max(0, FN(k) - FP(k)), \\ I(k) &= \max(0, FP(k) - FN(k)), \end{aligned} \quad (3)$$

where k denotes the index of the 1 s segment. One of the evaluation criteria, error rate (ER), is calculated as follows.

$$ER = \frac{\sum_k S(k) + \sum_k D(k) + \sum_k I(k)}{\sum_k N(k)}, \quad (4)$$

where $N(k)$ denotes the number of active events in the k -th segment. F-score, on the other hand, is used as another criterion by calculating as follows:

$$Fscore = \frac{2 \sum_k TP(k)}{2 \sum_k TP(k) + \sum_k FP(k) + \sum_k FN(k)}, \quad (5)$$

We evaluate our model by using 4-fold cross validation provided by organizer.

Table 2 shows the experimental results for each metric and fold. At first, it indicates that the proposed model shows the better error rate and F-score in all the folds and the metrics. However, detailed results are different for each fold and class. We expect this difference is caused largely due to the two reasons. First, each fold has different class distributions, and some may be more severely imbalanced, which can lead to the lack of training data for specific classes. In case of the difference between class, the different acoustic characteristics of classes also could be another reason. Some classes might have the acoustic nature that can be easily detected or well-suited for the presented model.

5.2. Submission for DCASE 2017

In DCASE 2017, We submitted 4 models based on above frameworks. We empirically applied the following post-processing steps.

- For submission 1, we took the ensemble of 4 folds CV model using majority vote. 50% voting was considered as ‘active’.
- For submission 2, we took the ensemble of 4 folds CV model using majority vote. 50% voting was considered as ‘inactive’.
- For submission 3, we took the ensemble of 3 models (fold 2, 3, and 4) from 4 folds CV using majority vote. We worried that the exceedingly poor performance in fold 1 might mean that this model failed learning.
- For submission 4, we also took the ensemble of 4 fold CV model, but used weighted vote based on those validation error rate.

The ensemble output for c -th class in k -th segment, $\bar{y}_c(k)$, is calculated as

$$\bar{y}_c(k) = \frac{\sum_f (1 - CER_{f,c}) y_{f,c}(k)}{\sum_f (1 - CER_{f,c})} \quad (6)$$

where $CER_{f,c}$ and $y_{f,c}(k)$ denote the class-wise ER of c -th class in f -th fold.

According to DCASE 2017 results [12], above submissions scored ER of 0.9260, 0.8673, 0.8080, and 0.8985, respectively, and F-score of 42.0%, 27.9%, 40.8%, and 43.6%, respectively. In particular, submission 3 ranked third in ER. However, All the submissions showed relatively low F-score, and it is needed to be improved.

6. FUTURE WORK

Although the proposed model achieved meaningful results compared to the baseline system, there still exists room for improvement in future works. First, although we have tried various approaches to handle the class imbalance problems, such as class weights, these were not included in the final submission models except adaptive thresholding because they could not make any improvement. We have a plan to apply other techniques, including data sampling methods [13, 14]. Also, in our experiments, we found that the training and validation loss severely fluctuates over mini-batch iteration. We conjecture that validating model more frequently than one epoch might be one of the reasons, but we are still tried to avoid or reduce this problem. Finally, we have an interest in finding the proper preprocessing method for deep learning of audio. Although log mel-spectrogram what we used is widely used in other studies, it lost a substantial amount of information, and a better approach is still required. We expect that proper deep learning model is capable of replacing these preprocessing step and conducting an end-to-end model which detects events from raw waveform [15].

7. CONCLUSION

This paper has described the model for sound event detection submitted to DCASE 2017: task 3. We presented a convolutional neural networks architecture using two input data, which are short-term and long-term data. Several optimization strategies were also presented, including frequent validation with adaptive thresholds and the class-wise early-stopping. The proposed framework showed meaningful improvements compared to the baseline system.

8. REFERENCES

- [1] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “Dcase 2017 challenge setup: tasks, datasets and baseline system,” in *Proc. the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.
- [2] <http://www.cs.tut.fi/sgn/arg/dcase2016/>.
- [3] S. Adavanne, G. Parascandolo, P. Pertila, T. Heittola, and T. Virtanen, “Sound event detection in multichannel audio using spatial and harmonic features,” in *Proc. the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, September 2016, pp. 6–10.
- [4] A. Gorin, N. Makhazhanov, and N. Shmyrev, “DCASE 2016 sound event detection system based on convolutional neural network,” *DCASE2016 Challenge, Tech. Rep.*, September 2016.
- [5] T. Heittola, A. Mesaros, and T. Virtanen, “DCASE2016 baseline system,” *DCASE2016 Challenge, Tech. Rep.*, September 2016.
- [6] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, “Experiments on the DCASE challenge 2016: Acoustic scene classification and sound event detection in real life recording,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, September 2016, pp. 20–24.
- [7] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.
- [8] K. Choi, D. Joo, and J. Kim, “Kapre: On-gpu audio pre-processing layers for a quick implementation of deep neural network models with keras,” in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.
- [9] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [10] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [11] A. Mesaros, T. Heittola, and T. Virtanen, “Metrics for polyphonic sound event detection,” *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.
- [12] <http://www.cs.tut.fi/sgn/arg/dcase2017/>.
- [13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
- [14] S.-J. Yen and Y.-S. Lee, “Cluster-based under-sampling approaches for imbalanced data distributions,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.
- [15] J. Lee, J. Park, K. L. Kim, and J. Nam, “Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms,” *arXiv preprint arXiv:1703.01789*, 2017.

DCASE 2017 TASK 1: ACOUSTIC SCENE CLASSIFICATION USING SHIFT-INVARIANT KERNELS AND RANDOM FEATURES

Abelino Jiménez, Benjamín Elizalde and Bhiksha Raj

Carnegie Mellon University, Department of Electrical and Computer Engineering, Pittsburgh, PA, USA
abjimenez@cmu.edu, bmartin1@andrew.cmu.edu, bhiksha@cs.cmu.edu

ABSTRACT

Acoustic scene recordings are represented by different types of handcrafted or Neural Network features. These features, typically of thousands of dimensions, are classified in state of the art approaches using kernel machines, such as the Support Vector Machines (SVM). However, the complexity of training these methods increases with the dimensionality of these input features and the size of the dataset. A solution is to map the input features to a randomized low-dimensional feature space. The resulting random features can approximate non-linear kernels with faster linear kernel computation. In this work, we computed a set of 6,553 input features and used them to compute random features to approximate three types of kernels, Gaussian, Laplacian and Cauchy. We compared their performance using an SVM in the context of the DCASE Task 1 - Acoustic Scene Classification. Experiments show that both, input and random features outperformed the DCASE baseline by an absolute 4%. Moreover, the random features reduced the dimensionality of the input by more than three times with minimal loss of performance and by more than six times and still outperformed the baseline. Hence, random features could be employed by state of the art approaches to compute low-storage features and perform faster kernel computations.

Index Terms— Acoustic Scene Classification, Laplacian Kernel, Gaussian Kernel, Cauchy Kernel, Kernel Machines, Random Features, DCASE Challenge

1. INTRODUCTION AND RELATED WORK

The DCASE Task 1 - Acoustic Scene Classification (ASC) aims to identify a recording as belonging to a predefined set of scene-classes that characterizes an environment, for example *park*, *home*, or *office*. Typically, ASC approaches capture the diverse characteristics from the audio signal by computing different types of features, either hand-crafted [1, 2, 3, 4, 5] or derived from Neural Networks [6, 7, 8]. These features are commonly of high-dimensionality (up to ten of thousands) and state of the art ASC approaches classified them using Support Vector Machines, the best known member of kernel methods.

Kernel methods have the kernel trick property, which employs a non-linear kernel function to operate in a high-dimensional space by computing the inner products between the all pairs of transformed input features. The inner products are computed and stored in the Kernel or Gram matrix, which computing time and storage complexity increases in the dimensionality and number of the input features. A solution is to compute random features [9], which have been well studied mainly for shift-invariant kernels because of their closed form. The process maps the input features into a

low-dimensional random space. Then, the resulting random features approximate non-linear kernels with linear kernel computations, hence speeding up the kernel matrix generation.

In this paper, we evaluated our random features in the context the 2017 DCASE Task 1 - Acoustic Scene Classification [10]. First, we computed input features with over six thousand dimensions, then we computed random features to approximate three types of shift-invariant kernels, Gaussian, Laplacian and Cauchy. Both type of features, input and random, were classified using an SVM. Experiments show that the baseline is outperformed by 4% by all features. Moreover, random features reduced their dimensionality by more than three times with minimal loss of performance and by six times and still outperformed the baseline.

The paper is organized as follows: In Section 2 we describe in detail the kernel functions used. In Section 3 we present experiments and results for Task 1. Finally, in Section 4 we conclude discussing the scope of the presented technique as well as future directions.

2. METHODS: SHIFT-INVARIANT KERNELS AND RANDOM FEATURES

In this section we describe the computation of random features for three types of shift-invariant kernels in the context of SVM. Acoustic Scene Classification has been explored by state of the art approaches based on kernel methods, which find non-linear decision boundaries using a kernel function. The function takes input features (extracted from the audio) in a space \mathcal{X} and yield output scene classes in \mathcal{Y} . In this paper, we consider $\mathcal{X} = \mathbb{R}^N$ and $\mathcal{Y} = \{1, 2, \dots, C\}$. Moreover, the kernel function can be expressed as $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, which is positive-definite and yields the value corresponding to the inner product between $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$. The function ϕ maps \mathbb{R}^N to some space \mathcal{H} , which is generally of higher dimensionality and has better class separability.

However, computing the kernel function could become a prohibitive task if the dimensionality of the input, N , is large and if the size of the training set n is large. This happens because in order to learn the decision boundary function f from the input audio and the corresponding labels in the dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$, we need to compute the value $K(\mathbf{x}_i, \mathbf{x}_j)$ for every element $i, j \in \{1, \dots, n\}$.

Therefore, our solution for this problem are *random features*, which approximate a kernel function by finding a map Φ from \mathbb{R}^N to a low-dimensional random space \mathbb{R}^M , such that

$$K(\mathbf{x}_1, \mathbf{x}_2) \approx \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle \quad (1)$$

Although different random features mappings have been proposed for different kernel functions [11] [12], we focused on random features for *shift-invariant* kernels. We say that a kernel is shift-invariant if for any $\mathbf{x}_1, \mathbf{x}_2, \mathbf{z} \in \mathbb{R}^N$

$$K(\mathbf{x}_1 + \mathbf{z}, \mathbf{x}_2 + \mathbf{z}) = K(\mathbf{x}_1, \mathbf{x}_2) \quad (2)$$

Which is equivalent to say that, for any \mathbf{x}_1 and \mathbf{x}_2

$$K(\mathbf{x}_1, \mathbf{x}_2) = K(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{0}) \quad (3)$$

Shift-invariant kernels have been proven to admit a closed form of computing random features as stated by the use of the Bochner's theorem [9]. The function to compute random features $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^M$ is given by

$$\Phi(\mathbf{x}) = \sqrt{\frac{2}{M}} \cos(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (4)$$

where \mathbf{W} is a $M \times N$ matrix, \mathbf{b} is a vector with M components and the cos function is element-wise. The randomness comes from the generation of the components of \mathbf{W} and \mathbf{b} , where b_i comes from a uniform distribution between 0 and 2π , and w_{ij} comes from the Fourier Transform of the function $g(\delta) = K(\delta, \mathbf{0})$. Therefore, the approximation stated in equation 1, depends on the kernel function involved and the distribution used to generate the matrix \mathbf{W} .

In this paper, we focus on three well studied shift-invariant kernel functions, Gaussian, Laplacian and Cauchy. Their definition and corresponding distributions used to generate random features are described below.

2.1. Gaussian Kernel and Random Features

The Gaussian kernel, also known as Radial Basis Kernel, is perhaps the most popular after the Linear kernel. The Gaussian function employs the ℓ_2 norm and we define,

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2) \quad (5)$$

To compute the random features, we generate the components of the matrix \mathbf{W} according to a Gaussian distribution as follows,

$$w_{ij} \sim \mathcal{N}(0, 2\gamma)$$

2.2. Laplacian Kernel and Random Features

The Laplacian kernel is similar to the Gaussian, but the main difference is that it employs the ℓ_1 norm, where $\|\mathbf{x}\|_1 = \sum_{i=1}^N |x_i|$. In this work, we consider the Laplacian kernel,

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|_1) \quad (6)$$

To compute the random features, we generate the components of the matrix \mathbf{W} according to a Cauchy distribution as follows,

$$w_{ij} \sim \text{Cauchy}(0, \gamma)$$

2.3. Cauchy Kernel and Random Features

The Cauchy kernel is less known in comparison to the previous two and computing this kernel can be even a more expensive task with high-dimensional vectors due to its mathematical form, hence benefiting more from the speed of processing random features. We define the kernel,

$$K(\mathbf{x}_1, \mathbf{x}_2) = \prod_{i=1}^N \frac{1}{1 + \gamma^2(x_{1i} - x_{2i})^2} \quad (7)$$

To compute the random features, we generate the components of the matrix \mathbf{W} according to a Laplace distribution,

$$w_{ij} \sim \text{Laplace}(0, \gamma)$$

2.4. Training SVMs with Random Features

An SVM is a kernel method that can perform non-linear classification by solving the quadratic optimization of the dual form and taking advantage of the kernel trick [13]. The kernel trick uses a non-linear function to map the input features into a high-dimensional feature space by computing the kernel matrix.

An SVM using a non-linear shift-invariant kernel using the input features could be approximated by a linear SVM using the random features. The kernel matrix resulting from computing the inner product between the random features correspond to an approximation of the kernel matrix using the input features and the shift-invariant kernel. The linear computation has an important implication because there are libraries optimized for these problems.

3. EXPERIMENTAL SETUP AND RESULTS

Our two set of experiments addressed the DCASE Task 1 - Acoustic Scene Classification [10]. We evaluate and compare the performance of the input features using SVMs with three non-linear shift-invariant kernels against the random features corresponding to the three kernel types using linear SVMs. Both pipelines are illustrated in Figure 1.

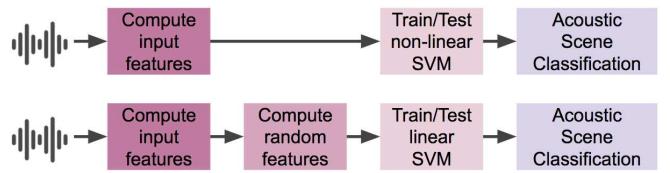


Figure 1: The acoustic scene dataset is used to extract input features for each recording. Then, the input features are used to train the SVM in two different ways. One is to pass the features directly to a non-linear shift-invariant kernel SVM, and the other is to first compute the random features and then pass them to a linear kernel SVM. Lastly, the trained SVM is used for multi-class classification on the test recordings.

3.1. Acoustic Scene Dataset

For our experiments we used the development set of the “DCASE: TUT Acoustic Scenes 2017” dataset [14]. It consists of recordings from various acoustic scenes of 3-5 minutes long divided into 4

cross-validation folds. The original recordings were then split into segments with a length of 10 seconds. Recordings were made using a binaural microphone and a recorder using 44.1 kHz sampling rate and 24 bit resolution. The 15 acoustic scenes are: *Bus, Cafe / Restaurant, Car, City center, Forest path, Grocery store, Home, Lakeside beach, Library, Metro station, Office, Residential area, Train, Tram, Urban park*.

3.2. Compute Input Features

We extracted a large set of audio features proposed in [3], which are later used to compute the random features. The set include different features to capture different information from the acoustic scenes, which consist of multiple sound sources. The set is computed with the open-source feature extraction toolkit openSMILE [15] using the configuration file *emolarge.conf*. The features are divided in four categories: cepstral, spectral, energy related and voicing and are extracted every 10 ms from 25 ms frames. Moreover, included are functionals, such as mean, standard deviation, percentiles and quartiles, linear regression functionals, or local minima/maxima. The total dimensionality of the feature vector is 6,553.

3.3. Input Features and Non-linear SVM

The first set of experiments aimed to evaluate our large set of input features and non-linear SVMs in ASC. We used the input features to train the three types of non-linear shift-invariant SVMs, also, we included the linear kernel (without random features). The SVM parameter C was tuned using a search grid on the linear kernel and was fixed in all cases to $C = 100$ and the performance was measured using accuracy. The accuracy is the average classification accuracy over the 4 validation folds provided for this challenge. Additionally, we explored different values for γ , obtaining the best results with $\gamma = 2^{-18}$ for Gaussian Kernel, $\gamma = 2^{-14}$ for Laplacian Kernel, and $\gamma = 2^{-8}$ for Cauchy Kernel. Before training the models, in each fold we normalized the input features with respect to the training set. We computed the mean and the standard deviation using each feature file and then subtracted the mean and divided by the standard deviation every file in the training and the testing sets.

The classification performance for all kernel types was similar as shown in Table 1. Generally, non-linear kernels tend to perform better than linear kernels for ASC [1]. However, it's not uncommon to have a similar performance if the class separability given by the features is not so complex, which could be our case. Among our best classified scene-classes we have *Bus, Cafe/Restaurant* and *Grocery store* with improvements of up to 25%.

3.4. Random Features and Linear SVM

The second set of experiments aimed to show that the use of random features and linear SVM have a similar performance to the non-linear SVMs. For this, we used the training and testing input features to compute the random features corresponding to each of the three shift-invariant kernels described in Section 2. Then, these random features were used to train the SVM with linear kernel.

The performance of employing the random features indeed compared to the one of the input features with non-linear SVM as shown in Table 2. We can see that the results improve as M , the dimensionality of the random features increases, hence showing minimal loss of performance compared to the previous non-linear SVMs. Notice that M is always lower than the original dimensionality of our input features. If we would have further increased the

value of M , we would have an improvement of performance until convergence to the values from Table 1.

3.5. Acoustic Scene Classification

The reported DCASE baseline¹ was tailored to a multi-class single label classification setup, with the network output layer consisting of softmax type neurons representing the 15 classes and frame-based decisions were combined using majority voting to obtain a single label per classified segment. The classification resulted in 74.8% accuracy, which was outperformed by an absolute 4% using the input features and the SVM with Laplacian Kernel.

In relation to random features, we can observed that already with a reduction of dimensionality of $M = 2^{10} = 1024$, we obtained a similar performance to the DCASE baseline (74.8%) for the Gaussian (75.3%) and the Cauchy (75.1%) kernels. Thus, reducing the dimensionality up to one sixth from the original 6,553 dims. Moreover, with a reduction of dimensionality of $M = 2^{12} = 4096$, we obtained a minimal loss of an absolute 1% for the Gaussian and Cauchy kernels. Note that for the DCASE challenge we submitted a system using the input features and the Laplacian kernel SVM. The overall classification was 60% in comparison to the reported baseline of 61%.

The advantage of random features is that they can reduce significantly the amount of the storage and the computational processing by reducing the dimensionality and using linear inner products. Unlike other dimensionality reduction methods, such as PCA, the technique presented in this paper does not need heavy computation cost, like computing eigenvectors, but we just need to generate random numbers with the appropriate kernel-related distribution. Moreover, other machine learning algorithms that employ kernels could be benefited.

Multiple applications can take advantage of random features. For example, state of the art techniques are currently dealing with features of over 10,000 dims and with hundreds of thousands of segments [6, 7, 8], which are then passed to linear SVMs. Another example is when the audio is recorded on local devices and sent to the cloud, this technique helps to compress information by reducing the cost of transmission and preserve privacy. For instance, we can compute the random features keeping the parameters W and b private. Thus, we can still process the transformed data in the cloud with linear models without revealing the actual data.

4. CONCLUSIONS

In this paper we have addressed Task 1 - Acoustic Scene Classification and have outperformed the baseline accuracy by 4% using a large set of acoustic features and non-linear SVMs. Additionally, we computed random features that approximated three types of shift-invariant kernels, which were passed to a linear SVM. We showed how the dimensionality can be decreased by one sixth with a minimal degradation of performance of about 1%. The results may have significant implications in the big data context, where high dimensional features must be stored and quickly processed.

5. REFERENCES

- [1] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumley, "Detection and classification of

¹<http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification>

Table 1: The class-wise accuracy of the four different kernels outperformed the baseline of the development set. *Note that the linear kernel is without using the random features.

| Acoustic scene | Baseline | Linear* Kernel | Gaussian Kernel | Laplacian Kernel | Cauchy Kernel |
|------------------|---------------|----------------|-----------------|------------------|---------------|
| Beach | 75.3 % | 78.2 % | 78.8 % | 77.2 % | 77.9 % |
| Bus | 71.8 % | 93.3 % | 93.6 % | 92.0 % | 92.3 % |
| Cafe/Restaurant | 57.7 % | 79.2 % | 76.9 % | 82.7 % | 78.5 % |
| Car | 97.1 % | 95.2 % | 94.9 % | 94.2 % | 95.5 % |
| City center | 90.7 % | 92.0 % | 91.0 % | 92.3 % | 89.4 % |
| Forest path | 79.5 % | 87.8 % | 89.1 % | 85.9 % | 87.2 % |
| Grocery store | 58.7 % | 74.7 % | 74.7 % | 74.7 % | 74.0 % |
| Home | 68.6 % | 66.9 % | 66.3 % | 67.3 % | 66.3 % |
| Library | 57.1 % | 66.0 % | 65.7 % | 58.3 % | 65.1 % |
| Metro station | 91.7 % | 81.4 % | 82.7 % | 83.7 % | 83.3 % |
| Office | 99.7 % | 90.4 % | 89.7 % | 92.9 % | 90.4 % |
| Park | 70.2 % | 62.2 % | 65.1 % | 61.5 % | 60.9 % |
| Residential area | 64.1 % | 62.2 % | 65.7 % | 68.3 % | 63.5 % |
| Train | 58.0 % | 59.0 % | 57.7 % | 65.7 % | 61.9 % |
| Tram | 81.7 % | 81.1 % | 82.7 % | 84.3 % | 81.7 % |
| Overall | 74.8 % | 78.0 % | 78.3 % | 78.8 % | 77.9 % |

Table 2: Overall accuracy by computing random features and using a linear SVM depending on the value of M , which is the dimensionality of the random features. Note that all the values are smaller than the input features (6,553) and the larger the value the more it compare to Table 1.

| M | Gaussian Kernel ($\gamma = 2^{-18}$) | Laplacian Kernel ($\gamma = 2^{-14}$) | Cauchy Kernel ($\gamma = 2^{-8}$) |
|----------|---|--|--|
| 2^5 | 50.4 % | 49.8 % | 48.7 % |
| 2^6 | 57.3 % | 56.0 % | 56.2 % |
| 2^7 | 64.4 % | 61.5 % | 62.9 % |
| 2^8 | 69.1 % | 66.0 % | 67.9 % |
| 2^9 | 73.0 % | 67.2 % | 72.7 % |
| 2^{10} | 75.3 % | 70.3 % | 75.1 % |
| 2^{11} | 76.1 % | 73.0 % | 75.7 % |
| 2^{12} | 77.2 % | 75.8 % | 76.9 % |

acoustic scenes and events: an IEEE AASP challenge,” in *2013 IEEE WASPAA*. IEEE, 2013, pp. 1–4.

- [2] Z. Zhang and B. Schuller, “Semi-supervised learning helps in sound event classification,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 333–336.
- [3] J. T. Geiger, B. Schuller, and G. Rigoll, “Large-scale audio feature extraction and svm for acoustic scene classification,” in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.
- [4] F. Metze, S. Rawat, and Y. Wang, “Improved audio features for large-scale multimedia event detection,” in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.
- [5] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, “Experiments on the DCASE Challenge 2016: Acoustic scene classification and sound event detection in real life recording,” in *DCASE2016 Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.

- [6] Z. Zhang, D. Liu, J. Han, and B. Schuller, “Learning audio sequence representations for acoustic event classification,” *arXiv preprint arXiv:1707.08729*, 2017.
- [7] R. Arandjelović and A. Zisserman, “Look, listen and learn,” *arXiv preprint arXiv:1705.08168*, 2017.
- [8] Y. Aytar, C. Vondrick, and A. Torralba, “Soundnet: Learning sound representations from unlabeled video,” in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.
- [9] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Advances in neural information processing systems*, 2008, pp. 1177–1184.
- [10] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.
- [11] I. S. Fuxin LiCatalin, “Random fourier approximations for skewed multiplicative histogram kernels,” in *DAGM 2010: Pattern Recognition* pp 262–271, 2010.
- [12] A. Z. Andrea Vedaldi, “Efficient additive kernels via explicit feature maps,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(34):480492, 2012, 2012.
- [13] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
- [14] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.
- [15] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: the munich versatile and fast open-source audio feature extractor,” in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.

DNN-BASED AUDIO SCENE CLASSIFICATION FOR DCASE 2017: DUAL INPUT FEATURES, BALANCING COST, AND STOCHASTIC DATA DUPLICATION

Jee-Weon Jung, Hee-Soo Heo, IL-Ho Yang, Sung-Hyun Yoon, Hye-Jin Shim, and Ha-Jin Yu[†]

School of Computer Science, University of Seoul, South Korea

ABSTRACT

In this study, we explored DNN-based audio scene classification systems with dual input features. Dual input features take advantage of simultaneously utilizing two features with different levels of abstraction as inputs: a frame-level mel-filterbank feature and segment-level identity vector. A new fine-tune cost that solves the drawback of dual input features was developed, as well as a data duplication method that enables DNN to clearly discriminate frequently misclassified classes. Combining the proposed methods with the latest DNN techniques such as residual learning achieved a fold-wise accuracy of 95.9% for the validation set and 70.6% for the evaluation set provided by the Detection and Classification of Acoustic Scenes and Events community.

Index Terms— audio scene classification, DNN, dual input feature, balancing cost, data duplication, residual learning

1. INTRODUCTION

Vast amounts of information can be acquired from an audio segment, including the surrounding environment. Audio scene classification is a task of classifying the surrounding environment based on a given audio segment. Various approaches have been explored for successful audio scene classification. For example, non-negative matrix factorization (NMF) [1], Gaussian mixture model (GMM) [2] were used in the 2016 Detection and Classification of Acoustic Scenes and Events (DCASE) competition.

In this study, we concentrated on exploiting deep learning-based systems. Recent advances in deep learning have made deep neural networks (DNNs) a state-of-the-art system for many tasks [3], [4]. We exploited approaches used in other tasks, such as image recognition and speaker verification. For example, residual network architecture [5] is a state-of-the-art system for image recognition. The identity vector (i-vector) [6], which composes state-of-the-art systems in speaker verification, is extracted from an audio segment and used as one of the two input features for the DNN classifier. The widely used mel-filterbank feature is also used simultaneously as the other input feature.

The remainder of this paper is organized as follows. Section 2 describes the proposed techniques. Subsection 2.1 describes the baseline, and the other subsections describe the methods that are stacked to compose the systems submitted to the 2017 DCASE challenge. Section 3 describes the experimental settings and configurations, along with the results and analysis.

The first and second author in this paper have the same contribution.

[†] Corresponding author

2. SYSTEM DESCRIPTION

The methods presented in this section were separately or simultaneously applied to our DNN baseline to compose four submitted systems. System 1 with dual input features was our baseline. The methods in each subsection were stacked to compose Systems 2 to 4. The newly defined balancing cost was applied to System 1 to create System 2. Stochastic data duplication based on the classification accuracy of the development set was added to System 2 to create System 3. The DNN architecture of System 3 was changed from multi-layer perceptron (MLP) to a residual network to create System 4 [5].

2.1. Dual Input Features (System 1)

Two different features were utilized as inputs for the DNN: the mel-filterbank feature (frame-level) and the i-vector (segment-level). Description about the mel-filterbank feature is omitted because it is a widely used method.

The i-vector (identity vector) [6], is a segment-level feature and represents the core identity of an audio segment. Thus, a single vector is extracted from each audio segment regardless of its length. One i-vector and several mel-filterbank features extracted from one audio segment were connected and used as the DNN input. In this case, one i-vector is duplicated and connected to the mel-filterbank features extracted in frame unit. Figure 1 shows an overview of the dual feature-based system. Although i-vectors were originally designed to represent the identity of a speaker, recent research has shown that they can be used for tasks related to scene classification [7]. Thus, we utilized i-vectors for audio scene classification.

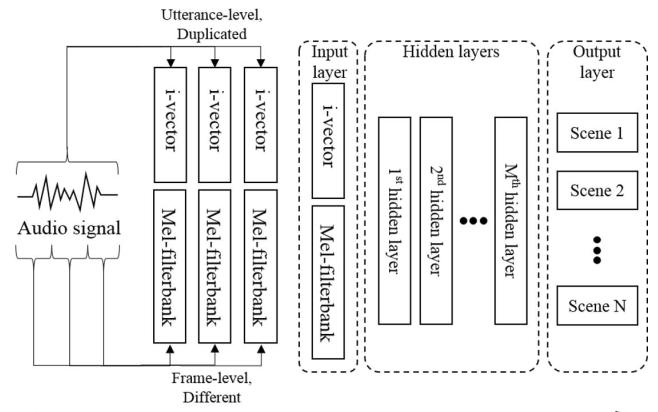


Figure 1: Illustration of mel-filterbank features and i-vectors being simultaneously used as input to the DNN.

In this study, the mel-filterbank feature and i-vectors were both used as inputs for the DNN classifier with the expectation of synergy between two features with different levels of abstraction: frame and segment, respectively. However, in our early experiments with the DCASE 2016 challenge dataset, the classification accuracy of the system using dual input features (84.1%) was almost the same as that of a system trained only with i-vectors (83.8%).

A performance gain could be achieved by pre-training the DNN classifier with only the mel-filterbank features and then fine-tuning it with both input features. The mel-filterbank feature was pre-trained by zero-masking the i-vector features and fine-tuning the network a certain number of times while maintaining the DNN architecture. It was possible to improve the scene classification accuracy of the DNN by applying the pre-training technique to System 1. However, applying the pre-training technique requires a large amount of training time and knowledge of the characteristics of input features.

2.2. Balancing Cost (System 2)

We discovered an unexpected characteristic of the DNN by visualizing the weight matrix between the input layer and the first hidden layer. The DNN tended to utilize only the i-vector and neglect the mel-filterbank features when the two features were input simultaneously. The absolute values for the weights of the network connecting the mel-filterbank features to the first hidden layer were almost zero. In contrast, the absolute value for the weight of the network connecting the i-vector to the first hidden layer was relatively high. We hypothesized that this phenomenon occurred because the i-vector, which is a segment-level feature, was easier to utilize. In contrast, the mel-filterbank feature is a frame-level feature and was harder to utilize.

To solve the problem of the network neglecting the mel-filterbank features, we added BF_1 and BF_2 to the network's negative log likelihood (NLL) fine-tune cost in (1) with α and β as scale factors. Equation (2) corresponds to the average effectiveness of one node from the input feature to all nodes in the first hidden layer. Here, $W_{x,y}$ means the weight connections between x^{th} node in the input layer and y^{th} node in the first hidden layer where X, Y refer to the number of nodes in the input layer and the first hidden layer, respectively. Thus, BF_1 in (3) represents the variance of influence for each element in the input layer. A lower BF_1 means that the input feature's elements are evenly utilized. By adding BF_1 to the objective function, each element of the input feature is forced to have similar effectiveness on the next layer.

$$cost = NLL + \alpha * BF_1(W) + \beta * BF_2(W) \quad (1)$$

$$f_1(W_x) = \frac{1}{Y} \sum_{y=1}^Y |W_{x,y}| \quad (2)$$

$$BF_1(W) = Var(f_1(W_1), f_1(W_2), \dots, f_1(W_X)) \quad (3)$$

However, the easiest way to make BF_1 equal to zero is by converging all weights to zero. To prevent this, BF_2 in (5) was introduced. For every mini-batch, the average scale of the weight matrix, W^{cur} , is compared with the scale of initial weight matrix, W^{init} . By applying ReLU function, BF_2 is active only if the scale of weight matrix has decreased. Thus, BF_2 prevents the weights from converging to zero.

$$f_2(W) = \frac{1}{X} \frac{1}{Y} \sum_{x=1}^X \sum_{y=1}^Y W_{x,y} \quad (4)$$

$$BF_2(W) = \text{ReLU}(f_2(W^{init}) - f_2(W^{cur})) \quad (5)$$

With the added terms to balance mel-filterbank features and i-vector, the two input features were evenly used judging by the absolute value of the weight matrix.

2.3. Stochastic data duplication (System 3)

A confusion matrix was used to analyze the performance of the subsystems. A combination of three channels that we divided and four cross-validation folds provided by the DCASE community generates 12 subsystems. Details regarding the channels are addressed in subsection 2.5. Figure 2 shows the confusion matrix generated by 12 subsystems from the classification experiment. Diagonal elements for the confusion matrix are omitted in Figure 2 for better visualization of misclassified audio segments. The results confirmed that the misclassification of each subsystem was concentrated in few classes unique to each subsystem. Thus, we devised a simple method to emphasize specific audio scenes in the training phase.

The suggested method (i.e., stochastic data duplication) duplicates each scene's train dataset proportional to the number of misclassified audio segments. It was applied after every epoch during the training phase based on class-wise accuracy with the validation set. Equations (6) and (7) describes how stochastic data duplication is conducted. C corresponds to the confusion matrix. $C_{j,k}$ is the number of misclassified audio segments where scene k was classified as scene j . E_k refers to the number of misclassified audio segments for scene k . In (6), A_k , between 0 and 1, is the proportion of data from scene k that is duplicated where K refers to the set of audio scenes.

$$E_k = \sum_j C_{j,k} - C_{k,k} \quad (6)$$

$$A_k = \frac{E_k}{\sum_i^K E_i} \quad (7)$$

Strictly speaking in terms of validation accuracy, this approach may not be appropriate because validation result itself is exploited. Thus, the classification accuracy derived from the validation set was described only for reference. However, from the perspective of the actual DCASE 2017 challenge, the validation set is part of the development set. Therefore, the evaluation result with stochastic data duplication applied is valid.

2.4. Residual learning (System 4)

The number of hidden layers has always been a core hyperparameter in deep learning with a major effect on system performance. A residual network was proposed in [5] to resolve this problem. As illustrated on the right side of Figure 3, the residual network is composed of several residual blocks. Each residual block has an identity mapping connection. In identity mapping [8], an input x is directly mapped to the output, and $F(x, W)$ calculate the residual only, where W is the weight matrix. Based on the identity mapping connection, the input can easily be identically mapped to the output by making the weight matrix into zeros. Thus, as far as the hardware supports it, higher performance is expected with deeper networks because unneeded residual blocks are trained for identity mappings.

$$y = F(x, W) + x \quad (8)$$

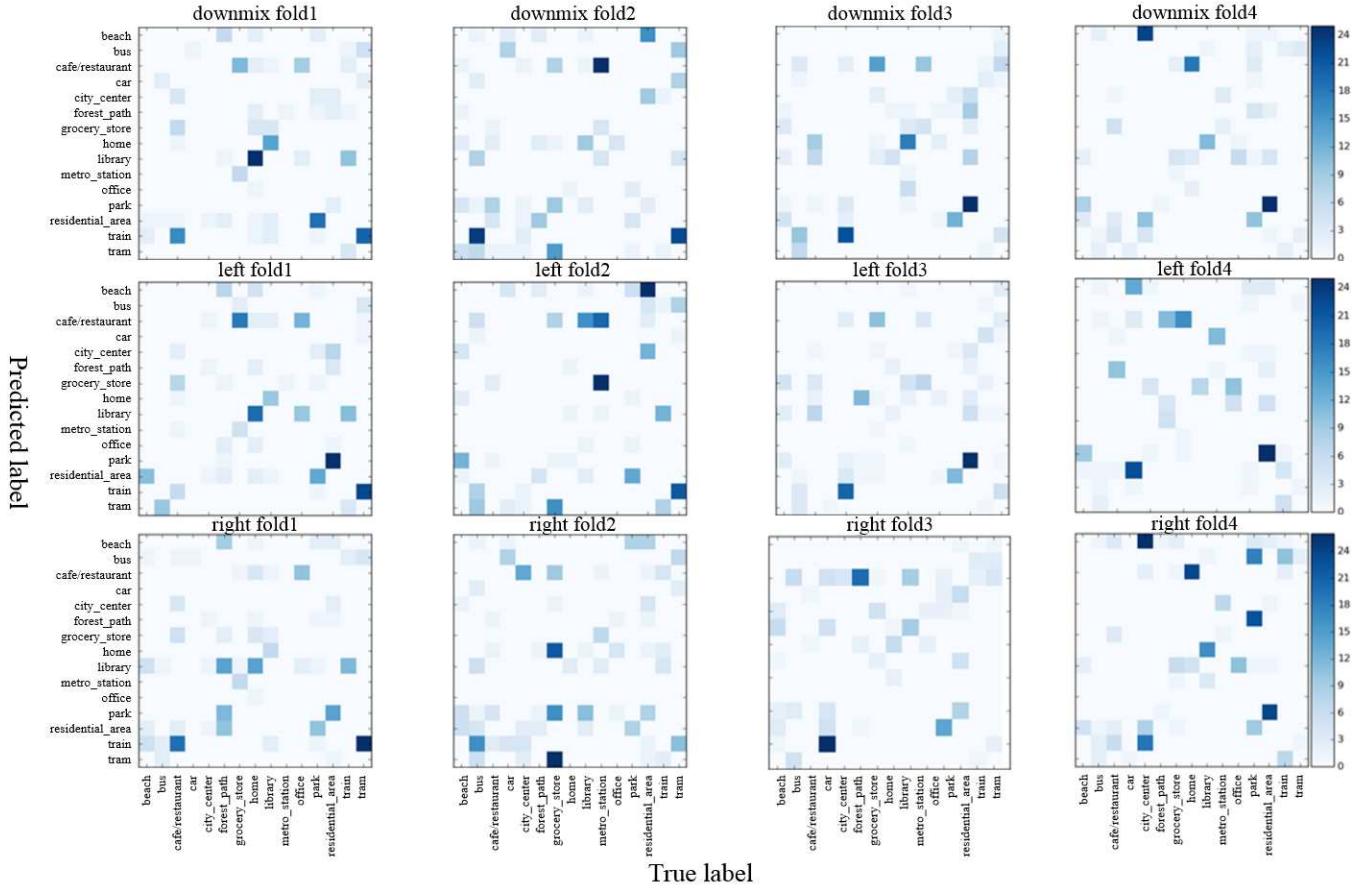
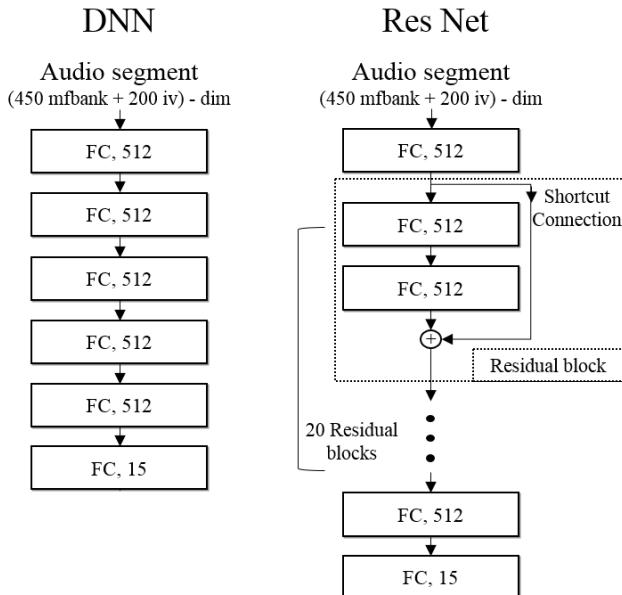


Figure 2: Confusion matrices for all channels and folds.

Figure 3: Network architectures. **Left:** DNN of 5 fully-connected hidden layers. **Right:** Residual network of 42 hidden layers (20 blocks + 2 fully-connected).

In [8], a different residual block composition was proposed where the sequence of weight operation, batch normalization, and then the activation function is replaced by a sequential implementation of batch normalization, activation function and then weight summation. The new form, which is called full pre-activation, can preserve clean information paths and gives a performance gain. This technique is adopted for System 4 by changing the network of System 3 to a 42-layer MLP.

2.5. Score-level ensemble

Audio segments from the DCASE 2017 challenge were provided at a 44.1kHz sampling rate and 24-bit resolution, in stereo. In many other studies, stereo audio is normally converted to mono audio by averaging two channels at the sample-level before processing. However based on former studies for the DCASE 2016 challenge [9], all systems in this study, including System 1 (baseline), extracted three different channels from one audio segment: the left and right channels of the stereo audio segment and the converted mono. Then, feature extraction and DNN classifier training were conducted sequentially for each channel. During the verification and evaluation stage, we use the ensemble of the decision scores of the three independent systems to make the final decision for an audio segment.

The results showed that a performance gain was achieved when ensemble was conducted on networks of different channels. Table 1

shows the classification accuracy of individual channels and when ensemble of the three channels is used.

Table 1: Classification accuracy(%) of individual channels and score-level ensemble network using the dual input feature model (System 1).

| Channel | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Average |
|-----------------|-------------|-------------|-------------|-------------|-------------|
| converted | 84.8 | 81.8 | 80.9 | 83.7 | 82.8 |
| left | 82.2 | 81.9 | 81.4 | 82.7 | 82.0 |
| right | 83.5 | 81.6 | 82.3 | 79.8 | 81.8 |
| Ensemble | 86.2 | 86.1 | 84.3 | 85.2 | 85.5 |

3. EXPERIMENTS AND RESULTS

The systems were implemented with Theano [10], [11], which is a Python library for deep learning. The open-source toolkit Kaldi [12] was used to extract i-vectors.

3.1. Feature Configuration

40-dimensional mel-filterbank feature was extracted by using 25ms windows with 10ms shift, following [13]. Linear discriminant analysis (LDA) [14] was used to reduce the dimension of mel-filterbank features to 10. The total feature of a frame is the concatenation of the mel-filterbank feature vectors of the frame and 22 frames before and after the current frame and an i-vector of the segment.

A diagonal GMM with 1024 components is trained with 60-dimensional mel-frequency cepstral coefficients, and a total variability matrix that can extract an i-vector of 200 dimensions was trained for 10 iterations. A 450-dimensional mel-filterbank features ($10 \times (22 + 1 + 22)$) and a 200-dimensional i-vector were concatenated to form 650-dimensional input feature for the DNN classifier.

3.2. Network Configuration

For Systems 1 to 3, MLP has four hidden layers, each having 512 nodes. For System 4, the MLP has 42 hidden layers, each having 512 nodes with a residual connection for every two layers.

The L-2 weight decay [15] with a lambda of 10^{-4} is applied. Dropout [16] was applied to all systems with 20% dropout at the first hidden layer and 50% for the rest of the hidden layers except for System 4. The exclusion of dropout for System 4 follows the practices in [5], [8]. Batch normalization [17] and learning rate decay following implementation in [18] were also utilized. α and β in our balancing cost were set to 1000 and 100 respectively.

For a residual network, the first and last hidden layers are normal fully-connected layers without a residual connection. In addition, when full pre-activation is applied, the batch normalization and activation functions should be excluded for the first layer in the first residual block. This is to avoid duplicate application of the batch normalization and activation function.

3.3. Results and Analysis

Table 2 presents the validation results for the four-fold cross validation of our submitted systems.

In System 2, the balancing cost reduced the classification accuracy by 0.5%p compared with System 1. However, the balancing

cost does not require pre-training with only mel-filterbank features. Thus, it can reduce the complexity of the system.

Applying stochastic data duplication led to a recognizable performance gain for all folds except fold 1. However, because the results of the validation set is utilized, strictly speaking in the perspective of validation set, this may not be considered a fair experiment. The performances of Systems 3 and 4 with validation set are therefore presented for reference only. The actual performance gain from stochastic data duplication in the evaluation is addressed in Table 2.

Table 2: Classification accuracy(%) of 4-fold average on validation set and evaluation set.

| System # | Validation set | Evaluation set |
|----------|----------------|----------------|
| System 1 | 85.5 | 67.0 |
| System 2 | 85.1 | 66.2 |
| System 3 | 95.5 | 67.3 |
| System 4 | 95.9 | 70.6 |

- System 1: dual input feature
- System 2: dual input feature + balancing cost
- System 3: dual input feature + balancing cost + stochastic data duplication
- System 4: dual input feature + balancing cost + stochastic data duplication + residual network

4. CONCLUSION

In this paper, the latest DNN-based approaches was evaluated by application to audio scene classification with proposed approaches applied. System 1(baseline), which used dual input features, showed 67.0% classification accuracy with the evaluation dataset, 6.0%p higher than the classification accuracy of the DCASE baseline system. Necessity of pre-training was resolved with balancing cost in System 2. A technique for further training the DNN for frequently misclassified classes was applied to System 3. System 4 which applied proposed approaches in Systems 1 through 3 and residual network achieved a classification accuracy of 70.6%, showing that the proposed approaches are valid.

5. ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(2017R1A2B4011609)

6. REFERENCES

- [1] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.
- [2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.
- [3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
- [5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.
- [7] B. Elizalde, H. Lei, G. Friedland, and N. Peters, “An i-vector based approach for audio scene detection,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.
- [9] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, “Cpjk submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.
- [10] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: A cpu and gpu math compiler in python,” in *Proc. 9th Python in Science Conf.*, 2010, pp. 1–7.
- [11] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, “Theano: new features and speed improvements,” *arXiv preprint arXiv:1211.5590*, 2012.
- [12] D. Povey, A. Ghoshal, G. Boulian, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al., “The kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.
- [13] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, “Deep neural networks for small footprint text-dependent speaker verification,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.
- [14] J. Qin and A. Waibel, “Application of lda to speaker recognition,” in *Interspeech*, 2000.
- [15] A. Krogh and J. A. Hertz, “A simple weight decay can improve generalization,” in *Advances in Neural Information Processing Systems (NIPS)*, 1992.
- [16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
- [17] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [18] J. Lee, J. Park, K. L. Kim, and J. Nam, “Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms,” *arXiv preprint arXiv:1703.01789*, 2017.

NEUROEVOLUTION FOR SOUND EVENT DETECTION IN REAL LIFE AUDIO: A PILOT STUDY

Christian Kroos and Mark D. Plumbley

Centre for Vision, Speech and Signal Processing (CVSSP)
 University of Surrey
 Guildford, Surrey, GU2 7XH, UK
 {c.kroos, m.plumbley}@surrey.ac.uk

ABSTRACT

Neuroevolution techniques combine genetic algorithms with artificial neural networks, some of them evolving network topology along with the network weights. One of these latter techniques is the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. For this pilot study we devised an extended variant (Joint NEAT, J-NEAT), introducing dynamic cooperative co-evolution, and applied it to sound event detection in real life audio (Task 3) in the DCASE 2017 challenge. Our research question was whether small networks could be evolved that would be able to compete with the much larger networks now typical for classification and detection tasks. We used the wavelet-based deep scattering transform and k-means clustering across the resulting scales (not across samples) to provide J-NEAT with a compact representation of the acoustic input. The results show that for the development data set J-NEAT was capable of evolving small networks that match the performance of the baseline system in terms of the segment-based error metrics, while exhibiting a substantially better event-related error rate. In the challenge, J-NEAT took first place overall according to the F1 error metric with an F1 of 44.9% and achieved rank 15 out of 34 on the ER error metric with a value of 0.891. We discuss the question of evolving versus learning for supervised tasks.

Index Terms— Sound event detection, neuroevolution, NEAT, deep scattering transform, wavelets, clustering, co-evolution

1. INTRODUCTION

Neuroevolution algorithms evolve artificial neural networks using genetic algorithms (see [1] for an overview). They have been successfully applied for finding the solution policy in intricate reinforcement tasks such as guiding a robot through a maze [2] or autonomous computer game playing [3]. Almost exclusively these tasks are simulated or situated in virtual environments, where environmental information is accessible without being affected by noise. Neuroevolution performance in real world tasks is unclear and many real world equivalents of the simulated tasks might be even out of reach due to the time consuming nature of artificial evolution. For instance, for the scenario of a wheeled robot driving autonomously through a physical maze, robots might not be available in large enough numbers and/or take weeks to complete a single full evaluation run. In this study we employ neuroevolution in an heretofore unfamiliar task, sound event detection with noisy real-world data in

The research leading to this submission was funded by EPRSC grant EP/N014111/1.

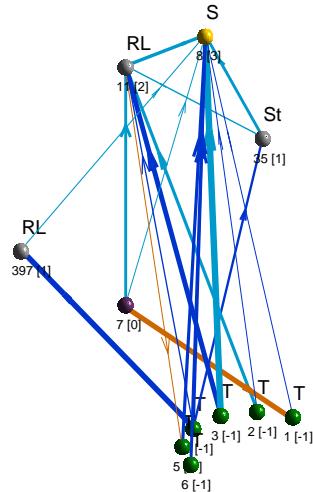


Figure 1: Structure of the first network of the ensemble with the best training performance on the full data set for the ‘people walking’ detector. Blue lines represent forward connections (light blue negative, dark blue positive weights), red/orange lines recurrent connections (orange negative, red positive weights). The relative magnitude of the weight is indicated by line width. Input nodes are depicted in green, bias nodes in dark purple and the output node in yellow. Each node has its identification number next to it and in square brackets the layer to which it was assigned. There is also a letter coding the activation function, where S = sigmoid, St = a steeper sigmoid function used in NEAT, T = tanh, I = identity, R = rectified linear, RL = leaky rectified linear and M = softmax.

the form of the DCASE 2017 challenge Task 3 (sound event detection in real life audio) dataset.

Neuroevolution can be applied for only evolving the weights of neural networks or for evolving the topology and the weights together. The latter class of methods is known as TWEANNs, Topology and Weight Evolving Artificial Neural Networks. There are two major encoding approaches: In indirect coding the code refers to rules on how to construct the network phenotype, in direct encoding all neurons and synapses are explicitly specified in the genome. In this study we used a variant of the NEAT algorithm (NeuroEvolution of Augmenting Topologies) [4, 5], which uses direct encoding. NEAT starts with a minimal network consisting of input, bias and output nodes and then grows the networks using crossover (mating) and mutations. It protects evolving, more complex networks that

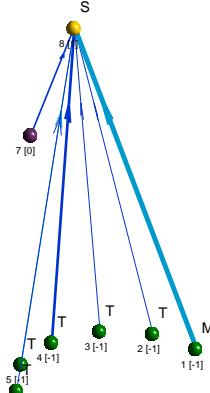


Figure 2: Structure of the second network of the ensemble with the best training performance on the full data set for the ‘people walking’ detector. See the caption of Figure 1 for the legend.

are likely to have lower fitness at the beginning through speciation: Individual networks are categorised into species at each generation based on their similarity and subsequently have to compete only with other networks within their own species.

A variant of NEAT, *Learning-NEAT* [6], has been used in classification of small data sets (e.g. Fisher’s Iris flower dataset). L-NEAT’s hybrid training scheme combines learning with evolution by incorporating backpropagation into NEAT. For the two tested datasets good performance was reported. However, the research appears to have not been followed up upon. Another variant, termed *Layered NEAT*, closer to the original NEAT, was used for shale lithofacies prediction in geology [7] with good results.

The current study makes three major contributions: The application of neuroevolution to sound event detection, a novel way of feature extraction (though based on established procedures, the deep scattering transform and k-means clustering) and a unique extension of the NEAT algorithm introducing dynamic cooperative co-evolution (for use of co-evolution in other neuroevolution methods see e.g. [8] or [9]). The latter two contributions are to a large part a consequence of the need to adapt the neuroevolution procedure to the new task. In fact, the straightforward formulation of an offline detection and classification task had also to be revised: Instead of considering the entire input data set (or at least substantial parts of it as in the mini-batch procedure in deep learning), our starting concept included a sample-by-sample step-wise evaluation within each generation, with the fittest individuals (producing the least errors) progressing the farthest and the evaluation of unfit networks stopped early. For practical reasons, the step-wise procedure had to be later parallelised and the number of steps reduced and fixed in order to take advantage of the speed of matrix computations over loop executions in our implementation language of choice Matlab (The MathWorks, Inc.).

2. METHOD

2.1. Feature extraction

Most current neuroevolution algorithms can evolve only small networks within a reasonable time frame compared to even moderately sized current classification networks with a single hidden layer let alone deep neural networks. This is certainly true for the NEAT algorithm (but see HyperNeat [10, 11] for an approach to funda-

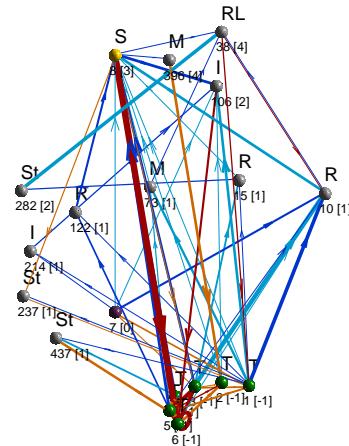


Figure 3: Structure of the third network of the ensemble with the best training performance on the full data set for the ‘people walking’ detector. See the caption of Figure 1 for the legend.

mentally change this limitation). For classification and detection tasks this entails that a compact feature representation is required even on the expense of losing detail information. Due to the computationally intense procedure of evaluating hundreds of individual networks over hundreds of generations the temporal resolution of the acoustic input has also to be kept relatively low, at least so long as standard desktop computers are used. The combined requirements call for a representation that preserves high-frequency properties of the data despite averaging over time and reduces the parameters to a small number of components characteristic for the data. The wavelet-based deep scattering transform [12, 13] fulfills the first constraint. It computes in a cascade multiple orders of coefficients that are locally translation invariant. The second order coefficients preserve transient phenomena such as the amount of attack despite averaging over larger window sizes. The deep scattering transform results, however, in a large number of coefficients for each time slice, e.g., in our case with audio default settings and a window size of 0.372 s in 520 coefficients. To reduce the number of components we applied k-means clustering. In the clustering, the dimensions were switched: The observations (samples) were treated as input variables for the clustering, while the dimensions (scattering coefficients for the different scales) were treated as observations on these variables. The resulting k by n matrix (with n being the number of samples) of the cluster centroids constitutes a low-dimensional representation of the data consisting of the major modes of the scattering scale contributions.

The deep scattering transform was computed using the *ScatNet* toolbox [14] for Matlab, progressing file by file through the raw audio files of the development data set, using the first channel of the stereo files. We used a window size of 14^2 samples and kept all other parameters at the default values recommended by the creators for audio signals [15] (two orders, $q_1 = 8$ and $q_2 = 1$). We re-normalised and log-transformed the coefficients using the routines provided by the toolbox and then concatenated all training data for the clustering. In the clustering, we set k to 17 and used the squared Euclidean distance as distance measure (keep in mind that distances were computed on the original observations, i.e. samples). We re-normalised the resulting matrix containing the centroids by

subtracting the mean and dividing it by its standard deviation - separately for each component. We also calculated the channel differences from the raw spectral information to become the 18th input component: The spectrum was computed with the same temporal resolution as the deep scattering transform using the *rastamat* toolbox [16]. For each of the 8193 spectral components the difference between the two channels was calculated and then averaged to arrive at a single number characterising the channel differences.

Since our target resolution was 1 Hz (matched to the DCASE challenge's segment length), we re-sampled the combined input data with Matlab's *resample* function.

2.2. J-NEAT - a novel cooperative co-evolution extension of the NEAT algorithm

For the event classification and detection task we devised a modified version of the NeuroEvolution and Augmenting Topology (NEAT) algorithm [4, 5]. The modifications can be divided into two classes:

- Modifications that adapted aspects of the original NEAT algorithm within its general paradigm;
- Modification that extended NEAT and changed its nature.

In the first class fall many changes that were undertaken to stabilise essential hyper-parameters, i.e., make them independent from each other and from the input value range, as well as providing more control over the rate of evolutionary weight increase and decrease. Due to space limitations we cannot describe them here in detail and focus on the more crucial second class, modifications that changed some of the core mechanisms of NEAT and transformed it into a new algorithm which we named J-NEAT for Joint Neuro-Evolution of Augmenting Topologies.

1. New nodes added by the mutation process do not intersect existing connections (synapses). If a new node is added, it is not constraint by the current synapses of the network, but established independently with two new synapses connecting it to existing nodes (or a single synapse if it is a new bias node). Additionally, new synapses between existing nodes are added in the mutation process.
2. The number of offspring for each crossover pair is not fixed to a single individual, but made dependent on the combined fitness of the two parent individuals. Higher relative fitness leads to a higher number of offspring in accordance with principles from biological evolution.
3. Contrary to standard NEAT, synapses and nodes can also be removed in the mutation process.
4. While in NEAT the activation function is fixed, we subject the choice for each node to the evolutionary process (except for the output nodes). During mutation a new activation function is selected from a pre-defined set (including sigmoid, RLU, leaky RLU, softmax, identity and tanh). This applies to both new nodes and existing nodes. The latter in the form of activation function mutations, albeit occurring only with a relatively low probability.
5. Following [7], we determine network layers by the longest path of forward connections to reach a node - not only because it enables analysing the network structure, but also because in our implementation it is required in order to be able to distinguish forward from recurrent synapses. The

additional computational effort is reduced by only partially determining layer assignments (that is, only for the affected nodes) when a new node or synapse is added and re-analysing the full network only when necessary, e.g., when a node or a synapse is removed. We designed a fast recursive algorithm for this task.

6. We included standard recurrence. Wang et al. [7] showed that the recurrence calculation in the standard implementation of NEAT is not correctly working, revisiting nodes several times, leading to higher computational costs and incorrect output values. Unfortunately, their suggested alternative implementation solves only the revisiting problem, but does not realise recurrence in the standard way. It only creates a looped calculation of the current node states, which might be helpful in itself, but differs from recurrence in the typical definition, which requires the previous states (output from the previous evaluation step) to be considered. In our approach synapses that connect nodes of the same layer or connect from a higher to a lower layer are defined as recurrent and work with the previous states of the source nodes. Layer-wise the impact of the recursive nodes is computed first and then the impact of the forward nodes before finally the activation functions are applied.
7. To break the complex classification problem into smaller partial problems, we introduced cooperative co-evolution. Several populations exist concurrently and evolve simultaneously, coupled loosely through cooperation: Each of them gets a part of the input at each sample point, but the individuals from the populations solve the overall task by cooperating across population boundaries. In our set-up, there are three populations and they receive each a third of the input, that is, initially randomly chosen 6 values per sample out of the 18 components we obtained in the feature extraction. After determining the output for the current sample, ensembles are created based on their present energy (see below): triplets consisting of one selected individual from each population. The output of each individual network is considered as the probability that the target event was detected and is treated as a confidence value. Is is averaged within ensemble, but with higher confidence values (closer to 1 or 0) boosted by:

$$\tilde{w}_k = 1 - 4 p_k (1 - p_k), \quad (1)$$

$$\tilde{p}_k = (p_k - 0.5) \tilde{w}_k, \quad (2)$$

$$p_o = \frac{\frac{1}{N} \sum_{i=1}^N \tilde{p}_i}{\frac{1}{N} \sum_{i=1}^N \tilde{w}_i} + 0.5, \quad (3)$$

where p_k is the estimate of ensemble member k (from population k), N the number of ensemble members (populations), \tilde{w} the boosting factor, \tilde{p} the adjusted estimate and p_o the averaged adjusted final output. This allows individual members of the ensemble with high confidence to override the influence of the others if these are weighing in for the opposite decision but are not too far from the 0.5 chance level border. Based on whether the result is a true positive, true negative, false positive or false negative rewards and penalties are given equally to all members of the ensemble and then converted into an energy measure. Ensembles are dissolved and reformed at the next evaluation sample and, thus,

also not carried over to the next generation. However, the individual members keep their acquired energy.

We used three populations with each 400 individuals. For each sound event class a separate neuroevolution run was conducted, i.e., all networks had always only one output node. Each run started with all individuals possessing a minimal, fully forward-connected network, consisting of six input nodes, one bias node and one output node with randomly assigned weights (drawn from a standard uniform distribution). The activation function of the output node was set to the sigmoid function. For implementation reasons input nodes possess an activation function, too. In the beginning we set them to the identity function, but in test runs the evolutionary process often replaced them by the hyperbolic tangent function in successful networks. We therefore decided to make \tanh the default start input activation function.

In each generation and each evaluation step we evaluated 250 randomly selected samples simultaneously. Their selection indices were determined by drawing in each generation consecutive distance values from a normal distribution with mean 20 and standard deviation 1.79 and accumulating them (e.g., values 18, 26 and 15 resulting in the selection of the 18th, 44th and 59th sample). Each evaluation step progressed then by one sample, keeping all the distances intact. Forty-four steps were taken per generation, which meant that each sample point was on average evaluated 2.6 times. A complete run for a single class consisted of 500 generation. The best fitness value achieved over the course of the 500 generations was stored and the ensemble which accomplished it taken as the final classifier network. The results were aggregated over classes and folds and the final evaluation conducted using the Python routine provided by the DCASE 2017 Task 3 organisers [17].

For comparison, we also ran J-NEAT with no co-evolution, employing only a single population and each network receiving the entire input per sample. Finally, to gauge the impact of the feature extraction method and create a minimal node-size classifier using learning, we also applied a simple single-layer feed-forward network with no hidden nodes. The learning rate was set to 0.2. No regularisation techniques were used.

3. RESULTS

Figure 1 to 3 show as an example the three networks that formed the ensemble with the best training performance on the full development data set for the ‘people walking’ detector. Table 1 summarises the performance results across the four-fold validation and Table 2 shows the results from the challenge evaluation [18].

Table 1: Performance on the development test data set showing both segment- and event-based evaluation results.

| Method | Seg. ER | Seg. F1 | Ev. ER | Ev. F1 |
|------------------|---------|---------|--------|--------|
| Baseline | 0.72 | 51.40 | 3.30 | 6.74 |
| J-NEAT ensemble | 0.73 | 49.24 | 1.46 | 6.46 |
| J-NEAT plain | 0.72 | 50.55 | 1.37 | 5.66 |
| Single-layer FFN | 0.69 | 56.47 | 1.40 | 5.85 |

4. DISCUSSION

One could question the wisdom of using neuroevolution for tasks that are in principle solvable with supervised learning, presenting

Table 2: Challenge evaluation.

| Method | Segment-based value | | Rank | |
|------------------|---------------------|------|------|----|
| | ER | F1 | ER | F1 |
| DCASE Baseline | 0.936 | 42.8 | 19 | 8 |
| J-NEAT ensemble | 0.898 | 44.9 | 15 | 1 |
| J-NEAT plain | 0.891 | 41.6 | 14 | 12 |
| Single-layer FFN | 1.014 | 43.8 | 28 | 3 |

no obstacles for backpropagation of the error through the network. Our interest in neuroevolution stems from the desire to develop parsimonious neural network-based detectors and classifiers, consisting ideally only of a few nodes. These small systems could be used in, for instance, autonomous micro-robots such as insect-like air-born robots. A powerful all-purpose (hard-coded) feature extracting system might be available, but the detectors and classifiers would need to be of minimal size: In real-world navigation and other task solving a large number of them would be required, but the number of available neurons would be very limited. Neuroevolution using TWEANNs offers the potential to find small network solutions without a human experimenter having to specify network size and topology.

The results using the development data set demonstrated that neuroevolution techniques can evolve small networks able to compete with the much bigger network of the baseline. Here they were, however, still outperformed by a minimal human-designed learning network. The picture changes when taking the challenge results into account. On these previously unseen data – never touched upon in the system development – abstraction and regularisation shortcomings become evident. The two J-NEAT systems performed approximately equally according to the ER metric. Both were above average of all 34 submitted systems and substantially outperformed our learning simple single-layer network. When looking at the complementary F1 metric the ensemble-based J-NEAT system clearly got the upper hand over the plain J-NEAT system and indeed performed better than all other submitted systems.

With caution this can interpreted as evidence for the ability of ensemble-based J-NEAT classifiers to tune in on specific characteristics of some but not all of the involved audio event classes and avoid to a certain degree interference from overlapping sound events in this difficult polyphonic task of the DCASE 2017 challenge. The challenge results certainly establish J-NEAT as a serious alternative to the DNN approaches, which dominated not just this challenge, but by now reign over most of machine learning.

Future work will investigate task dependency problems more closely. For static co-evolution it is advantageous for the subordinate tasks to be as independent as possible [19]. If the input is split to form subtasks as in our classification and detection system, independence is not likely. However, in our system ensemble assignments change dynamically at each evaluation step. It is therefore unclear whether greater independence would improve the results. We will also explore alternatives to the current approach, for instance by providing all populations with the full input, but still using cooperative co-evolution in the form of dynamic cross-population ensembles. A specialisation of individual networks on parts of the input will then be left to evolution as well. Another alternative will be to enable symbiotic relationships between networks of different populations through synapses connecting networks across the population barrier.

5. REFERENCES

- [1] D. Floreano, P. Dürr, and C. Mattiussi, “Neuroevolution: from architectures to learning,” *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.
- [2] J. Lehman and K. O. Stanley, “Exploiting open-endedness to solve problems through the search for novelty,” in *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, 2008, pp. 329–336.
- [3] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, “A neuroevolution approach to general Atari game playing,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 355–366, 2014.
- [4] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.
- [5] ———, “Competitive coevolution through evolutionary complexification,” *Journal of Artificial Intelligence Research*, vol. 21, pp. 63–100, 2004.
- [6] L. Chen and D. Alahakoon, “NeuroEvolution of augmenting topologies with learning for data classification,” in *International Conference on Information and Automation (ICIA 2006)*. IEEE, 2006, pp. 367–371.
- [7] G. Wang, G. Cheng, and T. R. Carr, “The application of improved NeuroEvolution of Augmenting Topologies neural network in Marcellus Shale lithofacies prediction,” *Computers & Geosciences*, vol. 54, pp. 50–65, 2013.
- [8] F. Gomez, J. Schmidhuber, and R. Miikkulainen, “Accelerated neural evolution through cooperatively coevolved synapses,” *Journal of Machine Learning Research*, vol. 9, no. May, pp. 937–965, 2008.
- [9] Y. Liu, X. Yao, and T. Higuchi, “Evolutionary ensembles with negative correlation learning,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.
- [10] J. Gauci and K. O. Stanley, “Autonomous evolution of topographic regularities in artificial neural networks,” *Neural Computation*, vol. 22, no. 7, pp. 1860–1898, 2010.
- [11] K. Stanley and P. Verbancsics, “Constraining connectivity to encourage modularity in HyperNEAT,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, 2011.
- [12] J. Andén and S. Mallat, “Deep scattering spectrum,” *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.
- [13] S. Mallat, “Group invariant scattering,” *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.
- [14] L. Sifre, M. Kapoko, E. Oyallon, and V. Lostanlen, *Scatnet: a MATLAB toolbox for scattering networks*, 2013.
- [15] “ScatNet - Quickstart for audio processing,” online web resource. [Online]. Available: <http://www.di.ens.fr/data/software/scatnet/quickstart-audio/>
- [16] D. P. W. Ellis, “PLP and RASTA (and MFCC, and inversion) in Matlab,” 2005, online web resource. [Online]. Available: <http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/>
- [17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: tasks, datasets and baseline system,” in *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, submitted.
- [18] “DCASE 2017 - Task 3 (Sound event detection in real life audio) - results,” online web resource. [Online]. Available: <http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-sound-event-detection-in-real-life-audio-results>
- [19] R. Chandra, M. Frean, and M. Zhang, “On the issue of separability for problem decomposition in cooperative neuro-evolution,” *Neurocomputing*, vol. 87, pp. 33–40, 2012.

COMBINING MULTI-SCALE FEATURES USING SAMPLE-LEVEL DEEP CONVOLUTIONAL NEURAL NETWORKS FOR WEAKLY SUPERVISED SOUND EVENT DETECTION

Jongpil Lee¹, Jiyoung Park¹, Sangeun Kum¹, Youngho Jeong², Juhannam¹,

¹ Graduate School of Culture Technology, KAIST, Korea,

² Realistic AV Research Group, ETRI, Korea,

{richter, jypark527, keums, juhannam}@kaist.ac.kr, yhcheong@etri.re.kr

ABSTRACT

This paper describes our method submitted to large-scale weakly supervised sound event detection for smart cars in the DCASE Challenge 2017. It is based on two deep neural network methods suggested for music auto-tagging. One is training sample-level Deep Convolutional Neural Networks (DCNN) using raw waveforms as a feature extractor. The other is aggregating features on multi-scaled models of the DCNNs and making final predictions from them. With this approach, we achieved the best results, 47.3% in F-score on subtask A (audio tagging) and 0.75 in error rate on subtask B (sound event detection) in the evaluation. These results show that the waveform-based models can be comparable to spectrogram-based models when compared to other DCASE Task 4 submissions. Finally, we visualize hierarchically learned filters from the challenge dataset in each layer of the waveform-based model to explain how they discriminate the events.

Index Terms— Sound event detection, audio tagging, weakly supervised learning, multi-scale features, sample-level, convolutional neural networks, raw waveforms

1. INTRODUCTION

Understanding the sounds of everyday life has received great attention in recent years due to its practical applications such as the hearing impaired, smart cars and smart appliances [1, 2, 3, 4, 5]. Among others, Sound Event Detection (SED) is a particularly challenging task because it predicts not only possible descriptive words of environment sounds but also their start and end times. Most SED systems are based on hard annotated data where both event classes and their timestamps are present [4, 6, 7, 8, 9]. However, it is time-consuming and expensive to construct a large dataset with such labels and so this has limited the use of highly data-driven learning algorithms such as deep neural networks. To take account of this problem, Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge 2017 has opened a new task [5]: Large-scale weakly supervised sound event detection for smart cars where subtask A is audio tagging and subtask B is sound event detection. Especially for subtask B, the objective is to construct novel SED system based on a dataset without timestamps.

Recent deep learning based SED systems that use timestamps information can be divided into two approaches. One is using the sequence information to predict the order of the timestamps, for example, using Recurrent Neural Networks (RNN) [6, 7]. The other is dividing an audio clip into the same length of small segments (e.g. 1 second long) and using the segments as input for the models, for

example, using Deep Neural Networks (DNN) [4] or Convolutional Neural Networks (CNN) [9]. This segmentation-based approach does not use sequence information, but it can capture local audio characteristics well [4, 9]. The effectiveness was shown in many audio tagging tasks [2, 10, 11, 12]. The main difference is that segments in the SED systems have their own labels depending on the presence of events at the moment whereas those in audio tagging systems are annotated with the same labels as long as they are from the same audio file.

For the weakly supervised SED task, we mix the two settings. That is, in training phase, we use the same labels for all segments within an audio file whereas, in test phase, we regard the outputs for segments as separate event predictions. With this setting, we can apply some of the methods developed primarily for audio tagging to the weakly supervised SED task. In the following sections, we describe the methods and show that the approach is effective in our target task.

2. PROPOSED METHOD

2.1. Combination of Multi-Scale Features

Event sounds have different timbre patterns in terms of feature hierarchy and time-scales [13, 11]. For example, bicycle and motorcycle sounds are generated as a repetition of specific sound sources. They tend to be local and repetitive within an audio clip. On the other hand, car and train sounds are more sustained or ambient. They are relatively more global and require longer audio segments to discriminate them. We previously addressed the issue by using multiple CNNs, each of which covers different time scales [11, 14]. The proposed method is performed in three steps: feature learning by multiple CNNs, feature aggregation, and final classification. The CNNs are trained with the sound labels, taking different input sizes to capture both local and global characteristics of the sounds. We then use these trained networks as a feature extractor. Since these feature extractors are trained with different input sizes, these can capture different audio characteristics. After the features are extracted, we summarize them for the given task-specific format. For example, for the audio tagging task, we summarize segment-level features to audio-clip-level by averaging the whole segment features. For the SED task, segment-level features are averaged every second. Lastly, the final prediction is performed using a fully-connected neural network for each subtask.

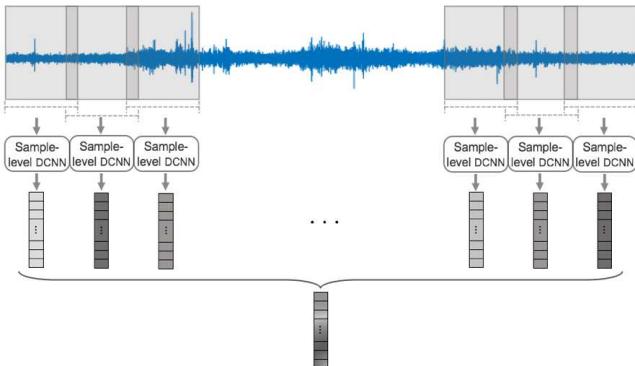


Figure 1: Feature aggregation method for subtask A of Task 4 (audio tagging). The features of models with different input sizes are concatenated.

2.2. Sample-level Deep Convolutional Neural Networks

Using raw audio as input allows the network to learn very low-level features. Generally, in audio classification tasks, raw waveforms are converted to a time-frequency representation before used as input to the system. However, in this preprocessing stage, short-time Fourier transform (STFT) parameters, such as hop size or window size, are often ignored in parameter optimization even though optimal parameters for each sound class may vary [15, 16]. To take account of this and also to avoid exhausting parameter search, we used the previously proposed network as a feature extractor which learns from raw waveforms with very small sample-level filters [14].

3. EXPERIMENTS

3.1. Datasets

The DCASE Challenge 2017 Task4 uses a subset of AudioSet [2]. This subset consists of 17 sound events and the classes are unbalanced and multi-labeled. The task setup comes with training, testing and evaluation set. The split includes 51172, 488, and 1103 audio clips, respectively. Because the evaluation set is saved for the challenge evaluation, we split the training set by randomly selecting 10% of audio clips for each class and using them as a validation set. Since the audio clips are multi-labeled, we in fact selected more than 10% audio clips per class. As a result, the sub-training set consists of 45313 clips and the validation set contains 5859 clips.

3.2. CNN Models

We followed CNN model configuration and training settings in our previous work [15]. For example, all audio clips are segmented according to the network input size and each segment is used as a single sample for training with its corresponding event labels. We used a total of eight CNN models with different lengths of waveforms as inputs from 372ms, 557ms, 627ms, 743ms, 893ms, 1486ms, 2678ms and up to 3543ms. After the networks are trained, they can directly predict the results of subtask A and subtask B. This is termed as Sample level Deep Convolutional Neural Networks (SD-CNN) in our experiment.

The difference from the previous work is that the audio sampling rate increased by a factor of 2 (i.e. 44100 Hz) and the model

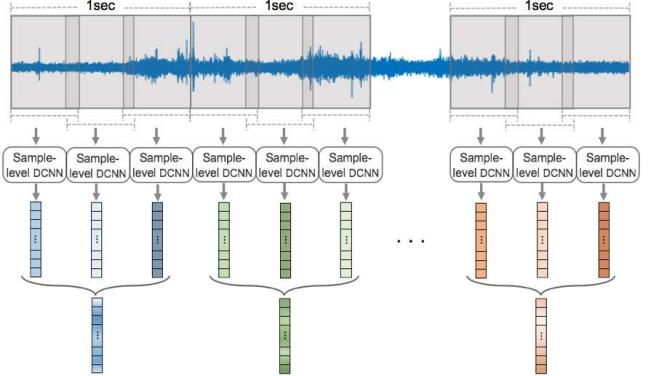


Figure 2: Feature aggregation method for subtask B of Task 4 (sound event detection). The features of models with different input sizes are concatenated.

size is expanded accordingly (11 to 16 convolution layers and 128 to 512 filters). Also, since we measure detection performance rather than ranking, we predicted the presence of tags with a threshold value. In the SDCNN model, we used 0.1 for the tagging task and 0.5 for the SED task.

3.3. Feature Aggregation and Final Classification

We also used the trained CNN models as a feature extractor instead of using their prediction results, following the multi-level and multi-scale feature aggregation approach [11]. By multi-level features, we mean to use top three hidden layers in the sample-level CNN models. The purpose of using multi-level features is considering various abstraction levels of the sound tags. Although the tag descriptions are limited to smart cars and so the diversity in feature hierarchy is not strong, we put the multi-level concatenation in our experiment.

For the tagging task, the features of all segments are averaged into a single feature vector for each model as shown in Figure 1. We then combined multi-scaled features and fed them into a fully connected layer for final decision. For the SED task, we summarized the features every second as depicted in Figure 2. If the input length of the CNN model is less than one second, we computed the number of segments by dividing one second by the input size and rounding it up, and overlapped adjacent segments such that all segments fit within one second. We then averaged the features from segments as a single vector. If the input length exceeds one second, we extracted a single feature only. We then move the model by one second for next event detection. Finally, we fed the features into a fully connected layer to make a final decision for each period. We term this setup as Multi-Level Multi-Scale (MLMS) model. In all MLMS models, we used a threshold of 0.2 for tagging predictions and 0.5 to SED predictions to make a final decision.

3.4. List of Submissions

Based on the experimental setup above, we submitted four settings of models for DCASE Challenge 2017 Task 4 (Large-scale weakly supervised sound event detection for smart cars) as follows:

- SDCNN: Sample-level Deep Convolutional Neural Networks that takes 893ms of audio as input. This is one of the models used as a feature extractor for the rest submissions.

Table 1: The class-wise performance of submitted systems and their comparisons on the development set. In the middle section, we show the results with multi-level only (termed as ML) to observe the sensitivity of tag prediction to different input sizes (the numbers after ML). When the performance of a tag has a trend according to the input size, we highlighted the tag and the value of the optimal input size.

| Subtask A F-score | SDCNN893 | ML372 | ML557 | ML627 | ML743 | ML893 | ML1486 | ML2678 | ML3543 | MLMS5 | MLMS3 | MLMS8 |
|---------------------------------|----------|-------------|-------|-------------|-------------|-------------|-------------|--------|-------------|-------|-------|-------|
| Train horn | 48.7 | 22.8 | 32.4 | 36.8 | 26.3 | 28.5 | 33.3 | 36.8 | 27.0 | 47.6 | 41.0 | 41.0 |
| Air horn, truck horn | 43.9 | 27.7 | 27.7 | 27.8 | 31.5 | 35.9 | 22.8 | 17.1 | 37.8 | 35.0 | 36.8 | 41.0 |
| Car alarm | 27.7 | 0.0 | 6.4 | 0.0 | 6.4 | 0.0 | 0.0 | 0.0 | 0.0 | 6.4 | 0.0 | 0.0 |
| Reversing beeps | 40.0 | 6.5 | 18.1 | 12.5 | 23.5 | 28.6 | 28.6 | 18.2 | 12.5 | 18.1 | 33.3 | 18.2 |
| Ambulance (siren) | 40.0 | 21.6 | 24.4 | 40.9 | 29.2 | 34.1 | 15.8 | 21.6 | 10.8 | 50.9 | 27.9 | 36.4 |
| Police car (siren) | 44.6 | 38.6 | 43.6 | 43.2 | 44.4 | 46.3 | 41.3 | 44.9 | 47.1 | 42.9 | 46.6 | 43.9 |
| Fire engine, fire truck (siren) | 40.8 | 43.5 | 43.4 | 42.0 | 44.0 | 42.4 | 44.9 | 42.8 | 46.9 | 46.8 | 40.4 | 42.2 |
| Civil defense siren | 67.4 | 78.8 | 77.1 | 76.7 | 80.0 | 77.7 | 77.6 | 74.6 | 72.7 | 77.8 | 73.2 | 76.7 |
| Screaming | 52.6 | 40.9 | 41.6 | 47.8 | 48.9 | 50.0 | 48.9 | 36.7 | 44.9 | 53.1 | 39.1 | 48.0 |
| Bicycle | 42.5 | 58.1 | 58.0 | 52.3 | 55.1 | 48.5 | 56.1 | 55.7 | 44.8 | 53.1 | 45.6 | 61.0 |
| Skateboard | 71.4 | 71.1 | 70.0 | 72.4 | 75.0 | 80.0 | 72.4 | 73.3 | 71.2 | 77.2 | 73.7 | 71.4 |
| Car | 23.1 | 30.7 | 30.3 | 32.3 | 30.7 | 32.9 | 32.1 | 33.5 | 31.8 | 32.8 | 33.8 | 35.0 |
| Car passing by | 19.0 | 5.7 | 4.9 | 12.7 | 14.6 | 13.6 | 23.2 | 12.8 | 23.5 | 13.0 | 10.0 | 16.6 |
| Bus | 29.6 | 26.1 | 37.0 | 38.2 | 33.3 | 37.0 | 31.7 | 34.5 | 32.5 | 34.6 | 30.0 | 33.3 |
| Truck | 32.9 | 42.8 | 40.7 | 41.1 | 42.9 | 41.1 | 41.8 | 44.5 | 43.9 | 43.0 | 42.7 | 40.4 |
| Motorcycle | 53.8 | 61.0 | 54.2 | 52.8 | 52.6 | 46.6 | 49.1 | 54.5 | 46.1 | 53.3 | 46.4 | 57.6 |
| Train | 61.2 | 58.4 | 62.2 | 64.4 | 62.9 | 64.4 | 65.2 | 63.7 | 62.5 | 67.3 | 68.8 | 68.1 |
| Subtask B ER | | | | | | | | | | | | |
| Train horn | 0.85 | 0.90 | 0.93 | 0.90 | 0.92 | 0.85 | 0.93 | 0.91 | 0.90 | 0.84 | 0.91 | 0.87 |
| Air horn, truck horn | 0.82 | 0.86 | 0.91 | 0.86 | 0.90 | 0.81 | 0.93 | 0.88 | 0.92 | 0.82 | 0.89 | 0.85 |
| Car alarm | 0.97 | 0.94 | 0.97 | 0.98 | 0.96 | 0.97 | 0.92 | 0.98 | 0.97 | 0.95 | 0.96 | 0.95 |
| Reversing beeps | 0.91 | 0.98 | 0.92 | 0.92 | 0.92 | 0.92 | 0.92 | 0.91 | 0.90 | 0.91 | 0.94 | 0.88 |
| Ambulance (siren) | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Police car (siren) | 1.12 | 1.14 | 1.09 | 1.16 | 1.15 | 1.04 | 1.23 | 1.16 | 1.13 | 1.15 | 1.11 | 1.09 |
| Fire engine, fire truck (siren) | 0.98 | 0.95 | 0.99 | 1.00 | 0.99 | 0.98 | 0.98 | 1.00 | 1.00 | 1.01 | 0.99 | 0.97 |
| Civil defense siren | 0.58 | 0.60 | 0.61 | 0.62 | 0.63 | 0.58 | 0.63 | 0.62 | 0.64 | 0.62 | 0.59 | 0.59 |
| Screaming | 0.93 | 0.95 | 0.95 | 0.95 | 0.95 | 0.91 | 0.94 | 0.94 | 0.98 | 0.92 | 0.91 | 0.91 |
| Bicycle | 0.87 | 0.87 | 0.90 | 0.86 | 0.85 | 0.91 | 0.96 | 0.96 | 0.95 | 0.90 | 0.88 | 0.91 |
| Skateboard | 0.80 | 0.78 | 0.80 | 0.75 | 0.78 | 0.80 | 0.81 | 0.83 | 0.81 | 0.73 | 0.79 | 0.72 |
| Car | 3.32 | 3.29 | 3.66 | 2.83 | 3.42 | 3.03 | 3.62 | 3.29 | 2.82 | 3.00 | 3.00 | 2.85 |
| Car passing by | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Bus | 1.03 | 1.00 | 1.01 | 1.02 | 1.01 | 1.00 | 1.00 | 1.02 | 0.99 | 1.03 | 1.01 | 0.98 |
| Truck | 0.96 | 0.90 | 0.97 | 0.95 | 0.97 | 0.92 | 0.97 | 0.95 | 0.95 | 0.92 | 0.98 | 0.95 |
| Motorcycle | 0.92 | 0.90 | 0.97 | 0.91 | 0.91 | 0.94 | 0.99 | 0.91 | 0.96 | 0.95 | 0.91 | 0.93 |
| Train | 0.93 | 0.92 | 0.95 | 0.95 | 0.95 | 0.94 | 0.94 | 0.90 | 0.89 | 0.94 | 0.93 | 0.92 |

- MLMS5: Multi-Level and Multi-Scale features extracted from models taking 372ms, 557ms, 627ms, 743ms and 893ms as input.
- MLMS3: Multi-Level and Multi-Scale features extracted from models taking 1486ms, 2678ms, and 3543ms as input.
- MLMS8: Multi-Level and Multi-Scale features extracted from models taking 372ms, 557ms, 627ms, 743ms, 893ms, 1486ms, 2678ms, 3543ms as input.

Details about the models can be found in our DCASE submission webpage link¹.

4. RESULTS AND DISCUSSION

4.1. Evaluation on the Development set

We report the performance of the proposed method in Table 2 (tagging) and Table 3 (SED). From the results, we can find that the feature aggregation and final classification stage improve performance compared to the direct result of SDCNN. Also, as the number of model combinations increases, the performance is generally improved as well.

¹<https://github.com/jongpillee/dcase2017submission>

Table 2: Instance-based results of submitted systems for subtask A of Task 4 (audio tagging)

| | Development set | | | Evaluation set | | |
|-------|-----------------|-------|-------|----------------|-------|-------|
| | F-score | Prec. | Rec. | F-score | Prec. | Rec. |
| SDCNN | 37.8% | 26.7% | 64.8% | 40.3% | 31.3% | 56.7% |
| MLMS5 | 44.3% | 38.8% | 51.7% | 47.3% | 48.0% | 46.6% |
| MLMS3 | 42.2% | 39.0% | 45.9% | 47.2% | 49.6% | 45.0% |
| MLMS8 | 43.8% | 39.2% | 49.5% | 47.1% | 48.5% | 45.9% |

Table 3: Instance-based results of submitted systems for subtask B of Task 4 (sound event detection)

| | Development set | | Evaluation set | |
|-------|-----------------|---------|----------------|---------|
| | ER | F-score | ER | F-score |
| SDCNN | 0.88 | 28.1% | 0.82 | 39.4% |
| MLMS5 | 0.86 | 30.7% | 0.78 | 42.6% |
| MLMS3 | 0.86 | 31.2% | 0.78 | 44.2% |
| MLMS8 | 0.84 | 34.2% | 0.75 | 47.1% |

We report class-wise performance as well on Table 1. From the class-wise tagging results, we can find the sensitivity of tags to different time scales. For example, tags such as *Reversing beeps*, *Ambulance (siren)*, *Screaming*, *Civil defense siren* and *Skateboard* are optimal around one second. On the other hand, *Bicycle* and *Mo-*

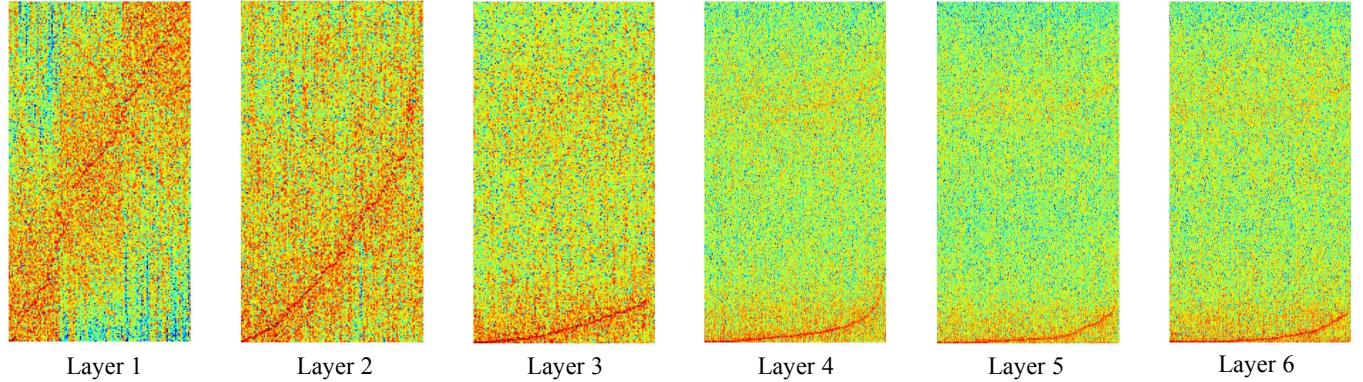


Figure 3: Spectrum of the filters in the sample-level convolution layers which are sorted by the frequency at the peak magnitude. The x-axis represents the index of the filters and the y-axis represents the frequency. The visualization was performed using a gradient ascent method to obtain the input waveform that maximizes the activation of a filter in the layers [15].

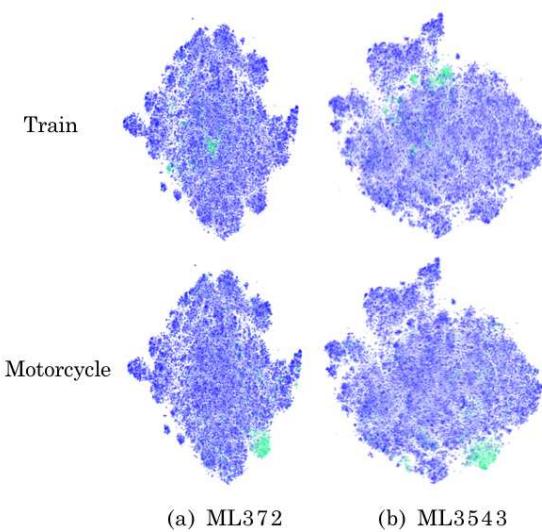


Figure 4: Visualization of aggregated features with *Train* tags and *Motorcycle* tags using t-SNE in the training set. Each dot corresponds to one audio clip. The green dots indicate those belonging to the tag denoted in the left side. ML372 indicates the model with multi-level features and 372ms as input.

torcycle favor shorter seconds, and *Police car (siren)*, *Car passing by*, *Bus* and *Train* prefer longer seconds. These trends can also be observed in Figure 4 where we displayed 2-D embedding space of aggregated features in the ML models using t-Distributed Stochastic Neighbor Embedding (t-SNE). We can see that audio clips with *Train* tags are more closely clustered in ML3543 whereas those with *Motorcycle* tags are more in ML372. This may explain why combining multi-scale features improves the performance. Also, in Table 1, we can find that SCDNN shows good results in class-wise performance. Especially when the sound is alarming ones, for example, *Car alarm*, *Reversing beeps*, *Ambulance (siren)* and *Police car (siren)*. However, the MLMS models achieve better performance on instance-based metrics as shown in Table 2 and 3. This is probably because about half of the dataset have car tags and the

MLMS models tend to improve the performance for those with the car tags significantly.

4.2. Comparison with other submissions in DCASE 2017

Nine teams submitted their algorithms to subtask A and seven teams to subtask B in the DCASE2017 Task 4. Our team was ranked at the 5th for subtask A and at the 3rd for subtask B. Most submitted algorithms used mel-scaled spectrogram as input and CNN as a classifier. These results show that our model using raw waveform as input can be comparable those using spectrogram.

4.3. Filter Visualization of SDCNN

We visualize learned filters on each layer in the sample-level CNN. Figure 3 shows the filters obtained from a gradient ascent method [15] and sorted with the frequency at the peak magnitude. We can observe that they are sensitive to more log-scaled in frequency as the layer goes up. Compared to the learned filters from music audio 3, these filters tend to have more low-frequency concentration and less complex patterns.

5. CONCLUSIONS

In this paper, we presented sample-level DCNN models using raw waveforms and multi-scale feature aggregation method developed for the DCASE Challenge 2017. We showed that our proposed method is comparable to CNN-based models using spectrogram as input. Class-wise performance and feature visualization indicate that audio clips with different tags are optimal in different time scales. Combining the multi-scaled features improves overall performance. We also visualized hierarchically learned filters in the sample-level CNN. They showed the spectral patterns are adapted to the characteristic of the acoustic scene sounds.

6. ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (1711055381, Development of Human Enhancement Technology for auditory and muscle support).

7. REFERENCES

- [1] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.
- [2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.
- [3] T. Heittola, A. Mesaros, and T. Virtanen, “DCASE2016 baseline system,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [4] Q. Kong, I. Sobieraj, M. Plumley, and W. Wang, “Deep neural network baseline for DCASE challenge 2016,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [5] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*. November 2017.
- [6] S. Adavanne, G. Parascandolo, P. Pertil, T. Heittola, and T. Virtanen, “Sound event detection in multichannel audio using spatial and harmonic features,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [7] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, June 2017.
- [8] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, “Experiments on the dcse challenge 2016: Acoustic scene classification and sound event detection in real life recording,” *arXiv preprint arXiv:1607.06706*, 2016.
- [9] A. Gorin, N. Makhazanov, and N. Shmyrev, “DCASE2016 sound event detection system based on convolutional neural network,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [10] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al., “CNN architectures for large-scale audio classification,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.
- [11] J. Lee and J. Nam, “Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging,” *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1208–1212, 2017.
- [12] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.
- [13] Y. Xu, Q. Huang, W. Wang, and M. D. Plumley, “Hierarchical learning for dnn-based acoustic scene classification,” *arXiv preprint arXiv:1607.03682*, 2016.
- [14] J. Lee and J. Nam, “Multi-level and multi-scale feature aggregation using sample-level deep convolutional neural networks for music classification,” *International Conference on Machine Learning (ICML), Machine Learning for Music Discovery Workshop*, 2017.
- [15] J. Lee, J. Park, K. L. Kim, and J. Nam, “Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms,” *Sound and Music Computing Conference (SMC)*, pp. 220–226, 2017.
- [16] K. Choi, D. Joo, and J. Kim, “Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras,” *arXiv preprint arXiv:1706.05781*, 2017.

ENSEMBLE OF CONVOLUTIONAL NEURAL NETWORKS FOR WEAKLY-SUPERVISED SOUND EVENT DETECTION USING MULTIPLE SCALE INPUT

Donmoon Lee^{1,2}, Subin Lee^{1,2}, Yoonchang Han², Kyogu Lee¹

¹ Music and Audio Research Group, Seoul National University, Seoul, Korea,

² Cochlear.ai, Seoul, Korea,

{dmlee, sblee, ychan}@cochlear.ai, kglee@snu.ac.kr

ABSTRACT

In this paper, we propose to use an ensemble of convolutional neural networks to detect audio events in the automotive environment. Each of the networks is based on various lengths of analysis windows for multiple input scaling. Experiments showed that the structures with tagging different scales are complementary to each other on, i) detecting and ii) localizing sound events, therefore, an effective ensemble results in performance improvements for both tasks. The proposed model, an ensemble of the structures, achieved 0.4762 in the event-based F1-score and 0.7167 in the segment-based error rate on DCASE 2017 in development set. And it achieved 0.536 in the event-based F1-score and 0.66 in the segment-based error rate in evaluation set. Our model accomplished the 2nd place on audio tagging and the 1st place on sound event detection.

Index Terms— DCASE 2017, Weakly-supervised learning, Convolutional neural networks, Sound event detection

1. INTRODUCTION

Sound event detection (SED) aims to find sound objects and events from the audio content. SED has been studied in the contexts of various applications including acoustic scene analysis [1, 2], surveillance [3, 4, 5], health-care monitoring [6], and multimedia analysis [7, 8]. One of the applications is the SED for assisting car drivers which aims to help a driver to acknowledge the surroundings using audio content analysis.

Human drivers use cognitive abilities to recognize the surroundings such as the location and movements of nearby objects, e.g., cars and pedestrians. Obviously, a temporary decline in cognitive abilities reduces driving performance and increases the risk of accidents [9]. Object detection systems have been intensively studied over the years as an assistant system for human drivers. Especially, visual object detection systems have made significant improvements and deployed to the real system. However, visual sensors-based systems are heavily affected by the environmental condition such as lighting, shadows, and reflections, which limits the reliability of the system, therefore motivates using SED for the safer driving.

‘Conventional’ machine learning techniques have been proposed, e.g., using Mel-frequency cepstral coefficients (MFCCs) and non-negative matrix factorization-based features [10, 11, 12, 13]. Recently, deep learning-based methods such as recurrent neural networks (RNNs) [14] and convolutional neural networks (ConvNets) [15, 16] have been proposed. ConvNets showed the promising results in a number of computer vision tasks and have been actively adopted for audio content analysis such as SED [16] and music related tasks [17, 18].

An effective use of ConvNets on audio signal requires the specialized designs and domain-specific procedures. One of the design choices is the resolution/length of the audio input. Choosing the optimal size of audio input is usually task-specific and, to some extent, arbitrary. For example, a relatively long window (29-second) was used for music [19]. On the other hand, small window turned out to be more suitable for instrument identification [20]. SED also used a small length of window (below 100ms) as the optimal input size [14, 21]. A comprehensive approach is to use a multi-scale input and allows the network to learn to extract relevant information from inputs with various scale selectively [22, 23].

In this paper, we propose a sound event detection system that can recognize strong-labeled sound event from weakly-labeled data. This is a technical paper regarding our submission to the detection and classification of acoustic scene and events (DCASE) 2017 [24], *large-scale weakly supervised sound event detection for smart cars* which aims to simulate the SED problem in the real automotive environment by detecting 17 sound event categories including warning and vehicle sounds. Section 2 describes the proposed SED system. Section 4 shows and discusses the experiment results based on the results on the provided test set. Finally, Section 4 summarizes the final results of the competition.

2. PROPOSED SYSTEM

2.1. DCASE 2017 Dataset

The dataset of DCASE 2017 is a subset of AudioSet [25] that contains 17 warning and vehicle sounds that are related to the automotive environment. The dataset is divided into a training set and a test set, each with 51,172 and 488 audio clips. Each data sample may correspond to more than one sound event, and a binary decision is made for each class, i.e., the task is a multi-label classification problem. The audio signal is mono-channel and sampled at 44,100 Hz with a maximum duration of 10 seconds. The development set has only weak labels, i.e., only the presence of a given sound event is labeled without the exact time stamps while the test set is strongly labeled with both the categories of the existing sound events and their timestamps. There is a heavy class imbalance in the data set. The numbers of positive labels of the classes are between 180 and 25,077 and summarized in Fig. 1.

2.2. Audio Preprocessing

The amplitudes of the audio signal are normalized to the full-range. There are 10,785 signals that are shorter than 10-second, and they are zero-padded to equalize the length. 14 signals are excluded from the training set since they contain nothing. For separated-model, the

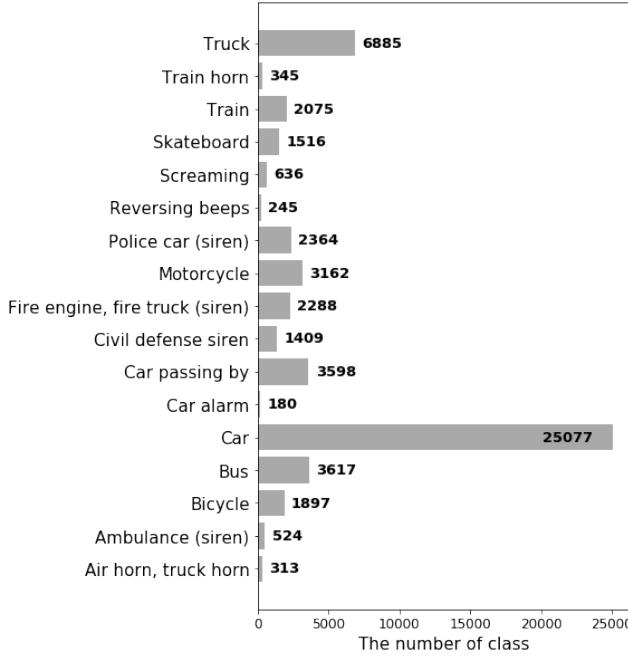


Figure 1: The class distribution in SED development set

waveform is segmented by 44,100 frames (1-second). It is chosen because 1-second is presumably long enough to contain a complete single sound event. The signal is converted into Mel-spectrogram with 2,048 FFT points (46 ms), 128 mel-bins, then its magnitudes are logarithmically mapped, i.e., $X \rightarrow \log 10X$. To simplify network design, we use the hop size of 431 and 460 for the global-input model and the separated-input model, respectively.

2.3. Background Noise Removal

An additional step in the audio preprocessing is performed to remove the background noise and enhance the target sound event in the Mel-spectrogram. For each Mel-spectrogram, the 128 median values are computed along the time axis and subtracted from it. It is known to have the effect of eliminating low-frequency background drift in continuous signals [26].

2.4. Network Architecture

The proposed system uses multiple models to predict audio events in a short-time segment. There are two networks: global-input model and separated-input model. It depends on whether the model uses the entire or a segmented audio clip. The system outline is illustrated in Fig. 2.

2.4.1. The global-input model

The details of the global-input model structure are illustrated in Fig. 3. It uses a 10-second waveform as input. It then converted to Mel-spectrogram with the shape of (1, 128, 1024) which correspond to the numbers of channels, mel-bins, and the frame. We use the homogeneous 2D (3×3) convolutional filters with the same number of feature map, 64, to form a fully convolutional network

structure, which is similar to [19]. The double_conv block and the max-pooling layer alternates, learning features while reducing the sizes of the feature maps. The double_conv block is a stack of two sets of a convolution layers, batch normalization, and a ReLU (rectified linear unit) activation function. The global average pooling layer follows after the last convolution block. The output layer is a densely-connected layer with the sigmoid activation function since it is a multi-class classification problem. The position of batch normalization follows the recent study in [27]. The weights are initialized using ‘He normal’ [28].

In the training, Adam optimizer [29] is used for an adaptive learning rate control. We allocate 15 % of the development set as a validation set and the final model is selected based on the validation set performance.

2.4.2. The separated-input model

As mentioned earlier, the separated-input model predicts the occurrence of sound events in a short audio segment. It uses a n -second segmented waveform as input. It then converted to Mel-spectrogram with shape of (1, 128, 96 \times n). The network structure is similar to that of the global-input model with changing the subsampling sizes as in Fig. 3.

Multiple models of the same structure are trained and correspond to inputs of 1, 2, 3, 4, and 5-second waveform with a 1-second sliding window. All the segments that make up the same clip are considered to have the same label. The other settings for training are the same as for the global-input model.

2.5. Predict Time Stamps

The proposed system is designed to predict the sound event probability of a given audio clip in seconds. This procedure primarily uses separated-input models. An input audio which has 10-second lengths is divided into pieces, and each segment is used in the separated-input model. Results from separated-input models are then converted to sound event occurrence probability matrix with the shape of (17 \times 10) which correspond to the kind of events, and the time in seconds. When the length of input segment is 1, each result from an input segment is considered to the probability at that time window. If the length of input audio is longer than 1-second, a specific one-second can be contained input segment multiple times. That is, for each one-second, the system can have a maximum n prediction (for n -second of input). We then average all possibilities and determine the existence of the event in that one-second. Once the sound event occurrence probability matrix has been made, we can easily mix multiple models to predict timestamps. The ensemble of the individual models is computed by averaging the probabilities of that time.

The global-input model is expected to have higher performance because it uses the entire audio clips with the correct label. However, timestamps cannot be predicted using the global-input model alone. We use the global-input model in two ways. Firstly, we use it in the same way with the separated-input model (*ClipAvg*). In this case, it is assumed that predictions from the global-input model are spread evenly across the 10-second time windows. The probability is then averaged together with other models above. Also, we use the global-input model as a sound event detector and detect the location using separated-input model only for clips where the event occurs (*ClipGate*).

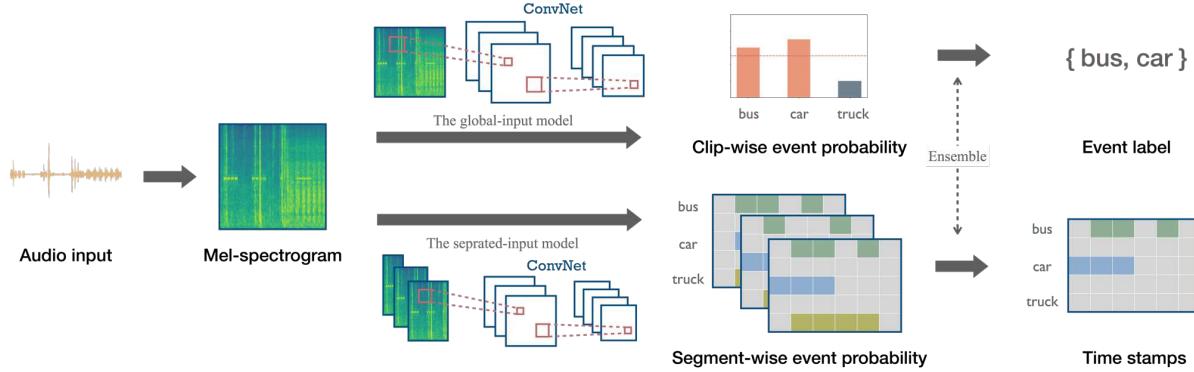


Figure 2: The overall system architecture of the proposed system. The global-input model takes the entire audio signal as an input and predicts the presence of an event in the signal while the separated-input model learns to find the presence of an event for given small segments. The final prediction is then given by ensembling both models probabilities.

| The global-input model (data shape) | | The separated-input model (data shape) | |
|-------------------------------------|----------------|--|----------------|
| Audio Input | (1, 44100) | Audio Input | (1, 44100 n) |
| Mel-spectrogram | (1, 128, 1024) | Mel-spectrogram | (1, 128, 96 n) |
| Double_Conv. block | | Double_Conv. block | |
| 4 x 4 Max-pooling | (64, 32, 256) | BatchNorm. | |
| Double_Conv. block | | ReLU | |
| 4 x 4 Max-pooling | (64, 8, 64) | Double_Conv. block | |
| Double_Conv. block | | 3 x 3 Conv. (64) | |
| 4 x 4 Max-pooling | (64, 8, 64) | BatchNorm. | |
| Double_Conv. block | | ReLU | |
| 2 x 4 Max-pooling | (64, 8, 16) | 3 x 3 Conv. (64) | |
| Double_Conv. block | | Double_Conv. block | |
| 2 x 4 Max-pooling | (64, 4, 4) | 4 x 3 Max-pooling | (64, 8, 8 x n) |
| Double_Conv. block | | Double_Conv. block | |
| GlobalAveragePooling | (1024) | 2 x 2 Max-pooling | (64, 4, 4 x n) |
| Output | (17) | Double_Conv. block | |
| | | 2 x 2 Max-pooling | (64, 2, 2 x n) |
| | | Double_Conv. block | |
| | | GlobalAveragePooling | (256) |
| | | Output | (17) |

Figure 3: The detailed network structure of the global-input model and the separated-input model.

2.6. Ensemble Method

We apply the ensemble selection method to find the optimal combination of learned models, expecting a better combination than the empirically chosen one. The ensemble selection algorithm method proposed by Caruana et al. [30] is used since we can apply it to the probability matrix that our system uses in the ensemble procedure. It works by repeating iterations and adding a model that maximize performance at that point.

We think that timestamped data is insufficient and fitting too much into small data makes model vulnerable. Therefore, ensemble selection is performed for the entire test data. We used F1 or ER as the performance metric to choose the weights specialized to each subtask. In addition, we used F1-ER as the performance metric, because the process that satisfies both tasks is expected to work as a kind of regularization.

2.7. Evaluation Measures

In DCASE challenge, the performances of classification and detection are evaluated by the event-based F1-score and the segmented-based error rate, respectively. For the classification of the whole

10-second audio signal, F1-score is used:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \quad (1)$$

, where P and R indicate the precision and recall respectively.

To evaluate the detected time stamps of sound events, the error rate (ER) is calculated in one-second segments over the entire test set. ER computes the percentage of all types of errors in every 1-second subsegment. For each data sample, ER is computed as below:

$$ER = \frac{S + D + I}{N} \quad (2)$$

, where N corresponds to the total number of event segments in the ground truth and S , D , I correspond to substitution/deletion/insertion errors.

3. EXPERIMENT RESULTS AND DISCUSSIONS

The experimental results are summarized in Table 1. For the audio tagging, the 10-second input model achieved the highest F-1 score. It suggests that for the global classification, using the entire audio in a single model and allowing the network to aggregate the prediction works better than manually aggregating the predictions from models with the shorter inputs. Among the models with various segment lengths, 3-second achieved the best performance, suggesting there exists the most suitable duration, probably depending on the types and intervals of the sound event.

Background subtraction improved the tagging performance of most systems, but a significant degradation was observed in the 5-second and 10-second input models, implying that our approach is not suitable for long time windows. The effect on the error rate is not clear, since the performance may be improved or decreased. However, by combining multiple separated-input models with or without BS, our system showed improved error rate than single models. The combined systems with 3, 4 s input models and 10-second input model (*ClipAvg*), show up to 0.7167 error. We assume that although the model with background subtraction shows similar error rates, behave differently, improving ensemble performance.

The result of ensemble selection is denoted in Fig. 4. The weight is used for the mean probability calculation. It could be interpreted as a kind of importance for each model. In this context, we

| Networks | Subtask A <i>F-1</i> | Subtask B <i>ER</i> |
|--------------------------------------|-------------------------|------------------------|
| Baseline (MLP) | .1310 | 1.0200 |
| 10-second input (w/BS) | .4745 (.3378) | - |
| 1s-segmented input (w/BS) | .4125 (.4373) | .7963 (.8362) |
| 2s-segmented input (w/BS) | .4229 (.4316) | .8071 (.8007) |
| 3s-segmented input (w/BS) | .4538 (.4561) | .7546 (.7610) |
| 4s-segmented input (w/BS) | .4304 (.4313) | .7633 (.7718) |
| 5s-segmented input (w/BS) | .4335 (.3588) | .8028 (.8431) |
| MeanProb of 5 models (w/BS) | .4408 (.4448) | .7667 (.7688) |
| MeanProb of 10 models | .4430 | .7475 |
| ClipAvg in 5 best models | .4762 | .7167 |
| ClipGate in 5 best models | .4745 | .7287 |
| *Ensemble selection (<i>F1</i>) | .5139 | .7477 |
| *Ensemble selection (<i>ER</i>) | .4831 | .7021 |
| *Ensemble selection (<i>F1-ER</i>) | .4885 | .7089 |

Table 1: SED performance on the test set. The performance of 12 single models is listed with multiple input scales and with background subtraction (BS). MeanProb model results using the mean probabilities of *ns*-segmented input models with and without BS. *ClipAvg* and *ClipGate* are the result using 5 best models (a 10-second input model and the 3 and 4-second input model with and without BS). Ensemble selection algorithm used the performance metric in a bracket. Note that the result with * used test label which should not be directly compared to other approaches.

| Networks | Subtask A <i>F-1</i> | Subtask B <i>ER</i> |
|-------------------------------------|-------------------------|------------------------|
| Baseline (MLP) | .182 | .930 |
| ClipAvg in 5 best models | .523 | .670 |
| ClipGate in 5 best models | .523 | .670 |
| Ensemble selection (<i>F1</i>) | .526 | - |
| Ensemble selection (<i>ER</i>) | - | .670 |
| Ensemble selection (<i>F1-ER</i>) | .521 | .660 |

Table 2: The results of our system in the DCASE 2017 competition.

can again guess the effect of background subtraction. The 5-second and 10-second models showed very low weights when using background subtraction, but the 1s-segmented models (w/BS) showed higher weights, although the lower performance. It suggests that the background subtraction works in a time window that is not too long.

4. DCASE2017 SUBMISSIONS AND RESULTS

Our submission 1, 2, and 4 used the same ensemble model for subtask A and B, only submission 3 used distinct models for subtask A and B. Details of submission are as follow: Submission 1 and Submission 2 are the ensembles of the top five models with the us-

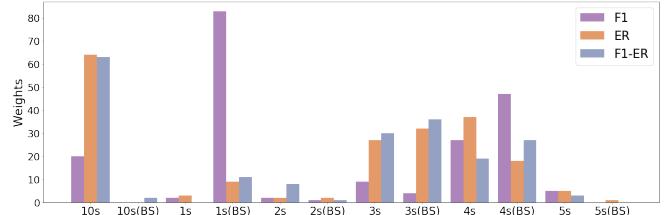


Figure 4: The weights of single models according to the performance metric of the ensemble selection.

ing of *ClipAvg* and *ClipGate*, respectively. Submission 3 and 4 are results of the ensemble selection method. We apply the ensemble selection to the entire 12 single models to find the best combinations of weights. Submission 3 is the result of ensemble selection over-fitting to test data specific to audio tagging and sound event detection, respectively. Submission 4 is the result of an ensemble selection suitable for both tasks.

There is no big performance difference between the submission when compared to the development set of the competition results. It suggests that the strategy that prevents overfitting into small data in ensemble method is valid, and the ensemble selection procedure does not significantly affect our system. In the DCASE competition, Submission 3 achieved second prize on audio tagging and Submission 4 achieved first prize on sound event detection.

5. CONCLUSIONS

In this paper, we used the ensemble of ConvNets with multiple analysis windows for the SED task. We segmented audio with duplicated labels to find the timestamps of weakly labeled data. The global-input model is superior to other single models when detecting the presence of the sound event in the entire audio clip, but there is a limitation to analyzing a small time window. Therefore, our system mixed the results from the global-input and separated-input models to predict the timestamps of the input audio and minimize errors using the ensemble selection methods.

We believe that there are potential improvements in our work. 1) In our experiments, the background subtraction is implemented on the entire time axis of the input audio, while it has more advantages in the short-time window. The ensemble of various models using short-time background subtraction can lead to improvements. 2) Experimental results show that the segmented-input is still useful for SED tasks, but all models in this study used the same structure regardless of the input shape. We think that using tailored network structures for analysis window in different lengths can improve the performance further.

6. ACKNOWLEDGEMENTS

We thank Keunwoo Choi for helpful comments that greatly improved the manuscripts. This research was supported by Korean government, MSIP provided financial support in the form of Bio & Medical Technology Development Program (2015M3A9D7066980).

7. REFERENCES

- [1] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumley, “Detection and classification of acoustic scenes and events: An ieee aasp challenge,” in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.
- [2] A. Mesaros, T. Heittola, and T. Virtanen, “Tut database for acoustic scene classification and sound event detection,” in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 1128–1132.
- [3] S. Ntalampiras, I. Potamitis, and N. Fakotakis, “On acoustic surveillance of hazardous situations,” in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 165–168.
- [4] C. Clavel, T. Ehrette, and G. Richard, “Events detection for an audio-based surveillance system,” in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005, pp. 1306–1309.
- [5] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, “Scream and gunshot detection and localization for audio-surveillance systems,” in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*. IEEE, 2007, pp. 21–26.
- [6] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, “Healthcare audio event classification using hidden markov models and hierarchical hidden markov models,” in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, 2009, pp. 1218–1221.
- [7] D. Zhang and D. Ellis, “Detecting sound events in basketball video archive,” *Dept. Electronic Eng., Columbia Univ., New York*, 2001.
- [8] M. Xu, C. Xu, L. Duan, J. S. Jin, and S. Luo, “Audio keywords generation for sports video analysis,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 2, p. 11, 2008.
- [9] K. J. Anstey, J. Wood, S. Lord, and J. G. Walker, “Cognitive, sensory and physical factors enabling driving safety in older adults,” *Clinical psychology review*, vol. 25, no. 1, pp. 45–65, 2005.
- [10] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real life recordings,” in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.
- [11] X. Zhuang, X. Zhou, M. A. Hasegawa-Johnson, and T. S. Huang, “Real-world acoustic event detection,” *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1543–1551, 2010.
- [12] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, “Sound event detection in multisource environments using source separation,” in *Machine Listening in Multisource Environments*, 2011.
- [13] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, “Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 151–155.
- [14] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6440–6444.
- [15] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.
- [16] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 559–563.
- [17] J. Schluter and S. Bock, “Improved musical onset detection with convolutional neural networks,” in *Acoustics, speech and signal processing (icassp), 2014 ieee international conference on*. IEEE, 2014, pp. 6979–6983.
- [18] W. Zhang, W. Lei, X. Xu, and X. Xing, “Improved music genre classification with convolutional neural networks.” in *INTERSPEECH*, 2016, pp. 3304–3308.
- [19] K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” in *The 17th International Society of Music Information Retrieval Conference, New York, USA*. International Society of Music Information Retrieval, 2016.
- [20] Y. Han, J. Kim, K. Lee, Y. Han, J. Kim, and K. Lee, “Deep convolutional neural networks for predominant instrument recognition in polyphonic music,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 1, pp. 208–221, 2017.
- [21] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.
- [22] J. Lee and J. Nam, “Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging,” *arXiv preprint arXiv:1703.01793*, 2017.
- [23] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *European Conference on Computer Vision*. Springer, 2014, pp. 346–361.
- [24] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “Dcase 2017 challenge setup: Tasks, datasets and baseline system.”
- [25] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *IEEE ICASSP*, 2017.
- [26] A. W. Moore Jr and J. W. Jorgenson, “Median filtering for removal of low-frequency background drift,” *Analytical chemistry*, vol. 65, no. 2, pp. 188–191, 1993.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.
- [28] ———, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

- [29] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [30] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, “Ensemble selection from libraries of models,” in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 18.

RARE SOUND EVENT DETECTION USING 1D CONVOLUTIONAL RECURRENT NEURAL NETWORKS

Hyungui Lim¹, Jeongsoo Park^{1,2}, Kyogu Lee², Yoonchang Han¹

¹ Cochlear.ai, Seoul, Korea

² Music and Audio Research Group, Seoul National University, Seoul, Korea
 {hglim, jspark, ychan}@cochlear.ai, kglee@snu.ac.kr

ABSTRACT

Rare sound event detection is a newly proposed task in IEEE DCASE 2017 to identify the presence of monophonic sound event that is classified as an emergency and to detect the onset time of the event. In this paper, we introduce a rare sound event detection system using combination of 1D convolutional neural network (1D ConvNet) and recurrent neural network (RNN) with long short-term memory units (LSTM). A log-amplitude mel-spectrogram is used as an input acoustic feature and the 1D ConvNet is applied in each time-frequency frame to convert the spectral feature. Then the RNN-LSTM is utilized to incorporate the temporal dependency of the extracted features. The system is evaluated using DCASE 2017 Challenge Task 2 Dataset. Our best result on the test set of the development dataset shows 0.07 and 96.26 of error rate and F-score on the event-based metric, respectively. The proposed system has achieved the 1st place in the challenge with an error rate of 0.13 and an F-Score of 93.1 on the evaluation dataset.

Index Terms— Rare sound event detection, deep learning, convolutional neural network, recurrent neural network, long short-term memory

1. INTRODUCTION

Auditory information helps people recognize their surroundings. In an emergency situation, auditory information becomes even more important as it allows nearby people to react effectively and quickly. Rare sound event detection (RSED) is a set of algorithms that aim to automatically detect certain emergency sounds with high accuracy. As part of such efforts, task 2 in Detection and Classification of Acoustic Scenes and Events (DCASE) 2017 is organized, which asks to identify the presence of three target events – baby crying, glass breaking, and gunshot – and their corresponding onset time.

According to its necessity, sound event detection has been studied extensively in recent years. Some studies aim to recognize multiple sound events that occur simultaneously (polyphonic) [1], [2], [3], [4] while others detect one prominent event among multiple candidates (monophonic) [1], [5], [6], [7]. In the case of an emergency, monophonic detection is considered as more suitable approach since the emergency-related sounds scarcely occur simultaneously. Hence, detecting single type of sound with high accuracy is more valuable in such cases.

In terms of algorithms, a number of conventional research efforts have applied machine learning algorithms such as hidden Markov model (HMM) [5], non-negative matrix factorization (NMF) [8], [9], support vector machine (SVM) [10], and random forest [7]. Recent approaches use deep learning-based methods us-

ing deep neural network (DNN) [2], convolutional neural network (ConvNet) [11], recurrent neural network (RNN) [3], [12], and convolutional recurrent neural network (CRNN) [4].

In this paper, we apply a hybrid neural network of 1D ConvNet and RNN with long short-term memory units (LSTM). Frame-wise log-amplitude mel-spectrogram is fed into our proposed model, and the model returns the output for every incoming sequence. It makes possible to estimate a relatively accurate onset time by maintaining small temporal resolution. This single model is applied to compute event probability for all three target events. We also conduct experiments with different fixed length input (timestep) and different set of data mixtures to find the best hyperparameters. We confirm that our proposed method shows significant improvement in the test set of TUT rare sound events 2017 dataset compared to the baseline.

The rest of the paper is organized as follows. Section 2 describes the proposed method. Section 3 shows the experimental results with TUT rare sound events 2017 dataset. Conclusions are presented in Section 4. Algorithm description for DCASE 2017 submission is presented in Section 5.

2. PROPOSED METHOD

Fig. 1 shows an overall framework of our proposed method which consists of four parts: 1) extracting log-amplitude mel-spectrogram from audio, 2) converting spectral feature with 1D ConvNet, 3) incorporating temporal dependency with RNN-LSTM, and 4) determining the presence and the onset time of audio event with post-processing.

2.1. Log-amplitude mel-spectrogram

Mel-spectrogram is a 2D time-frequency representation extracted from an audio signal. It has been recognized as a useful feature and has been used for various deep learning-based audio analyses. Unlike normal spectrogram, the frequency components are filtered with log-scale filter banks to imitate the function of human ears. It leads compression of high frequency components and helps to concentrate more on low frequency components.

Considering these advantages of using mel-spectrogram, we also use it as the input feature of our proposed method. To extract this feature, a window is applied to an audio signal with a size of 46 ms, being overlapped with half size of the window. We also apply 128 mel-filter banks on the spectrum of each frame and take logarithm on the amplitude. The mel-spectrogram is divided into a chunk with the size of a timestep (τ), and fed into our network.

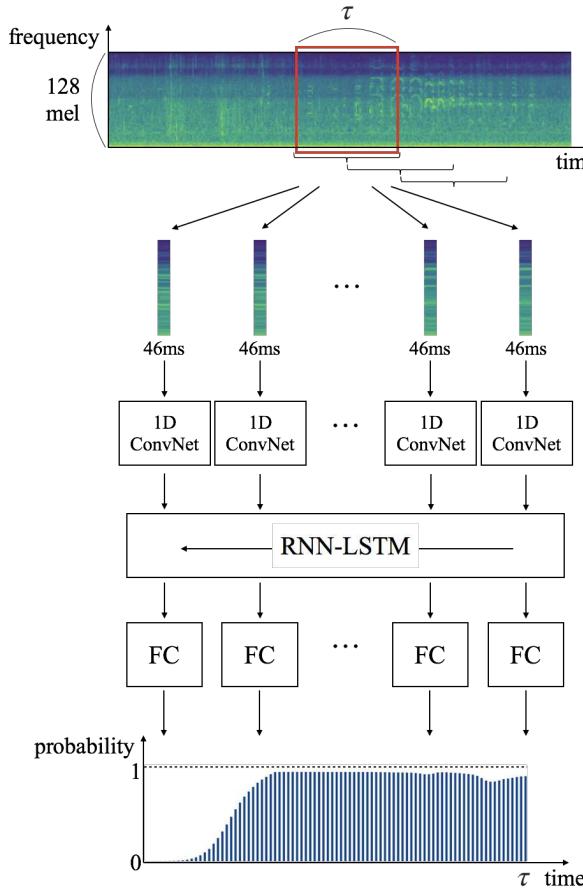


Figure 1: Overall framework of the proposed method.

2.2. 1D ConvNet

Many audio contents analysis studies that use 2D input features such as spectrogram, mel-spectrogram, and mel-frequency cepstral coefficients (MFCC) apply 2D ConvNet [13], [14], which is often used for image content analysis. It focuses on the spectral and temporal locality from the audio features to extract meaningful information. However, 2D ConvNet-based methods analyze the audio in chunk-level rather than frame-level. Since the precise estimation of onset time is necessary for this task, we apply spectral-side 1D ConvNet that enables frame-level investigation.

The 1D ConvNet step consists of 1D convolution layer, Batch Normalization (BN) [15] process, and pooling layer. Fig. 2 shows the concept of the 1D convolution layer and the max-pooling layer. The filter size of the convolution layer is set to 32, and 128 filters are used in total. Therefore, 128 outputs each contains 97 ($128 - 32 + 1$) elements are produced when single frame of the mel-spectrogram (128 frequency bin) is fed into the 1D convolution layer. In the next step, BN is applied on feature map outputs so that they maintain the mean close to 0 and the standard deviation close to 1. After that, rectified linear unit (ReLU) [16] is applied as an activation function. Finally, max-pooling with the size of 97 is applied to each output to extract representative value. Dropout is also applied with the value of 0.3 at the end of ConvNet to prevent overfitting.

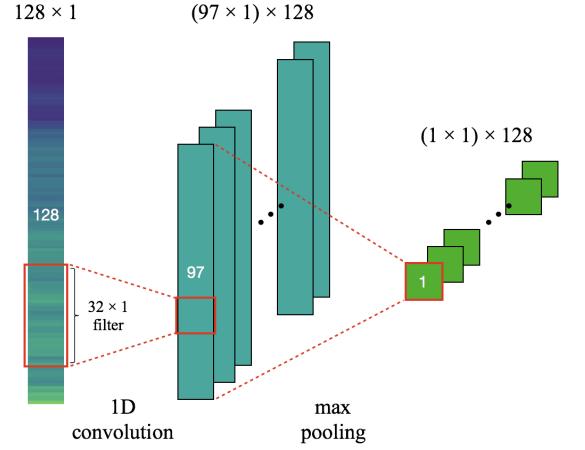


Figure 2: 1D-ConvNet structure for frame-wise feature extraction. The output feature size is same as the input mel-band size at 128.

2.3. RNN-LSTM

RNN has proven to be a powerful model for identifying sequential information such as speech recognition [17] and hand writing recognition [18]. In particular, RNN-LSTM is a well-known deep learning model that prevents vanishing gradient that disturbs long-term sequence learning [19]. Thus, we use RNN-LSTM here to incorporate the temporal dependency of the extracted features.

Here, we use two RNN layers each contains 128 LSTM units. Unlike general studies using forward or bidirectional RNN-LSTM, we apply unidirectional backward RNN-LSTM. This is because the information after the onset of an event is appeared to be more important for the precise onset detection compared to the information before the onset. According to our experiment, this unidirectional backward analysis has shown better performance than the other methods.

Fig. 3 shows the processing structure inside the RNN-LSTM step. The features extracted from the ConvNet ($x_t, x_{t+1}, \dots, x_{t+\tau-1}$) are fed into the networks that passes it through the layers. Note that the 128-dimensional output vectors ($z_t, z_{t+1}, \dots, z_{t+\tau-1}$) are obtained for each frame. We use hyperbolic tangent (tanh) as an activation function and apply a dropout rate of 0.3 for all RNN-LSTM layers.

2.4. Fully connected layer and post-processing

The returned features from the RNN-LSTM layer are fed into a fully connected layer (FC) that contains 128 hidden units. Similar to the previous 1D ConvNet step, BN and ReLU are applied as a normalization function and activation function, respectively. The updated features are then forwarded to a time-distributed output layer with one sigmoid unit, of which output represents the probability of presence of the target sound event. As a result, the probability values are calculated for each frame of the mel-spectrogram during the timestep.

In order to obtain the probability sequence of an entire audio clip at the test stage, sliding ensemble method is utilized. As the probability values are calculated in each chunk with our trained model, this method combines the entire probabilities by sliding the prediction chunk with a hop size of one frame (23 ms) and aver-

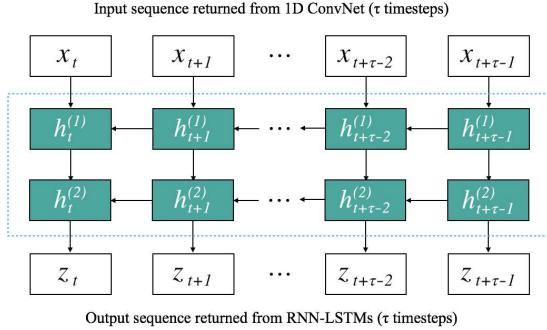


Figure 3: RNN-LSTM structure for sequential learning. Two hidden RNN-LSTM layers (h) are applied in a backward direction. They return the output (z) for all inputs (x) during the timestep (τ).

aging the probabilities of the indices where the value exists. An illustration of this method is shown in Fig. 4.

Fig. 5 shows an example of the determination of event presence and prediction of its corresponding onset from probability sequence. In order to determine the presence of a sound event, hard thresholding scheme with empirical assumptions is used. If the maximum value in the probability sequence is greater than 0.8 (0.5 for ‘gunshot’), the audio clip is considered to include the target event. To find the onset time of the sound event, we select the first index of the value greater than 0.5 among the 50 (200 for ‘baby crying’) preceding frames from the maximum value.

3. PERFORMANCE EVALUATION

3.1. Dataset

For the task 2 of DCASE 2017, ‘TUT Rare Sound Events 2017’ dataset is provided, which consists of isolated sound events for each target class and recordings of everyday acoustic scenes to serve as background. In the dataset, three target sound events are considered: ‘baby crying’, ‘glass breaking’, and ‘gunshot’. The background audio set contains recordings from 15 different audio scenes, which are a part of ‘TUT Acoustic Scenes 2016’ dataset.

The source code for creating a combination of different event-to-background is also given along with the audio recordings. Using the code, we can generate training data with different parameters such as number of mixtures, event-to-background ratio (EBR) and event occurrence probability. Annotations for the mixtures including the name of the target event and its temporal position are also produced automatically. We have created 4 sets of mixtures (S_1, S_2, S_3, S_4). Each set consists of 15,000 audio clips (5,000 per event class), generated with EBRs of -6, 0, 6dB and an event occurrence probability of 0.5. All mixtures are created as a 30-seconds monaural audio with 44,100 Hz and 24 bits. For the training, these mixtures are randomly divided into a train set and validation set at 8 to 2 ratios. Pre-combined test set which contains 1,500 audio clips (500 per event class) is used at the test process.

3.2. Deep learning setup

In the training stage, after the input chunks are fed into the model and converted to probability values, errors between the predicted values and correct values (0 or 1) are calculated with a binary cross

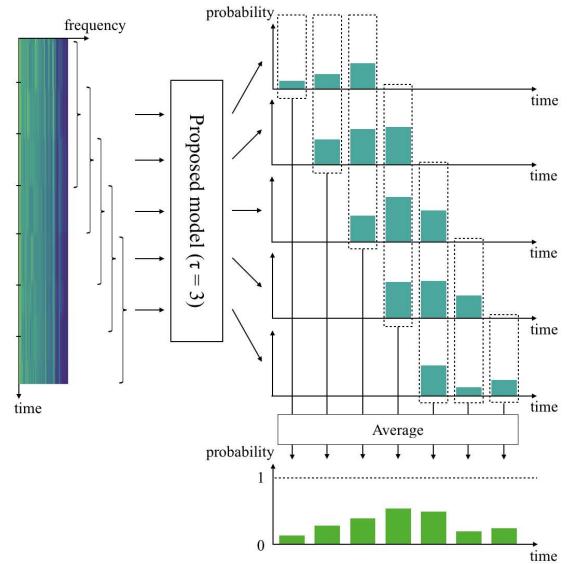


Figure 4: An example of a sliding ensemble method using a model with a timestep of 3. The predicted probability sequences from each sliding window are combined into a single probability sequence in this way.

entropy as a loss function. To optimize the loss, we apply adaptive momentum (Adam) as an optimizer and the size of a mini-batch is set to 256. The learning rate is initially set to be 0.001 and decayed over each epoch with decaying factor 0.01 of learning rate. Learning is stopped early when a validation loss has stopped improving for 10 epochs.

3.3. Evaluation metric

We evaluate our method using event-based metric [20], which requires calculation of true positives (TP), false positives (FP), and false negatives (FN). If the system’s output accurately predicts the presence of an event and its onset, it is computed as TP. The onset time detection is considered true only when it is predicted within the range of 500 ms of the actual onset time. Meanwhile, FP indicates that the system incorrectly detects the presence of an event when there is no event. If the system output misses the event, it is considered an FN. These metrics are used to calculate error rate and F-score in the final step, which are mathematically defined as

$$ER = \frac{FN + FP}{N} \quad (1)$$

$$F = \frac{2PR}{P + R}, \quad (2)$$

where N denotes the total number of samples in the evaluation dataset, and P and R denote precision and recall, defined as below.

$$P = \frac{TP}{TP + FP} \quad (3)$$

$$R = \frac{TP}{TP + FN} \quad (4)$$

These evaluation metrics were computed using sed_eval toolbox [20] which is given in the task.

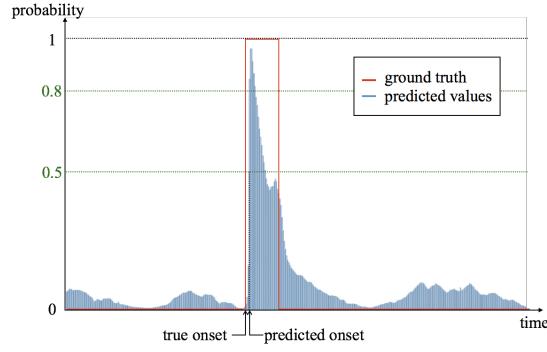


Figure 5: An example of applying a threshold to detect the presence and the onset time of an event.

3.4. Result and discussion

We have compared the experimental results by changing the set of mixtures and timestep. Then we have selected models that show relatively high-performance, followed by the ensemble method to combine them. Table 1 shows the types of models combined for an ensemble method and their mixing weights. p_a^b denotes the probability value calculated by the trained model using a mixture set of S_a and a timestep size of b . Table 2 shows the event based error rate and F-score results on the test set of the development dataset. Both results of our proposed method show better performance than the DCASE 2017 baseline system.

The result shows that our method achieves the best performance on ‘glass breaking’ followed by ‘baby crying’ and ‘gunshot’. In the case of ‘glass breaking’, the frequency component at the moment when the glass breaks is clear compared to the background sound. Therefore, the model with short timestep was effective for this class. In the case of ‘baby crying’, since the length of the sound event is longer than the others, it was better to apply a relatively longer timestep. For the same reason, a long frame range to find the onset time worked better as mentioned in Section 2.4. Still, there existed misclassified events such as bird sound which has similar tonality to the baby crying. In the case of ‘gunshot’, relatively short timestep was used because similar to ‘glass breaking’, the moment of the gunshot is obvious as it sounds like an impulse. However, since the gunshot sound has a lot of reverberations, it seemed to require slightly longer timestep than ‘glass breaking’. The result shows that the performance of ‘gunshot’ detection is worse than the others because the sounds vary according to the gun type. For that reason, several misclassifications are observed on impulse-like sound events such as footprint and metal door-closing sounds. Regardless of the event class, the onset time was relatively accurate even if the model estimated the presence of event incorrectly.

Overall, increasing the amount of training data by synthesizing various mixtures seemed more effective for the performance than adjusting the parameters of the model. The experimental result showed meaningful performance improvement when we boosted the audio clips 10 times more than the given mixture set.

4. CONCLUSION

In this paper, we have presented a rare sound event detection system using 1D convolutional recurrent neural networks. It has shown

Table 1: Selected models and their weights for an ensemble method.

| Event | Ensemble method | |
|----------------|---|-------------------------------|
| | $(p_1^{(100)} + 2p_2^{(50)} + p_3^{(50)} + p_3^{(100)}) / 5$ | $(p_1^{(5)} + p_3^{(5)}) / 2$ |
| Glass breaking | $(p_1^{(5)} + p_3^{(5)}) / 2$ | |
| Gunshot | $(2p_1^{(14)} + p_1^{(50)} + p_3^{(10)} + p_4^{(10)} + p_4^{(20)}) / 6$ | |

Table 2: Performance of baseline and proposed system in the development set.

| | ER | | F-score | |
|----------------|----------|-------------|----------|-------------|
| | baseline | proposed | baseline | proposed |
| Baby crying | 0.67 | 0.05 | 72.0 | 97.6 |
| Glass breaking | 0.22 | 0.01 | 88.5 | 99.6 |
| Gunshot | 0.69 | 0.16 | 57.4 | 91.6 |
| Overall | 0.53 | 0.07 | 72.7 | 96.3 |

Table 3: Performance of baseline and proposed system in the evaluation set.

| | ER | | F-score | |
|----------------|----------|-------------|----------|-------------|
| | baseline | proposed | baseline | proposed |
| Baby crying | 0.80 | 0.15 | 66.8 | 92.2 |
| Glass breaking | 0.38 | 0.05 | 79.1 | 97.6 |
| Gunshot | 0.73 | 0.19 | 46.5 | 89.6 |
| Overall | 0.64 | 0.13 | 64.1 | 93.1 |

promising results on IEEE DCASE 2017 Task 2. We believe that three key factors in the proposed method have contributed to the performance improvement. The first factor is frame-wise detection of the model which is effective in finding the precise onset time. The second is the internal/external ensemble method used in Section 2.4 and Section 3.4 which reduces various noises. The last and the biggest contributor to the performance improvement is a large amount of synthesized data consists of various mixtures.

5. DCASE 2017 SUBMISSION

We applied the same model settings of the development set to the evaluation set. For the final submission, we selected four different results by applying four different threshold set of event presence (mentioned in Section 2.4). We used the threshold set with 0.8 / 0.8 / 0.5 ('baby crying' / 'glass break' / 'gunshot') for submission 1, 0.7 / 0.7 / 0.5 for submission 2, 0.6 / 0.6 / 0.5 for submission 3, and 0.5 / 0.5 / 0.5 for submission 4. The error rate and F-score was 0.13 / 93.1 for submission 1, 0.13 / 93.0 for submission 2, 0.15 / 92.2 for submission 3, and 0.17 / 91.4 for submission 4. We achieved the best result with submission 1 and the results of each class from this submission are shown in Table 3.

6. ACKNOWLEDGEMENT

This research was supported by Korean government, MSIP provided financial support in the form of Bio & Medical Technology Development Program (2015M3A9D7066980).

7. REFERENCES

- [1] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, “Context-dependent sound event detection,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, p. 1, 2013.
- [2] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.
- [3] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6440–6444.
- [4] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *arXiv preprint arXiv:1702.06286*, 2017.
- [5] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real life recordings,” in *Signal Processing Conference, 2010 18th European*. IEEE, 2010, pp. 1267–1271.
- [6] C. V. Cotton and D. P. Ellis, “Spectral vs. spectro-temporal features for acoustic event detection,” in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*. IEEE, 2011, pp. 69–72.
- [7] H. Phan, M. Maaß, R. Mazur, and A. Mertins, “Random regression forests for acoustic event detection and classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 20–31, 2015.
- [8] J. F. Gemmeke, L. Vuggen, P. Karsmakers, B. Vanrumste, et al., “An exemplar-based nmf approach to audio event detection,” in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.
- [9] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, “Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 151–155.
- [10] A. Temko and C. Nadeu, “Classification of acoustic events using svm-based clustering schemes,” *Pattern Recognition*, vol. 39, no. 4, pp. 682–694, 2006.
- [11] A. Gorin, N. Makhzhanov, and N. Shmyrev, “Dcase 2016 sound event detection system based on convolutional neural network,” *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events*, 2016.
- [12] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, “A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1996–2000.
- [13] 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.
- [14] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.
- [15] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [16] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [17] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” *arXiv preprint arXiv:1402.1128*, 2014.
- [18] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2009.
- [19] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” 1999.
- [20] A. Mesaros, T. Heittola, and T. Virtanen, “Metrics for polyphonic sound event detection,” *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

DCASE 2017 CHALLENGE SETUP: TASKS, DATASETS AND BASELINE SYSTEM

Annamaria Mesaros¹, Toni Heittola¹, Aleksandr Diment¹, Benjamin Elizalde², Ankit Shah², Emmanuel Vincent³, Bhiksha Raj², Tuomas Virtanen^{1}*

¹ Tampere University of Technology, Laboratory of Signal Processing, Tampere, Finland

{annamaria.mesaros, toni.heittola, aleksandr.diment, tuomas.virtanen}@tut.fi

² Carnegie Mellon University, Department of Electrical and Computer Engineering,

& Department of Language Technologies Institute, Pittsburgh, USA

bmartin1@andrew.cmu.edu, aps1@andrew.cmu.edu, bhiksha@cs.cmu.edu

³ Inria, F-54600 Villers-lès-Nancy, France, emmanuel.vincent@inria.fr

ABSTRACT

DCASE 2017 Challenge consists of four tasks: acoustic scene classification, detection of rare sound events, sound event detection in real-life audio, and large-scale weakly supervised sound event detection for smart cars. This paper presents the setup of these tasks: task definition, dataset, experimental setup, and baseline system results on the development dataset. The baseline systems for all tasks rely on the same implementation using multilayer perceptron and log mel-energies, but differ in the structure of the output layer and the decision making process, as well as the evaluation of system output using task specific metrics.

Index Terms— Sound scene analysis, Acoustic scene classification, Sound event detection, Audio tagging, Rare sound events, Weak Labels

1. INTRODUCTION

Sounds carry a large amount of information about our everyday environment and physical events that take place in it. Humans are very skilled in perceiving the general characteristics of the sound scene around them, whether it is a busy street, a quiet park or a quiet office environment, and recognizing individual sound sources in the scenes, such as cars passing by, birds, or footsteps. Developing computational methods to automatically extract this information has huge potential in several applications, for example searching for multimedia based on its audio content [1], making context-aware mobile devices [2], robots, cars, etc., and intelligent monitoring systems [3, 4] to recognize activities using acoustic information. However, a significant amount of research is still needed to reliably recognize sound scenes and individual sound sources in real-life soundscapes, where multiple sounds are present, often simultaneously, and distorted by the environment.

Building up on the success of the previous editions, DCASE 2017 Challenge supports the development of computational scene and event analysis methods by comparing different approaches using common publicly available datasets. The continuous effort in this direction will set another milestone of development, and anchor the current performance for further reference. The challenge consists of four tasks: acoustic scene classification, detection of rare

sound events, sound event detection in real-life audio, and large-scale weakly supervised sound event detection for smart cars.

Acoustic scene classification is a prominent topic in environmental sound classification. It is defined as recognition of the environment in which a recording has been made, relying on the assumption that an acoustic scene, as a general characterization of a location or situation, is distinguishable from others based on its general acoustic properties. It has been present as a task in DCASE 2013 [5] and DCASE 2016 [6], and has been approached in a variety of ways. A review of the features and classifiers used for it is presented in [7], with features including the well-known mel-frequency cepstral coefficients [2, 8] or more specialized features such as histograms of sound events [9] or histogram of gradients learned from time-frequency representations [10], and acoustic models such as hidden Markov models (HMMs) [2], Gaussian mixture models (GMMs) [8] or support vector machines (SVMs) [10, 11]. More recently, the emergence of methods using deep learning is noticeable, with many of the submitted systems for DCASE 2016 being based on various types of deep neural networks (DNNs) [6].

Sound event detection is defined as recognition of individual sounds in audio, involving also estimation of onset and offset for distinct sound event instances, possibly for multiple sound classes. It assumes that similar sounds can be represented as a single class, and as such this class is sufficiently different from other sound classes to allow recognition. The most used features for sound event detection are mel-scale representations, namely cepstral coefficients or log energies [12, 13, 14], and they are used with various machine learning methods, including HMMs [12], non-negative matrix factorization (NMF) [15, 16], random forests [11], and DNNs [14, 17].

Sound event detection in real-life audio presents many difficulties for automatic methods, such as the inherent acoustic variability of the sounds belonging to the same sound event class, or other sounds overlapping with the sound event of interest. In some situations, the target sound events are very rare, imposing additional burden on detection systems to avoid false detections. DCASE 2017 Challenge addresses rare sound events and highly overlapping sounds through two separate tasks: detection of rare sound events, sound event detection in real-life audio.

Sound recordings are shared on the Internet on a minute-by-minute basis. These recordings are predominantly videos and constitute the largest archive of sounds we've ever seen. Most of their acoustic content is untagged; hence automatic recognition of sounds within recordings can be achieved by sound event detection.

*AM, TH and TV received funding from the European Research Council under the ERC Grant Agreement 637422 EVERYSOUND.

However, most of the literature and the previous two iterations of DCASE focus on audio-only recordings and supervised approaches, by which the training and test data are annotated with strong labels (including precise timestamps). Collecting such annotations hardly scales to the number of web videos and sound classes. Therefore, we argue that there is a need for semi-supervised approaches that are trained and evaluated with weak labels (not including precise timestamps). Current literature has shown potential using unsupervised [18, 19, 20] and semi-supervised approaches [21, 22] some of them employing weak labels [23]. Success in this task would complement other modalities for video content analysis.

This paper presents in detail the DCASE 2017 Challenge tasks. For each task we provide the task definition, information about the dataset, the task setup and baseline system, and baseline results on the development dataset. The baseline systems for all tasks rely on the same implementation and use the same features and techniques; they differ in the way they handle and map the input data to target outputs, as this is application specific and was chosen according to the task.

2. CHALLENGE SETUP

The challenge provided the potential participants with four tasks, with publicly available datasets and a baseline system for each task. Challenge submission consisted in system output(s) formatted according to the requirements. In addition, participants were required to submit a technical report containing the description of the system(s) in sufficient detail, to allow the community to compare and understand all submissions. The timeline of the challenge is presented in Table 1, and the general organization of the datasets and baseline systems presented in detail in the following sections.

2.1. Datasets

A *development dataset* was provided for each task when the challenge was launched, consisting of predefined *training* and *test* sets (for some tasks in a cross-validation folds format) to be used during system development. A separate dataset, referred to as *evaluation dataset*, was kept for evaluation of the developed systems. The development datasets consist of audio material and associated reference annotations in a task-specific format, and an experimental setup for reporting system performance on the development dataset. The organizers' recommendation was to use the provided experimental setup, in order to allow a direct comparison between submissions. Access to the datasets was provided through the challenge website¹.

As general rules applicable for all tasks, participants were not allowed to use external data for system development, with datasets from a different task considered as external data. However, manipulation of the provided training and development data was allowed, for augmentation without use of external data (e.g. by mixing data sampled from a probability distribution function or using techniques such as pitch shifting or time stretching).

The evaluation datasets were provided as audio only, without reference annotations, shortly before the challenge submission deadline. Participants were required to run their systems on this data and submit the system outputs to the organizers for evaluation. Participants were not allowed to make subjective judgments of the evaluation data, nor to annotate it. The use of the evaluation dataset

Table 1: Challenge timeline

| | |
|---------------------------------|----------------|
| Release of development datasets | 21 Mar 2017 |
| Release of evaluation datasets | 30 June 2017 |
| Challenge submission | 31 July 2017 |
| Publication of results | 15 Sept 2017 |
| DCASE 2017 Workshop | 16-17 Nov 2017 |

to train the submitted system was also forbidden. Reference annotations for the evaluation data were only available to the organizers, therefore they were responsible with performing the evaluation of the results according to the metrics for each task.

2.2. Baseline system

A baseline system was provided, with a common implementation for all tasks. The system consists of a basic approach that was tailored to each task. Its purpose is to provide a comparison point for the participants while developing their systems. The performance of the baseline system on the development set is provided for each task. When run with the default parameters, the system downloads the needed dataset and outputs the task-specific results [24].

The implementation is based on a multilayer perceptron architecture (MLP) and uses log mel-band energies as features. The features are calculated in frames of 40 ms with a 50% overlap, using 40 mel bands covering the frequency range 0 to 22050 Hz. The feature vector was constructed using a 5-frame context, resulting in a feature vector length of 200. The MLP consists of two dense layers of 50 hidden units each, with 20% dropout. The network is trained using Adam algorithm for gradient-based optimization [25]; training is performed for maximum 200 epochs using a learning rate of 0.001, and uses early stopping criteria with monitoring started after 100 epochs and a 10 epoch patience. The output layer of the network is task specific, and will be described in the corresponding section. The network is trained using the aforementioned features, and the learning target is presented according to the implemented task. The baseline system also includes evaluation of the system outputs using a specific metric for each task.

The baseline system was implemented using Python, using Keras for machine learning. It has all needed functionality for dataset handling, storing and accessing features and models, and evaluating the results, and allows straightforward adaptation and modification of the various involved steps. Participants were allowed and encouraged to build their system on top of the given baseline system.

3. TASK 1: ACOUSTIC SCENE CLASSIFICATION

The goal of acoustic scene classification is to classify a test recording into one of the provided predefined classes that characterizes the environment in which it was recorded for example “park”, “home”, “office”, as illustrated in Fig. 1.

The dataset provided for this task is TUT Acoustic Scenes 2017, which consists of TUT Acoustic Scenes 2016 [26] as the development set, and a newly recorded evaluation set. The main difference is that for this edition of the challenge, the original recordings of 3-5 minutes length were split into 10 s long segments which were provided in individual files and considered as independent. Shorter audio segments provide less information to the system for the decision making process, thus increasing the task difficulty from the

¹<http://www.cs.tut.fi/sgn/arg/dcase2017/>

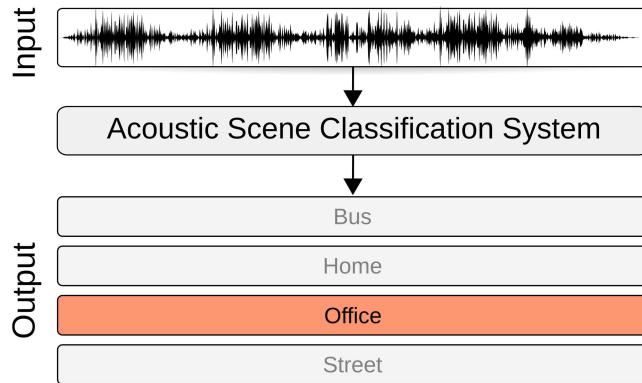


Figure 1: A schematic illustration of the acoustic scene classification addressed in Task 1.

Table 2: Class-wise accuracy of the baseline system for Task 1.

| Acoustic scene | Development set | Evaluation set |
|------------------|-----------------|----------------|
| | Acc. (%) | Acc. (%) |
| Beach | 75.3 | 40.7 |
| Bus | 71.8 | 38.9 |
| Cafe/Restaurant | 57.7 | 43.5 |
| Car | 97.1 | 64.8 |
| City center | 90.7 | 79.6 |
| Forest path | 79.5 | 85.2 |
| Grocery store | 58.7 | 49.1 |
| Home | 68.6 | 76.9 |
| Library | 57.1 | 30.6 |
| Metro station | 91.7 | 93.5 |
| Office | 99.7 | 73.1 |
| Park | 70.2 | 32.4 |
| Residential area | 64.1 | 77.8 |
| Train | 58.0 | 72.2 |
| Tram | 81.7 | 57.4 |
| Overall | 74.8 | 61.0 |

previous edition. This length is regarded as challenging for both human and machine recognition, based on the study in [2]. A detailed description of the data recording and annotation procedure can be found in [26].

The acoustic scene classes considered in this task were: bus, cafe/restaurant, car, city center, forest path, grocery store, home, lakeside beach, library, metro station, office, residential area, train, tram, and urban park. A cross-validation setup containing four folds was provided, splitting the available audio material in the development set such that all segments obtained from the same original recording are included to one side of the learning algorithm, either training or test. For each class, the development set contains 312 segments of 10 seconds (52 minutes of audio material).

For this task, the baseline system was tailored to a multi-class single label classification setup, with the network output layer consisting of softmax type neurons representing the 15 classes. The classification decision was based on the output of the neurons, which can be active only one at a time. Frame-based decisions were combined using majority voting to obtain a single label per classified segment. The system performance was measured using accuracy, defined as the ratio between the number of correct system outputs and the total number of outputs [27]. The system was trained

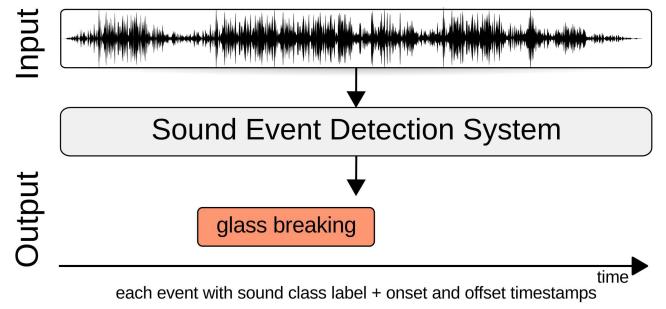


Figure 2: A schematic illustration of the detection of rare sound events addressed in Task 2.

and tested using the provided four fold cross-validation setup, obtaining an average classification accuracy of 73.8% on the development set and 61.0% on the evaluation set. Class-wise accuracy is presented in Table 2. Ranking of the systems submitted for the challenge is done using classification accuracy.

4. TASK 2: DETECTION OF RARE SOUND EVENTS

Task 2 focused on the detection of rare sound events, as illustrated in Fig. 2. The audio material used in this task consists of artificially created mixtures, allowing the creation of many examples at different event-to-background ratios. Here, “rare” refers to target sound events occurring at most once within a half-minute recording. For each of the three target sound event classes, a separate system is to be developed to detect the temporal occurrences of these events.

The provided dataset consists of source files for creating mixtures of rare sound events with background audio, as well as a set of readily generated mixtures and the so-called recipes according to which the mixtures were created. Additionally, a software package was provided, which performs further generation of additional mixture recipes and generates the audio mixtures.

The background recordings originate from the TUT Acoustic Scenes 2016 development dataset [26], with the exception of segments naturally containing the target class events and interference from mobile phone, which were removed. The rare sound events are of the following classes: baby cry (106 training, 42 test instances, mean duration 2.25 s), glass break (96 training, 43 test instances, mean duration 1.16 s) and gun shot (134 training, 53 test, mean duration 1.32 s). The recordings were downloaded from freesound.org through the API with python wrapper². In the “source” part of this dataset, these recordings were presented in their original form, and were accompanied by the added annotations of the temporal occurrences of isolated events.

We isolated the target sound events from the full-length recordings acquired from freesound.org that may consist of the actual event, silence regions and background noise in the following manner. First, a semi-supervised segmentation [28] was performed using an SVM trained to classify high-energy and low-energy frames. The active segments were then obtained with a dynamic threshold computed as a weighted average of top 10% and lower 10% of the onset probability values over all the analysis frames of a recording. Thereupon, a human annotator listened to each obtained segment, and segments containing irrelevant events were discarded (baby coughs, unrealistically sounding gun shots such as laser guns

²<https://github.com/xavierfav/freesound-python-tools>

Table 3: Baseline system results for Task 2, event-based metrics.

| Event Class | Development set | | Evaluation set | |
|-------------|-----------------|-------------|----------------|-------------|
| | ER | F-score (%) | ER | F-score (%) |
| Baby cry | 0.67 | 72.0 | 0.80 | 66.8 |
| Glass break | 0.22 | 88.5 | 0.38 | 79.1 |
| Gun shot | 0.69 | 57.4 | 0.72 | 46.5 |
| Average | 0.53 | 72.7 | 0.63 | 64.1 |

etc.). In the process of such screening, additional manual refinement of the timing of the events was performed with a step of 100 ms to eliminate pauses before and after the event, while not introducing any abrupt jumps at the boundaries.

The mixture generation procedure had the following parameters. For each target class in both training and test sets, there were 500 mixtures. The event presence rate was 0.5 (250 mixtures with target event present and 250 “mixtures” of only background). The event-to-background ratios (EBR) were -6, 0 and 6 dB. The EBR was defined as a ratio of average RMSE values calculated over the duration of the event and the corresponding background segment on which the event will be mixed, respectively. The background instance, the event instance, the event timing in the mixture, its presence flag and the EBR value were all selected randomly and uniformly. The data required to perform the generation of the exact mixtures (the filenames of the background and sound event, if present, the timing and the amplitude scaling factor of the event) was encoded in the so-called recipes. The recipes were generated randomly, but with a fixed seed of the random generator, allowing reproducibility.

The mixtures were generated by summing the backgrounds with the corresponding target event signals according to the recipes, with downsampling to 44100 Hz prior to summation in the case of a higher sampling rate. The resulting signals were scaled with a global empirical factor of 0.2, preserving the dynamics while avoiding clipping. The files were then saved in 24 bit format in order to avoid adding quantization noise.

The dataset is accompanied by a software package, which, given the default parameters, produces exactly the same mixture recipes and audio mixture files as in this dataset. It also allows for tuning the parameters in order to obtain larger and more challenging training datasets: number of mixtures, EBR values and event presence probabilities are adjustable.

The information needed to perform the split into training and test sets in terms of underlying source data was provided. The split of backgrounds was done in terms of recording location ID, according to the first fold of the DCASE 2016 task 1 setup, yielding 844 training and 277 test files. The sound events were split in terms of freesound.org user names. The ratio of target event examples was set to 0.71:0.29, and the split was performed in such a way that the isolated event counts are of a similar ratio. The resulting unique event counts are therefore the following:

- baby cry: 106 training, 42 test;
- glass break: 96 training, 43 test;
- gun shot: 134 training 53 test.

The baseline system follows the common implementation, with the following specifics. For each of the target classes, there is a separate binary classifier with one output neuron with sigmoid activation, indicating the activity of the target class. The performance of the baseline system is evaluated using event-based error rate and

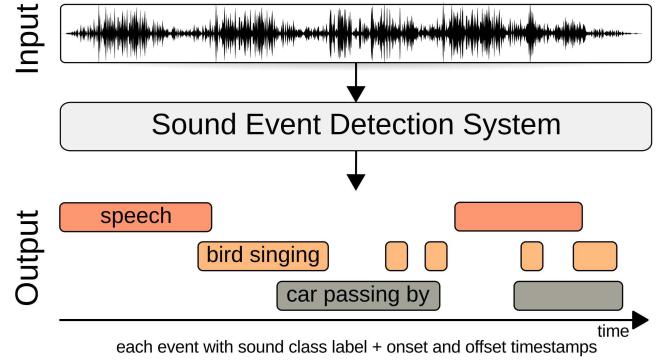


Figure 3: A schematic illustration of sound event detection in real-life audio addressed in Task 3.

event-based F-score as metrics using development dataset mixtures (provided training and test sets). Both metrics are calculated as defined in [27], using a collar of 500 ms and taking into account only the sound event onset. The performance of the baseline system is reflected in Table 3. The primary evaluation score for this task is the event-based error rate, and ranking of the systems submitted for the challenge is done using the average event-based error rate over the three classes.

5. TASK 3: SOUND EVENT DETECTION IN REAL-LIFE AUDIO

Task 3 evaluated the performance of sound event detection systems in multisource conditions similar to our everyday life, where the sound sources are rarely heard in isolation. A number of predefined sound event classes were selected, and systems are meant to detect the presence of these sounds, providing labels and timestamps to segments of the test audio, as illustrated in Fig. 3. In this task, there is no control over the number of overlapping sound events at each time, not in the training, nor in the test audio data.

The dataset used for this task is a subset of TUT Acoustic Scenes 2017, and is referred to as TUT Sound Events 2017. It consists of recordings of street acoustic scenes (city center and residential area) with various levels of traffic and other activity. The length of the audio is 3–5 minutes. The street acoustic scene was selected as representing an environment of interest for detection of sound events related to human activities and hazard situations.

Individual sound events in each recording were annotated by the same person using freely chosen labels for sounds, according to the annotation procedure described in [26]. Nouns were used to characterize the sound source, and verbs to characterize the sound production mechanism, using a noun-verb pair whenever this was possible. The annotator was instructed to annotate all audible sound events, decide the start time and end time of the sounds as he sees fit, and choose event labels freely.

The target sound event classes were selected so as to represent common sounds related to human presence and traffic. The selected sound classes for the task are: brakes squeaking, car, children, large vehicle, people speaking, and people walking. Mapping of the raw labels was performed, merging sounds into classes described by their source, for example “car passing by”, “car engine running”, “car idling”, etc into “car”, sounds produced by buses and trucks into “large vehicle”, “children yelling” and “children talking” into

Table 4: Event instances per class in Task 3

| Event label | Dev. set | Eval. set |
|------------------|----------|-----------|
| brakes squeaking | 52 | 24 |
| car | 304 | 110 |
| children | 44 | 19 |
| large vehicle | 61 | 24 |
| people speaking | 89 | 47 |
| people walking | 109 | 48 |
| total | 659 | 272 |

Table 5: Baseline system results for Task 3, segment-based metrics.

| | Development set | | Evaluation set | |
|-------------------------------|-----------------|-------------|----------------|-------------|
| | ER | F-score (%) | ER | F-score (%) |
| Overall | 0.69 | 56.7 | 0.93 | 42.8 |
| Class-wise performance | | | | |
| brakes squeaking | 0.98 | 4.1 | 0.92 | 16.5 |
| car | 0.57 | 74.1 | 0.76 | 61.5 |
| children | 1.35 | 0.0 | 2.66 | 0.0 |
| large vehicle | 0.90 | 50.8 | 1.44 | 42.7 |
| people speaking | 1.25 | 18.5 | 1.29 | 8.6 |
| people walking | 0.84 | 55.6 | 1.44 | 33.5 |

“children”, etc. Due to the high level of subjectivity inherent to the annotation process, a verification of the reference annotation was done using these mapped classes. Three persons (other than the annotator) listened to each audio segment annotated as belonging to one of these classes, marking agreement about the presence of the indicated sound within the segment. Event instances that were confirmed by at least one person were kept, resulting in elimination of about 10% of the original event instances.

Partitioning of data into development and evaluation datasets was done based on the amount of examples available for each sound event class. Because the event instances belonging to different classes are distributed unevenly within the recordings, the partitioning of individual classes can be controlled only to a certain extent, but so that the majority of events are in the development set. A cross-validation setup provided in order to make results reported with this dataset uniform. The setup consists of four folds containing training and test subsets, and is made so that each recording is used exactly once as test data. While creating the cross-validation folds, the only condition imposed was that the test subset does not contain classes which are unavailable in the training subset. The number of instances for each event class in the development set is presented in Table 4. Evaluation set statistics will be added in the camera ready version.

The baseline system was tailored to a multi-class multi-label classification setup, with the network output layer containing sigmoid units that can be active at the same time. This way, multiple output units can indicate activity of overlapping sound classes. The results are evaluated using segment-based error rate and segment-based F-score as metrics, using a segment length of one second. The four cross-validation folds are treated as a single experiment: the metrics are calculated by accumulating error counts (insertions, deletions, substitutions) over all folds [27], not by averaging the individual folds nor the individual class performance. This method of calculating performance gives equal weight to each individual sound instance in each segment, as opposed to being influenced by class balance and error types [29]. The system trained and tested using the provided cross-validation setup obtained an overall error

rate of 0.69 and an overall F-score of 56.7% on the development set, as shown in Table 5. On the evaluation dataset, the system obtained an error rate of 0.93 and an F-score of 42.8. For completeness, individual class performance is presented along the overall performance. The primary evaluation score for this task is the overall segment-based error rate, and ranking of the systems submitted for the challenge is also done using the same metric, calculated on the evaluation dataset.

6. TASK 4: LARGE-SCALE WEAKLY SUPERVISED SOUND EVENT DETECTION FOR SMART CARS

Task 4 evaluated systems for the large-scale detection of sound events using weakly labeled audio recordings. The audio comes from YouTube video excerpts related to the topic of transportation and warnings. The topic was chosen due to its industry relevance and the under use of audio in this context. The results will help define new grounds for large-scale sound event detection and show the benefit of audio for self-driving cars, smart cities and related areas. The task consisted of detecting sound events within 10-second clips and it was divided into two subtasks:

- Subtask A: Without timestamps (same as audio tagging, Fig 4)
- Subtask B: With timestamps (similar to Task 3, Fig 3)

The task employed a subset of AudioSet [30]. AudioSet consists of an ontology of 632 sound event classes and a collection of 2 million human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. To collect the dataset, Google worked with human annotators who listened, analyzed, and verified the sounds they heard within the YouTube 10-second clips. To facilitate faster accumulation of examples for all classes, Google relied on available YouTube metadata and content-based search to nominate candidate video segments that were likely to contain the target sound. Note that AudioSet does not come with precise time boundaries for each sound class within the 10-second clips and thus annotations are considered weak labels. Also, one clip may correspond to more than one sound event class. Task 4 relied on a subset of 17 sound events divided into two categories: *Warning* and *Vehicle*.

- *Warning sounds*: Train horn, Air horn, Truck horn, Car alarm, Reversing beeps, Ambulance (siren), Police car (siren), Fire engine fire truck (siren), Civil defense siren, Screaming.
- *Vehicle sounds*: Bicycle, Skateboard, Car, Car passing by, Bus, Truck, Motorcycle, Train.

For both subtasks, the data was divided in two main partitions: development and evaluation. The development data was itself divided into training and test. Training had 51,172 clips, which are class-unbalanced and had at least 30 clips per sound event. Test had 488 clips, with at least 30 clips per class. A 10-second clip may have corresponded to more than one sound event class. The evaluation set had 1,103 clips, with at least 60 clips per sound event. The sets had weak labels denoting the presence of a given sound event within the audio, but with no timestamp annotations. For test and evaluation, strong labels (timestamp annotations) were provided for the purpose of evaluating performance on Subtask B.

The task rules did not allow the use of external data, such as other datasets. Similarly, it was not allowed to use other elements

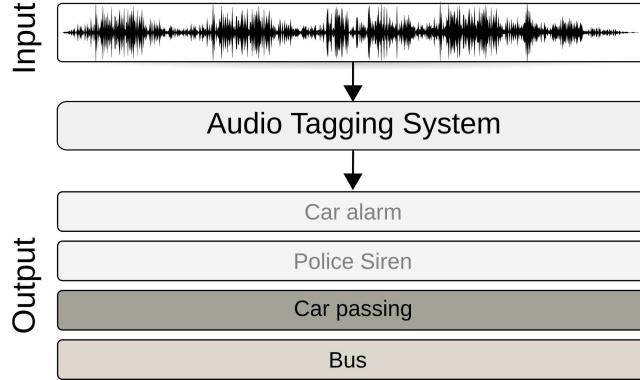


Figure 4: A schematic illustration of audio tagging addressed in Subtask A of Task 4.

of the video from which the 10-sec clip was extracted, such as the rest of the video soundtrack, the video frames and the metadata (e.g. text, views, likes). Moreover, participants were not allowed to use the embeddings provided by AudioSet or other features that used external data indirectly, such as the ones derived from Transfer Learning. Additionally, only weak labels and none of the strong labels (timestamps) could be used for training the submitted system.

The evaluation metric of the two subtasks was different. For Subtask A, sound event detection without timestamps (audio tagging), we used F-score, precision and recall, where ranking of submitted systems was based on F-score. For Subtask B, sound event detection with timestamps, we used segment-based error rate (SBER) [27] and F-score, where ranking of submitted systems was based on SBER for segments of length one-second.

The baseline system shares the code base with other tasks, with detection decision based on the network output layer containing sigmoid units that can be active at the same time. The system also includes evaluation of the overall and class-wise results. The baseline was trained using the training set and tested using the test set and later also tested on the evaluation set. The results for the testing and evaluation sets are shown in Tables 6 for Subtask A and 7 for Subtask B.

7. CONCLUSIONS

The DCASE 2017 Challenge proposed four tasks relevant to current research in environmental sound classification. Compared to previous challenge, the current edition tackled two specific situations, namely detection of sound events that may appear very rarely, and the problem of using weak labels for training sound event detection systems. Of the established tasks, acoustic scene classification and sound event detection in real life audio were seen as important and yet to be solved research problems, worthy of inclusion in the ongoing work.

Through its public datasets and reporting of results, the challenge promotes open research and publications, disseminating the outcome to a large audience. The provided baseline system also offered a starting point for further development, along with a set comparison reference for each task.

8. ACKNOWLEDGMENT

Thanks to Rohan Badlani for his contribution to Task 4.

Table 6: Baseline system results for Task 4 - Subtask A, sound event detection without timestamps (audio tagging) based on micro-averaging.

| | Development set (%) | | | Evaluation set (%) | | |
|---------|---------------------|-------|------|--------------------|-------|-------|
| | F-score | Prec. | Rec. | F-score | Prec. | Rec. |
| Overall | 10.9 | 7.9 | 17.6 | 18.18 | 15.0 | 23.07 |

Class-wise performance

| | | | | | | |
|------------------------------------|------|------|------|------|------|------|
| Train horn | 0.0 | 0.0 | 0.0 | 14.1 | 100 | 7.6 |
| Air horn, truck horn | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Car alarm | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Reversing beeps | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Ambulance (siren) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Police car (siren) | 35.8 | 29.1 | 46.6 | 38.8 | 32.6 | 47.8 |
| Fire engine, fire truck (siren) | 22.7 | 25.0 | 20.8 | 19.3 | 25.7 | 15.5 |
| Civil defense siren | 57.5 | 47.7 | 72.4 | 47.9 | 34.0 | 81.0 |
| Screaming | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Bicycle | 0.0 | 0.0 | 0.0 | 4.2 | 100 | 2.1 |
| Skateboard | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Car | 11.3 | 6.0 | 98.3 | 29.9 | 17.9 | 92.2 |
| Car passing by | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Bus | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Truck | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Motorcycle | 0.0 | 0.0 | 0.0 | 14.2 | 100 | 7.6 |
| Train | 4.5 | 100 | 2.3 | 7.8 | 100 | 4.0 |

Table 7: Baseline system results for Task 4 - Subtask B, sound event detection with timestamps, based on segment-based error rate. The character [-] represents no prediction output by the system.

| | Development set | | Evaluation set | |
|---------|-----------------|-----------|----------------|-----------|
| | ER | F-score % | ER | F-score % |
| Overall | 1.02 | 13.8 | 0.93 | 28.4 |

Class-wise performance

| | | | | |
|------------------------------------|------|------|------|------|
| Train horn | 1.00 | - | 0.98 | 3.9 |
| Air horn, truck horn | 1.00 | - | 1.0 | - |
| Car alarm | 1.00 | - | 1.0 | - |
| Reversing beeps | 1.00 | - | 1.0 | - |
| Ambulance (siren) | 1.00 | - | 1.0 | - |
| Police car (siren) | 1.03 | 28.7 | 1.01 | 34 |
| Fire engine, fire truck (siren) | 1.02 | 8.4 | 0.98 | 16.5 |
| Civil defense siren | 0.69 | 58.2 | 0.64 | 67.4 |
| Screaming | 1.00 | - | 1.0 | - |
| Bicycle | 1.00 | - | 0.99 | 2.5 |
| Skateboard | 1.00 | - | 1.0 | - |
| Car | 5.9 | 21.1 | 1.75 | 46 |
| Car passing by | 1.00 | - | 1.0 | - |
| Bus | 1.00 | - | 1.0 | - |
| Truck | 1.00 | - | 1.0 | - |
| Motorcycle | 1.00 | - | 0.97 | 6.1 |
| Train | 1.00 | 0.5 | 0.99 | 1.8 |

9. REFERENCES

- [1] M. Bugalho, J. Portelo, I. Trancoso, T. Pellegrini, and A. Abad, "Detecting audio events for semantic video search," in *Interspeech*, 2009, pp. 1151–1154.
- [2] 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," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, Jan 2006.
- [3] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustic (WASPAA)*, October 2015.
- [4] S. Goetze, J. Schröder, S. Gerlach, D. Hollosi, J. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, March 2012.
- [5] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. on Multimedia*, vol. 17, no. 10, pp. 1733–1746, October 2015.
- [6] T. Virtanen, A. Mesaros, T. Heittola, M. Plumbley, P. Foster, E. Benetos, and M. Lagrange, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing, 2016.
- [7] D. Barchiesi, D. Giannoulis, D. Stowell, and M. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, May 2015.
- [8] 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," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.
- [9] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *18th European Signal Processing Conference*, Aug 2010, pp. 1272–1276.
- [10] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 1, pp. 142–153, Jan. 2015.
- [11] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, "Experiments on the DCASE challenge 2016: Acoustic scene classification and sound event detection in real life recording," in *DCASE2016 Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [12] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real-life recordings," in *18th European Signal Processing Conference (EUSIPCO 2010)*, 2010, pp. 1267–1271.
- [13] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22st ACM International Conference on Multimedia (ACM-MM'14)*, Nov. 2014.
- [14] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, June 2017.
- [15] J. Gemmeke, L. Vugene, P. Karsmakers, B. Vanrumste, and H. Van hamme, "An exemplar-based NMF approach to audio event detection," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2013, pp. 1–4.
- [16] A. Mesaros, O. Dikmen, T. Heittola, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 151–155.
- [17] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.
- [18] B. Byun, I. Kim, S. M. Siniscalchi, and C.-H. Lee, "Consumer-level multimedia event detection through unsupervised audio signal modeling," in *INTERSPEECH*, 2012, pp. 2081–2084.
- [19] B. Elizalde, G. Friedland, H. Lei, and A. Divakaran, "There is no data like less data: Percepts for video concept detection on consumer-produced media," in *Proceedings of the 2012 ACM international workshop on Audio and multimedia methods for large-scale video analysis*. ACM, 2012, pp. 27–32.
- [20] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. Jackson, and M. D. Plumbley, "Unsupervised feature learning based on deep models for environmental audio tagging," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, 2017.
- [21] W. Han, E. Coutinho, H. Ruan, H. Li, B. Schuller, X. Yu, and X. Zhu, "Semi-supervised active learning for sound classification in hybrid learning environments," *PloS one*, vol. 11, no. 9, p. e0162075, 2016.
- [22] A. Shah, R. Badlani, A. Kumar, B. Elizalde, and B. Raj, "An approach for self-training audio event detectors using web data," *arXiv preprint arXiv:1609.06026*, 2016.
- [23] A. Kumar and B. Raj, "Weakly supervised scalable audio content analysis," in *2016 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2016.
- [24] T. Heittola, A. Diment, and A. Mesaros, "DCASE2017 baseline system," <https://github.com/TUT-ARG/DCASE2017-baseline-system>, accessed June 2017.
- [25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.
- [26] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, 2016.
- [27] ———, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016. [Online]. Available: <http://www.mdpi.com/2076-3417/6/6/162>

- [28] T. Giannakopoulos, “pyaudioanalysis: An open-source python library for audio signal analysis,” *PloS one*, vol. 10, no. 12, 2015.
- [29] G. Forman and M. Scholz, “Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement,” *SIGKDD Explor. Newslett.*, vol. 12, no. 1, pp. 49–57, nov 2010. [Online]. Available: <http://doi.acm.org/10.1145/1882471.1882479>
- [30] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

GENERATIVE ADVERSARIAL NETWORK BASED ACOUSTIC SCENE TRAINING SET AUGMENTATION AND SELECTION USING SVM HYPER-PLANE

Seongkyu Mun

Korea University
Dept. of Visual Information Processing, Seoul,
136-713, South Korea
skmoon@ispl.korea.ac.kr

Sangwook Park

Korea University
School of Electrical Eng.,
Seoul, 136-713, South Korea
swpark@ispl.korea.ac.kr

David K. Han

Office of Naval Research, Arlington VA,
USA
ctmkhan@gmail.com

Hanseok Ko

Korea University
School of Electrical Eng., Seoul,
South Korea
hsko@korea.ac.kr

ABSTRACT

Although it is typically expected that using a large amount of labeled training data would lead to improve performance in deep learning, it is generally difficult to obtain such DataBase (DB). In competitions such as the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge Task 1, participants are constrained to use a relatively small DB as a rule, which is similar to the aforementioned issue. To improve Acoustic Scene Classification (ASC) performance without employing additional DB, this paper proposes to use Generative Adversarial Networks (GAN) based method for generating additional training DB. Since it is not clear whether every sample generated by GAN would have equal impact in classification performance, this paper proposes to use Support Vector Machine (SVM) hyper plane for each class as reference for selecting samples, which have class discriminative information. Based on the cross-validated experiments on development DB, the usage of the generated features could improve ASC performance.

Index Terms— acoustic scene classification, generative adversarial networks, support vector machine, data augmentation, decision hyper-plane

1. INTRODUCTION

One of the fundamental issues in deep learning is availability of large labeled data set. It has been consistently shown over the last decade that larger labeled data set with deeper network layers can lead to improved results. However, it is not easy to collect large amounts of labeled data, so it is necessary to extract the maximum performance with a small amount of data depending on the application. An example of such constraint is the case of the IEEE DCASE challenge Task 1 for ASC [1-3]. Although it has been well known that the given ASC DB of the competition is insufficient for high classifier performance, there has not been much attempt on augmenting the insufficient amount of data among the participants by using methods such as semi-supervised learning (or pre-training) employing additional databases [4-7]. This is because one of the rules in DCASE challenge prohibits the use of external DBs other than the DCASE Task 1 development set. Obviously, pre-training network using additional DB larger than the development set could improve ASC performance, as shown in our previous research [8]. However, this is not allowed in the DCASE challenge.

Therefore, to improve ASC performance without employing additional DB, our DCASE 2017 work focuses on DB generation. To generate new samples using only the development DB, we propose to use GAN models. The GAN learns two sub-networks: a generator and a discriminator. The discriminator reveals whether a sample is generated or real, while the generator produces samples to pass through the discriminator as real data. The GANs are first proposed by Goodfellow et al. [7] to generate images and gain insights into neural networks. Then, Deep Convolutional GANs (DCGANs) [9] addressed the issue of instability inherent in training GAN. The discriminator of DCGAN can serve as a robust feature extractor. On the other hand, GANs also demonstrate potential in generating images for specific applications. Pathak et al. [10] proposed an encoder-decoder method for image inpainting, where GANs are used as the image generator. Several researches have attempted to use the GAN generated samples as training samples. For labeling the generated samples, the generated samples were all taken as one class in the discriminator in [4-5]. Zheng [6] adopted a novel regularization approach by assigning a uniform label distribution to the generated samples. Although additional data generated by GAN may lead to improved classifier training, it is not clear whether every data point generated by GAN would have equal impact in classifier performance. As it has been shown by SVM, those support vectors that reside near decision boundary are generally crucial in providing key information in classification [16]. We believe that performance could be improved by selecting the generated data by measuring decision value (distance) from decision hyper-plane of SVM for each class.

Recently, GAN has been applied to several acoustic applications, such as voice conversion, speech synthesis and speech enhancement [11-13]. These applications have reference signals for training, such as the same contents of speech set that multiple speakers uttered [11], reference speech already generated by conventional synthesis methods [12], or noisy/clean speech sample pairs [13]. In the case of classifications, typically there is no reference signal which the generator can be built up from. Therefore, instead of training the GAN based speech sample (raw waveform) generator, we propose to use the GAN as an ASC feature generator.

For ASC feature extraction, we used a combined structure of LSTM and CNN with inputs, such as spectrogram and log Mel-Filter Bank (MFB) energy. Using the extracted ASC features, a SVM hyper-plane and a GAN generator for each class were trained. Using the GAN generator, sample pool for each class were generated. Afterwards, based on the criterion of SVM deci-

sion value and the classification rate on the seen/unseen validation DB, feature sampling and new SVM training are conducted iteratively. We used the feature set configuration, which shows the highest performance on seen and unseen data, as the final training DB. More details will be covered in Section 2.2.

2. PROPOSED FRAMEWORK

The process of the proposed GAN based framework is depicted in Figure 1. Following the development DB setup of the baseline system [14], we divided the development DB at 3:1 ratio for training and validation. For validating the GAN generated samples, we divided the training part in half, Tr-A and Tr-B in Figure 1. The GAN based feature generation and selection were done individually for each class. Therefore, a total of 15 GANs were trained. After the feature samples were generated and selected by GAN and SVM, the augmented feature sets were used for training and validation with Fully Connected Neural Network (FCNN) and SVM for final classification. For improving performance, we conducted late fusion on SVM and FCNN results.

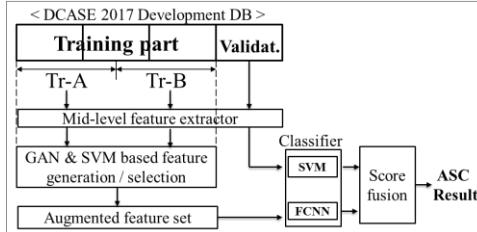


Figure 1: Block-diagram of the proposed framework

2.1. Mid-level ASC feature extraction

For spectrogram or Mel filter-based feature input, various types of networks have been studied in DCASE 2016 [3]. Among the various approaches, we chose a structure of parallel combination of LSTM and CNN [1] to extract both sequential and local time-frequency associated information. Using the network in Figure 2, we could get the classification results directly for the development DB, but we extracted the mid-layer values of network as ASC feature for further processing. For more information on mid-level feature extraction, see [15] or [8] for the visual object classification or ASC. As mentioned in the introduction, due to difficulty of generating raw waveforms using GANs, we used the mid-layer values as ASC features to train GAN and generate ‘fake’ data.

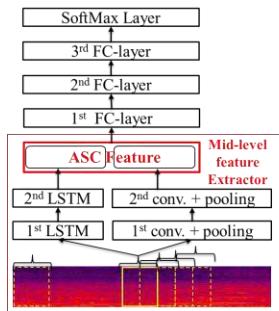
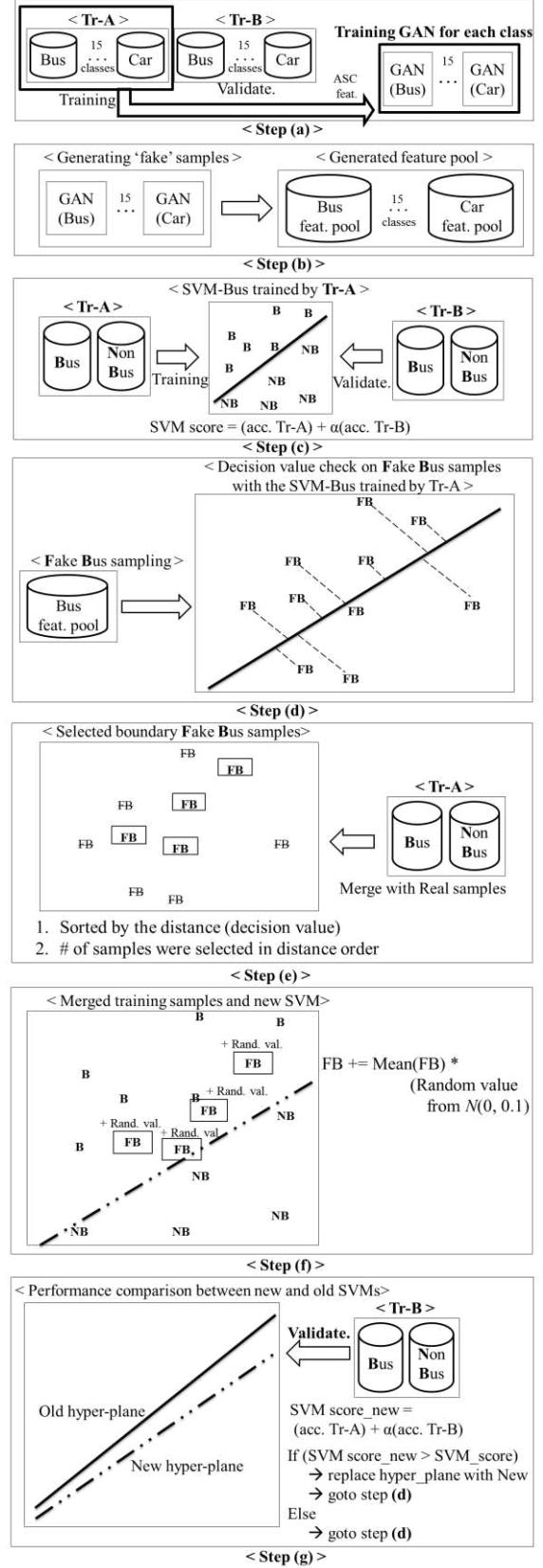


Figure 2: The neural net structure for the feature extractor



2.2. Generative adversarial net based training feature set augmentation

The process of the proposed GAN based feature generation and selection are depicted in Figure 3. As shown in step (a), a GAN for each class was trained using the part of the development set, which excludes the validation part for following steps. More details of GAN configuration will be covered in Section 3.2.

Using the trained GANs, we generated ‘fake’ samples and organized the sample feature pools for each class as shown in step (b). Before using the generated samples, an SVM hyper-plane for each class (target class vs. the others) was first determined from the real data set to establish a baseline performance. We chose the bus class as an example. Note that half of the training set was used for training and the other half was used for validating SVM performance. As shown in step (c), we checked classification performance of SVM with the sum of the training and validation set accuracy. Considering the SVM update in the next step, we added a weight (α , which is bigger than 1) to the unseen data, i.e. validation accuracy.

In step (d), we subsampled ‘fake bus’ features from the generated bus feature pool and checked decision values on the SVM hyper-plane trained from Tr-A set. As shown in step (e), we sorted the fake samples by the distance order, and chose a preset number of the nearest samples. Additionally, we also included small number of samples near the hyper-plane that were classified as non-bus by handicapping their decision value. We then merged the near boundary fake samples with the real samples of Tr-A set. Step (f) shows the new SVM hyper-plane trained by the merged set. Before training the new SVM, we added random vectors, which are scaled to the magnitude of the samples, to reduce the sample bias of the generation using GAN. As was done in step (c), the classification performance of new SVM was checked with the sum of the training (Tr-A) and validation set (Tr-B) accuracy. If the accuracy score of the new SVM outperforms the previous SVM score, the reference SVM hyper-plane was replaced with the new one and the iteration continues again with the fake sample subsampling in the step (d). If not, the iteration proceeds to the step (d) without replacing the reference hyper-plane. All steps of subsampling, sorting, selecting, merging and performance checking are repeated until the iterative process reaches a preset number of rounds or the performance converged. Once the SVM performance is optimized, the associated support vectors of fake bus features were used for the augmented training set. The entire process is repeated with the Tr-B as the training set for GAN and SVM, and Tr-A as the validation set. As shown in Figure 4, the whole processes are repeated for each acoustic scene class. The amount of feature pool was approximately 50 times that of the training DB (Tr-A or Tr-B), and amount of selected features was a similar to the training DB. In other words, the amount of the final augmented DB was about twice that of the original DB.

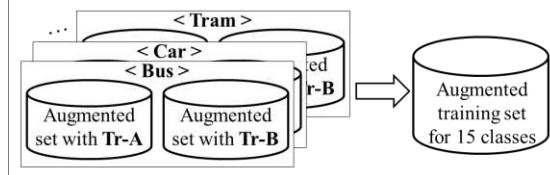


Figure 4: Final augmented training set for 15 classes

3. EXPERIMENTAL SETTINGS AND RESULTS

3.1. Input features and mid-level feature extractor configuration

For generating input features, audio signals sampled at 22.05 kHz sampling frequency were divided into 23.2ms frames with 512-size Discrete Fourier Transform (DFT). MFB and DFT spectrogram features were used as the input of the ASC feature extractor individually. Following [2], we used left, right, average and difference of both channel audio inputs. Total 4-types of sources were grouped into one DB set. We followed most of the details in neural network of [1]. The specific network architecture of the mid-level feature extractor is depicted in Table 1.

Table 1: The model specifications. (Batch size : 200 samples)

| | MFB case | DFT case |
|-------------------------|--|---|
| Input | Input feature length : 1 [sec] / overlapping : 0.5 [sec] [40 (feat.) x 42 (time-frame)] | [256 (feat.) x 42(time-frame)] |
| LSTM #1 & 2 | Hidden unit (300) / ReLU / Dropout (0.2) | Hidden unit (400) / ReLU / Dropout (0.2) |
| Conv. #1 | 4 x 4 (stride 1) / 10 filters / ReLU / Dropout (0.2) 2 x 2 max-pooling | 16 x 8 (stride 1) / 10 filters /ReLU / Dropout (0.2) 2 x 2 max-pooling |
| Conv. #2 | 4 x 4 (stride 2) / 4 filters / ReLU / Dropout (0.2) 2 x 2 max-pooling | 8 x 4 (stride 2) / 4 filters /ReLU / Dropout (0.2) 2 x 2 max-pooling |
| Mid-layer for ASC feat. | Hidden unit (800) / FCNN layer consists of two hidden layers with 400 ReLU units for each | |
| FCNN #1-3 | Hidden unit (300) / Final Soft-Max layer (15) / Dropout (0.2) | |

3.2. Generative adversarial network configuration

As mentioned in the introduction, various types of GANs have been widely researched. In this work, we do not focus on investigating more sophisticated sample generation methods. Instead, we use a basic GAN model [7, 9] to generate samples from the training data and show that these samples help to improve discriminative learning for the unseen validation data. In the GAN for 2-D images, the convolutional layer of DCGAN is generally used [4-6, 9] for 2-D matrix, but in this work, we used FCNN to process the feature vector (800×1 [dim.]) for simplicity. In order to help convergence of the discriminator, we added the normalized mean feature vector of the each class along with the random value as a GAN input. The specific network architecture of the GAN is depicted in Figure 5.

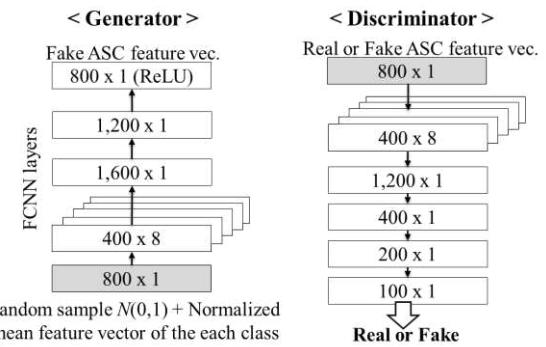


Figure 5: The neural network structure of GAN for each class

3.3. Classifier and late fusion

To assess the performance of the proposed DB augmentation, we conducted a number of experiments on the original DCASE 2017 development set and the augmented DB set, which consists of original samples and selected fake samples. As shown in the lower part of the Figure 1, SVM and FCNN were used as classifiers for the ASC feature inputs. The FCNN consists of 3-hidden layers with 300 hidden nodes and the SVM with a radial basis function kernel were used. For late fusion on the multiple classifier, we used linear logistic regression [2,17] on classification scores of DFT features with SVM and FCNN, and MFB features with SVM and FCNN in both cases of original and augmented DB. ($8 = 2 \times 2 \times 2 = [\text{feat. types}] \times [\text{classifier types}] \times [\text{DB types}]$)

3.4. Results

We compared the average ASC accuracies over all scenes for the SVM and FCNN classifiers trained by the original DB and the augmented DB expended by the proposed framework. The file-based (10 [sec]) classification results are given in Table 2. For evaluation, the average waveforms on stereo audio input were used. Table 3 shows the class-wise classification accuracy on the fusion cases. As shown in Table 2, the proposed framework with the augmented DB set achieved higher accuracy than other cases. This can be interpreted that the proposed augmented DB set could infer properties of unseen DB and usage of the generated features could generalize or improve ASC performance. As given in Table 3, although the performances of the all classes were not improved by the proposed method, but the overall average accuracy outperformed the conventional approaches.

Table 2: Comparing the performance of the conventional and the proposed method (average accuracy on 4-fold validation)

| Avg. acc. [%] | with original development set | | | | with augmented set | | | |
|---------------------|----------------------------------|--------------|-------------|-------------|--------------------|--------------|-------------|-------------|
| | DFT- FCNN | MFB- FCNN | DFT- SVM | MFB- SVM | DFT- FCNN | MFB- FCNN | DFT- SVM | MFB- SVM |
| 75.4 | 75.1 | 78.2 | 79.3 | 83.2 | 83.7 | 81.6 | 81.6 | 85.6 |

Table 3: The class-wise accuracy comparison on the dev. set

| Acc. [%] | Baseline [14] | Fusion w/o augmented DB case | Fusion on all cases |
|----------|------------------|---------------------------------|------------------------|
| Beach | 75.3 | 70.9 | 71.8 |
| Bus | 71.8 | 82.1 | 87.2 |
| Café | 57.7 | 71.8 | 87.2 |
| Car | 97.1 | 89.0 | 88.5 |
| City | 90.7 | 85.6 | 98.7 |
| Forest | 79.5 | 97.3 | 94.9 |
| Groce. | 58.7 | 83.3 | 79.5 |
| Home | 68.6 | 76.0 | 89.7 |
| Lib. | 57.1 | 82.0 | 96.2 |
| Metro | 91.7 | 90.7 | 84.6 |
| Office | 99.7 | 95.1 | 96.2 |
| Park | 70.2 | 69.9 | 71.8 |
| Resid. | 64.1 | 71.8 | 87.2 |
| Train | 58.0 | 71.8 | 82.1 |
| Tram | 81.7 | 84.6 | 91.0 |
| Avg. | 74.8 | 81.5 | 87.1 |

3.5. Submissions

The experiments shown in the Table 2-3 were conducted with the default setting of the DCASE 2017 development (4-fold cross validation). However, in order to reflect more information of the development set for the challenge submission, we conducted the additional ASC feature generation based on the various DB configurations, such as 2-fold, 3-fold and 8-fold frameworks. In particular, additional DB augmentation processing was conducted on similar class pairs, such as train/tram, home/library and park/residential area. We will analyze a quantitative relationship between DB configuration and performance for the future research, after ground truth of the evaluation DB is published.

4. CONCLUSION AND FUTUREWORK

In order to improve ASC performance, this paper proposed a framework to generate feature samples using GANs. The novel method of using SVM hyper-plane to select features for performance improvement was proposed. Based on the experimental result of DCASE 2017 development set, we confirmed that the usage of the generated features could improve ASC performance. GAN and Variational Auto Encoders (VAEs) have shown impressive performance improvements in some studies, but it is still difficult to generate suitable training samples without bias. In order to alleviate the issue, we used an iterative method with added random values in the generated samples to mitigate the issue of sample bias and over fitting. Nevertheless, further statistical considerations and additional quantitative experiments are needed for generalization of training from GAN generated samples.

5. ACKNOWLEDGMENT

This subject is supported by Korea Ministry of Environment (MOE) as "Public Technology Program based on Environmental Policy".

6. REFERENCES

- [1] S. Bae, I. Choi and N. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN", Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016), 2016.
- [2] H. E. Zadeh, et. al., "CP-JKU Submissions for DCASE-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks", IEEE AASP Challenge on DCASE 2016 technical reports, 2016.
- [3] <http://www.cs.tut.fi/sgn/arg/dcase2016/task-results-acoustic-scene-classification>
- [4] A. Odena, "Semi-supervised learning with generative adversarial networks", arXiv:1606.01583, 2016.
- [5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans", In proc. NIPS 2016, pp. 2234-2242, 2016.

- [6] Z. Zheng, L. Zheng, and Y. Yang, “Unlabeled samples generated by gan improve the person re-identification baseline in vitro”, arXiv preprint arXiv:1701.07717, 2017.
- [7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets., In proc. NIPS 2014, pp. 2672-2680, 2014.
- [8] S. Mun, S. Shon, W. Kim, D.K. Han and H. Ko, “Deep Neural Network based learning and transferring mid-level audio features for acoustic scene classification”, 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.796-800, 2017.
- [9] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks”, arXiv preprint arXiv:1511.06434, 2015.
- [10] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536-2544, 2016.
- [11] C. C. Hsu, et. al., “Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks”, arXiv preprint arXiv:1704.00849, 2017.
- [12] T. Kaneko, et. al., “Generative adversarial network-based postfilter for statistical parametric speech synthesis”, In Proc. ICASSP 2017, pp. 4910-4914, 2017.
- [13] S. Pascual, S. A. Bonafonte and J. Serrà, “SEGAN: Speech Enhancement Generative Adversarial Network”, arXiv preprint arXiv:1703.09452., 2017.
- [14] <http://www.cs.tut.fi/sgn/arg/dcase2017/>.
- [15] M. Oquab et. al., “Learning and transferring mid-level image representations using convolutional neural networks”, In Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR), pp. 1717-1724, 2014.
- [16] C. Cortes and V. Vapnik, “Support-vector networks”, *Machine learning*, vol. 20, no.3, pp. 273-297, 1995.
- [17] N. Brummer, “Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores tutorial and user manual”, Software available at <http://sites.google.com/site/nikobrummer/focalmulticlass>, 2007

ACOUSTIC SCENE CLASSIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORK USING DOUBLE IMAGE FEATURES

Sangwook Park

Korea University
School of Electrical Eng.,
Seoul, 136-713,
South Korea
swpark@ispl.korea.ac.kr

Seongkyu Mun

Korea University
Dept. of Visual Infor-
mation Processing, Seoul,
136-713, South Korea
skmoon@ispl.korea.ac.kr

Younglo Lee

Korea University
School of Electrical
Eng., Seoul, 136-713,
South Korea
yllee@ispl.korea.ac.kr

Hanseok Ko

Korea University
School of Electrical
Eng., Seoul,
South Korea
hsko@korea.ac.kr

ABSTRACT

This paper proposes new image features for the acoustic scene classification task of the IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events. In classification of acoustic scenes, identical sounds being observed in different places may affect performance. To resolve this issue, a covariance matrix, which represents energy density for each subband, and a double Fourier transform image, which represents energy variation for each subband, were defined as features. To classify the acoustic scenes with these features, Convolutional Neural Network has been applied with several techniques to reduce training time and to resolve initialization and local optimum problems. According to the experiments which were performed with the DCASE2017 challenge development dataset it is claimed that the proposed method outperformed several baseline methods. Specifically, the class average accuracy is shown as 83.6%, which is an improvement of 8.8%, 9.5%, 8.2% compared to MFCC-MLP, MFCC-GMM, and CepsCom-GMM, respectively.

Index Terms— Acoustic scene classification, covariance learning, double FFT, convolutional neural network

1. INTRODUCTION

Audio signals ranging from speech and general sounds (non-linguistic sound) to background sounds may be quite informative in characterizing context such as presence of humans, objects, their activities, or the environment. Among these contexts, location information is not only vital in multimedia analysis but also widely applicable to many tasks in scene understanding [1-2]. Location information is also useful as prior information for enhancing performance of speech/acoustic event recognition [3-4]. Thus, Acoustic Scene Classification (ASC) which focuses on identifying where the audio signal has been obtained has drawn considerable attention. IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events (DCASE) 2017 also included this task as the ASC challenge.

Typically, an ASC system consists of feature extraction and classification. In feature extraction, Mel Frequency Cepstral Coefficients (MFCCs) and Perceptual Linear Prediction (PLP) have been applied as the early stage of the proposed ASC system. Low-level spectral features such as zero-crossing rate,

spectral statistics, and timbre were employed for ASC by combining them with MFCCs [5]. A Bag-Of-Frames (BOF) method, which considers an acoustic scene as a set of bags of various sounds, was applied to ASC [6-8]. The BOF approach has used statistical distribution (e.g. histogram) as features, which represents the occurrence count of cepstral features, quantized by a codebook like dictionary. However, the BOF approach is too sensitive to training data due to the requirement of training phases in both feature extraction and classification.

Recently, approaches based on i-vector which is widely being used for speaker recognition has also been applied for ASC [9]. Since i-vector is extracted from hyper-dimensional vector space by applying factor analysis, potential discriminable characters can be revealed with the feature. In the last challenge, numerous approaches based on deep learning were introduced. Mun et al. proposed a classification framework based on bottleneck feature extraction with Deep Neural Networks (DNN) [10]. Takahashi et al. investigated DNN-Gaussian Mixture Model (GMM) framework for classifying MFCCs [11]. Similarly, Convolutional Neural Network (CNN) was applied for classifying log-mel spectrograms [12]. In [13], an ensemble method which is composed of hundreds of CNNs was proposed for stochastic feature extraction. Recurrent Neural Network (RNN) based approaches were also proposed [14-15]. In [16], combined CNN and Long Short-Term Memory (LSTM) model has been proposed for ASC. For applying deep learning methods, sufficient training data is required to avoid local optimum problems. Thus, manipulation of development datasets is also considered as one of the major issues for training DNNs.

Although many approaches have attempted ASC applications, they still suffer from realistic environment problems. Even in the same environment, audio signals may vary depending on presence of people, objects, and their behaviors. For example, in a café, a microphone may collect differing occurrences of sounds such as cleaning, coffee grinding, or people talking. Therefore, the feature vectors obtained in cafés will likely be widely scattered in a feature space, although these vectors have originated from the same place. Meanwhile, features representing conversations will always be observed in all environments where there are people. As illustrated by this example, performing location classification for ASC is a challenging task in realistic environments.

This paper describes an ASC method which is applied for the DCASE 2017 challenge under this practical issue. According

to a hypothesis that sound frequency over the time may differ according to places, two types of image features, covariance matrix and double Fast Fourier Transform (FFT) image, are proposed. These features represent temporal energy density and energy variations for each frequency, respectively. Also, they are insensitive to training data, because they can be obtained without any trained data. To perform scene classification with these features, Convolutional Neural Network (CNN) is considered. Additionally, the appropriate CNN structure is also investigated to improve performance. In experiment, the proposed method is demonstrated by using the DCASE 2017 challenge database [17].

The remainder of this paper is organized as follows. Section 2 explains proposed image features with its motivation. Section 3 introduces CNN approaches for classifying the proposed features. After a discussion on the experimental results, conclusions are drawn in the final section.

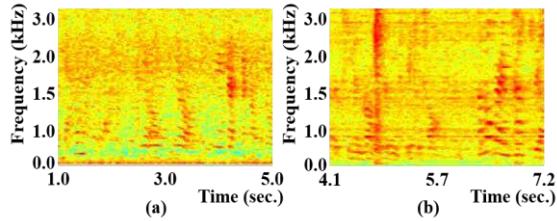


Figure 1: Human voice spectrograms included in development dataset for DCASE2017 challenge; (a) in library, a part of “b027_190_200.wav” (b) in home, a part of “a031_100_110.wav”

2. PROPOSED IMAGE FEATURE

2.1. Motivation

Although the spectrograms are obtained in distinct places, human voices are heard in both places, *library* and *home* as shown in Figure 1. In this case, many conventional features extracted from spectrum may encounter confusion due to not only human voice but also other sounds heard anywhere, because their spectrums look very similar to each other. To overcome this problem, it is necessary to search for difference by the combination of spectrums during a finite interval. One method to observe this combination of these spectral features is via histogram with BOF. However, BOF is too sensitive to training data, because it requires training phases for feature extraction as well as classification.

In this paper, to representing combination of spectrums during a finite interval, covariance matrix and frequency analysis of frequency bins are considered. Figure 2 shows covariance matrices and images obtained by performing FFT on spectrogram in each frequency bin. As shown in Figure 2, *library* and *home* can be distinguished by using these images. Based on this fact, two image features, covariance matrix of spectrums (COV) and Double FFT Image (DFI), are proposed.

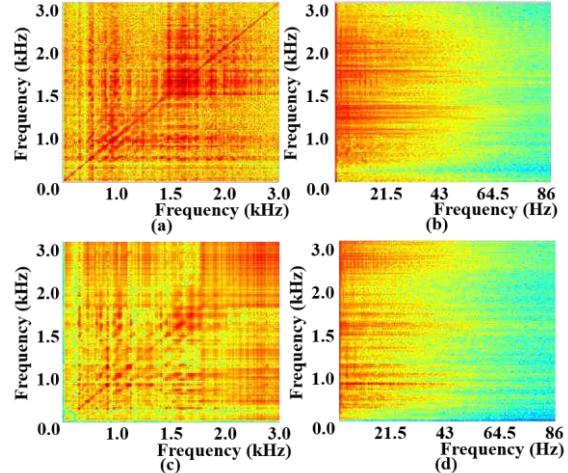


Figure 2: Two image features corresponding to Figure 1; (a) covariance matrix in library (b) frequency analysis in library (c) covariance matrix in home (d) frequency analysis in home

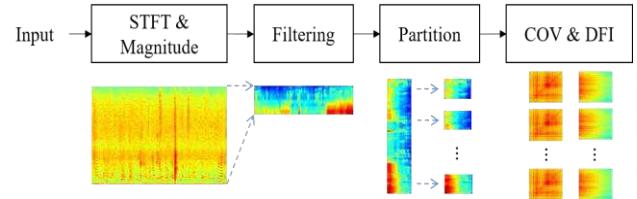


Figure 3: Procedure for obtaining proposed image feature

2.2. Image Feature Extraction

Figure 3 shows the procedure of obtaining the two image features. A 1-dimensional wave is transformed to spectrogram after pre-emphasis. In *Filtering*, a compressive Gammachirp filterbank is applied to all spectrums for dimension reduction [18]. The result of *Filtering* is partitioned into several blocks that are composed of consecutive filter responses. For each block, COV is calculated by performing expectation as

$$\mathbf{C}^i = E \left[\left(X[:,m] - \bar{X}^i \right) \left(X[:,m] - \bar{X}^i \right)^T \right], \quad X[:,m] \in B^i \quad (1)$$

where B^i is a set whose elements are filter responses included in the i^{th} block. \mathbf{C}^i is a covariance matrix of the i^{th} block. X and \bar{X}^i are filter responses in each frame and frame average, respectively, and m is frame index.

To obtain DFI, FFT is performed on each subband of filter responses as

$$\mathbf{D}^i = F_{k=1:K} \{ X[k,:] \}, \quad X[k,:] \in B^i \quad (2)$$

where \mathbf{D}^i is a DFI of the i^{th} block, F is a function for FFT. K and k is the number of frequency bin and frequency index, respectively. Finally, min-max normalization is performed on both \mathbf{C}^i and \mathbf{D}^i for representing gray-scale image.

3. CONVOLUTIONAL NEURAL NETWORK

Many deep learning methods suffer from several problems such as over-fitting, local optimum, and training time. In CNN, the number of parameters is reduced by sharing weights in convolutions to avoid over-fitting problems. Also, CNN is well known to be appropriate for classifying image. From these reasons, CNN is applied for classification of proposed image feature. As shown in Figure 4, the CNN structure consists of three parts; input, convolution, and fully connected.

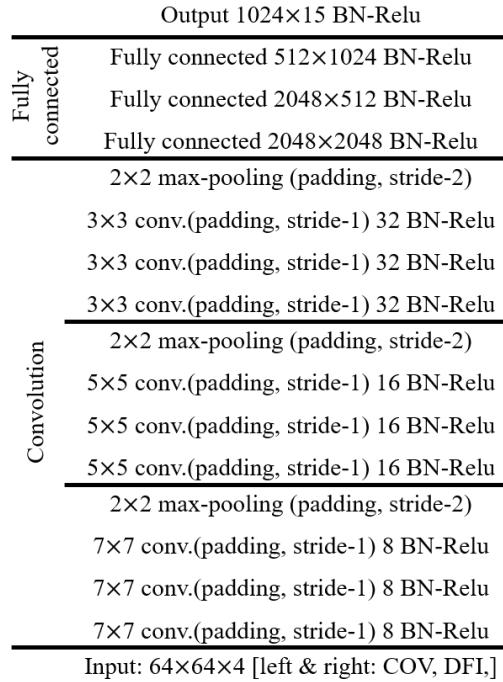


Figure 4: A CNN structure applied for classification of proposed image features

3.1. CNN Input

Similar to human perception, stereo sound contains additional information such as direction of sound and spatial characters. Since these are also helpful for location recognition, CNN input is composed of four images which are COVs and DFIs obtained from each channel. These inputs are converted to gray scale image whose pixel value is integer within 0 to 255. Thus, all pixels in the proposed feature are normalized with zero-mean and unit-variance.

In feature extraction, the number of filters included in the considered filterbank is set to 64, and the number of double FFT points is set to 128.

3.2. Convolution

In training of the deep network, obtaining initialization parameters is very important to avoid local optimum problem. Since CNN structures are empirically determined, training time is also another issue. To alleviate these problems, batch normali-

zation has been applied in every layer [19]. Thus, three sub-steps, convolution, batch normalization, and pooling, are conducted in this step.

In convolution, if a large filter is used, microscopic features can be obtained, but the number of training parameters is also increased. To avoid this constraint, convolutions are iteratively performed with a small filter. Based on this concept, filter sizes applied to final structure are depicted in Figure 4. After convolution, Batch Normalization (BN) is also performed.

In pooling, the image size is diminished in half by applying max pooling. Note that the number of tensors is increased after pooling. This is also iteratively performed after every convolution iteration.

3.3. Fully connected

After all iterations, the result of *Convolution* is reshaped to vector (8x8x32 dimension) for application to the fully connected network. In this time, Rectified Linear Unit (Relu) function is used for activation function. The number of nodes in each layer is depicted in Figure 4.

4. EXPERIMENT

4.1. Experimental Setting

For performance assessment, DCASE2017 development dataset that consists of 15 scenes, *bus*, *café/restaurant*, *car*, *city center*, *forest path*, *grocery store*, *home*, *lakeside beach*, *library*, *metro station*, *office*, *residential area*, *train*, *tram*, and *urban park*, was used. By using four fold lists provided by DCASE2017 committee, cross-validation tests were conducted.

For performance comparison, several baselines were considered. Firstly, the results of MFCC-MLP and MFCC-GMM were provided by DCASE2017 challenge committee [20]. Secondly, CepsCom that is a 240-dimensional vector composed by concatenating four cepstral features was evaluated by using 128 mixture GMM [21]. Finally, CepsCom based i-vector framework was considered [22]. Based on 128 mixture GMM, a 400 dimensional i-vector was extracted in this experiment. After applying multi-class Linear Discriminative Analysis (LDA) to 400 dimensional i-vector, classification was performed by using a minimum Cosine Distance Score (CDS).

In proposed method, the length of block was empirically set to 1 second. Thus, CNN was trained with approximately 33,000 inputs in each fold test. (Note that about 45,000 inputs were used for evaluation)

4.2. Experiment Results

The accuracies according to classes are summarized in Table 1. In baseline systems except i-vector-CDS, the performance is shown to be about 75%. Although logMel-MLP and MFCC-GMM shows similar performance (i.e. class averaging accuracy), CepsCom-GMM shows the best averaging accuracy, which is an improvement of 0.6%. In logMel-MLP and MFCC-GMM, accuracies above 90% can be obtained in *car*, *city center* and *office*, and accuracy below 60% is shown in *train*. On the other

Table 1. Experiment results for four baseline systems and proposed system

| [%] | Avg. | beach | bus | cafe | car | city | forest | groc. | home | lib. | metro | office | park | resid. | train | tram |
|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| logMel-MLP | 74.8 | 75.3 | 71.8 | 57.7 | 97.1 | 90.7 | 79.5 | 58.7 | 68.6 | 57.1 | 91.7 | 99.7 | 70.2 | 64.1 | 58.0 | 81.7 |
| MFCC-GMM | 74.1 | 75.0 | 84.3 | 81.7 | 91.0 | 91.0 | 73.4 | 67.9 | 71.4 | 63.5 | 81.4 | 97.1 | 39.1 | 74.7 | 41.0 | 79.2 |
| CepsCom-GMM | 75.4 | 78.2 | 82.9 | 75.0 | 91.0 | 91.7 | 61.1 | 81.4 | 70.8 | 58.0 | 78.1 | 96.8 | 53.9 | 74.7 | 54.6 | 83.5 |
| i-vector-CDS | 61.9 | 79.5 | 63.5 | 56.7 | 32.1 | 86.9 | 68.4 | 77.9 | 54.8 | 56.7 | 69.4 | 66.0 | 55.1 | 48.7 | 51.5 | 64.0 |
| Proposed | 83.6 | 88.5 | 93.8 | 73.1 | 93.2 | 86.3 | 97.1 | 84.6 | 82.5 | 77.8 | 89.2 | 91.0 | 73.1 | 70.5 | 70.1 | 82.7 |

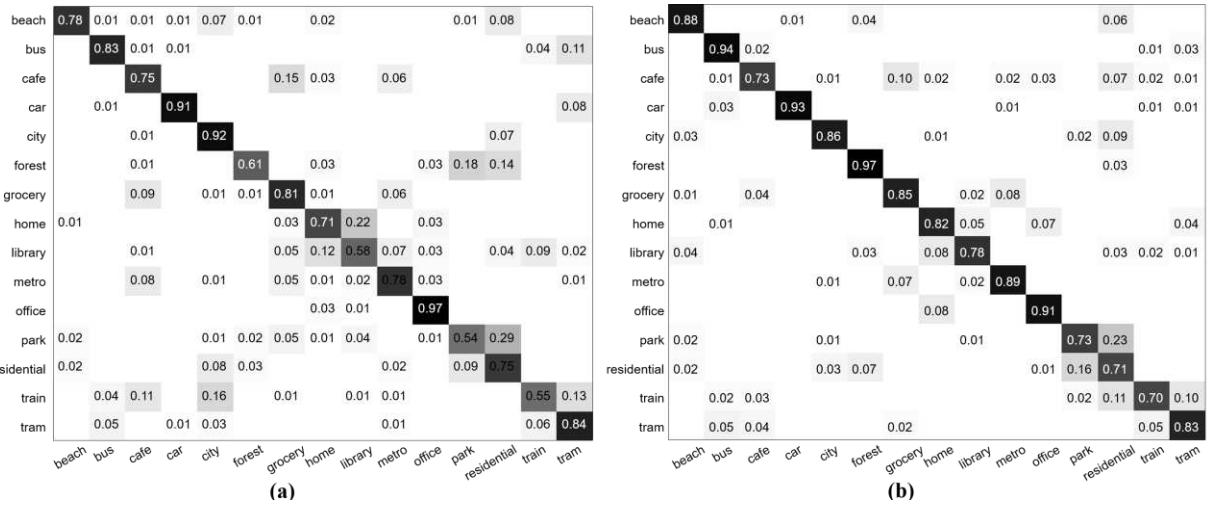


Figure 5: Confusion matrices for CepsCom-GMM and the proposed method (a) CepsCom-GMM (b) Proposed

hand, MFCC-GMM outperforms other methods in *café* and *residential area* while logMel-MLP outperforms others in *metro-station* and *office*. This difference may come from that different classifier, MLP or GMM, which has been applied to each method. In addition, the features is also different. Since 40 filters for Mel filterbank are used for feature extraction, logMel is a 40-dimensional vector while MFCC is a 12-dimentional vector by conducting Discrete Cosine Transform (DCT). In case of CepsCom-GMM, this method outperforms others in *city center* and *tram*.

In i-vector-CDS, the performance is about 62%. An UBM is very important for extracting i-vector, and a huge volume of database including a lot of scenes is required for training UBM. Despite this fact, development dataset consisted of 15 scenes is only used for training UBM in this experiment. To obtain reliable results using i-vector, a larger database is required to successfully training UBM.

According to the results, the proposed method outperforms other methods. The class average accuracy was observed as 83.6%, which is an improvement of 8.8%, 9.5%, 8.2% compared to MFCC-MLP, MFCC-GMM, and CepsCom-GMM, respectively, which implies that the best class accuracies were obtained for most classes. Additionally, confusion matrices for CespCom-GMM and the proposed method are shown in Figure 5. As mentioned previously, spectrum based features such as MFCC and CepsCom confuse scenes where common sound may be heard. Although class accuracies have been lower than base-

line in *café* and *office*, the proposed method resolves this problem as shown in *bus*, *library*, *home* and *train*. To additionally improve performance of the proposed method, the confusion between *park* and *residential area* has to be resolved.

5. CONCLUSIONS

This paper proposed new image features, COV and DFI, for resolving an issue that common sounds can be heard anywhere. The COV is a covariance matrix of spectrums which represents energy densities for each frequency subband. The other feature, DFI, represents variation of energy in each subband, which can be obtained by performing FFT. These features can be easily obtained without training data. To perform classifying using these features, CNN is applied with several techniques for reducing training time and resolving problems about initialization and local optimization. Efficiency of proposed method is demonstrated in experiment with development dataset provided for DCASE2017 challenge. From the results, proposed method outperforms other methods by means of class average accuracy with 83.6%, which is an improvement of 8.8%, 9.5%, 8.2% compared to MFCC-MLP, MFCC-GMM, and CepsCom-GMM, respectively.

6. ACKNOWLEDGMENT

This subject is supported by Korea Ministry of Environment (MOE) as "Advanced Technology Program for Environmental Industry".

7. REFERENCES

- [1] W. Choi, S. Kim, M. Keum, D. K. Han, and H. Ko, "Acoustic and visual signal based context awareness system for mobile application," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 738-746, 2011.
- [2] K. Yamano and K. Itou, "Browsing audio life-log data using acoustic and location information," in *IEEE Int. Conf. Mobile Ubiquitous Computing, Systems, Services and Technologies*, pp. 96-101, 2009.
- [3] T. Nishiura, S. Nakamura, K. Miki, and K. Shikano, "Environmental sound source identification based on hidden markov model for robust speech recognition," in *EUROSPEECH 2003*, pp. 2157-2160, 2003.
- [4] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal Audio, Speech, and Music Processing*, pp. 1-13, 2013.
- [5] 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," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321-329, 2006.
- [6] V. Carletti, P. Foggia, G. Percannella, A. Saggese, N. Strisciuglio, and M. Vento, "Audio surveillance using a bag of aural words classifier," in *IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, pp. 81-86, 2013.
- [7] 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," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881-891, 2007.
- [8] S. Pancoast and M. Akbacak, "Bag-of-Audio-Words approach for multimedia event classification," in *INTERSPEECH 2012*, pp. 2105-2108, 2012.
- [9] H. E. Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "A hybrid approach with multi-channel i-vectors and convolutional neural networks for acoustic scene classification," arXiv preprint arXiv:1706.06525, 2017.
- [10] S. Mun, S. Park, Y. Lee, and H. Ko, "Deep neural network bottleneck feature for acoustic scene classification," *DCASE2016 challenge technical report*, 2016
- [11] G. Takahashi, T. Yamada, S. Makino, and N. Ono, "Acoustic scene classification using deep neural network and frame-concatenated acoustic feature," *DCASE2016 challenge technical report*, 2016.
- [12] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE2016 acoustic scene classification using convolutional neural networks," *DCASE2016 challenge technical report*, 2016.
- [13] J. Kim and K. Lee, "Empirical study on ensemble method of deep neural networks for acoustic scene classification," *DCASE2016 challenge technical report*, 2016
- [14] T. H. Vu and J. C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *DCASE2016 challenge technical report*, 2016.
- [15] M. Zohrer and F. Pernkopf, "Gated recurrent networks applied to acoustic scene classification and acoustic event detection," *DCASE2016 challenge technical report*, 2016.
- [16] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," *DCASE2016 challenge technical report*, 2016.
- [17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: tasks, datasets and baseline system," in *Proc. Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*. Nov. 2017. submitted
- [18] T. Irino, "A compressive gammachirp auditory filter for both physiological and psychophysical data," *J. Acoust. Soc. Am.*, vol. 109, no. 5, pp. 2008-2022, May 2001.
- [19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. conf. Machine Learning*, vol. 37, pp. 448-456, 2015.
- [20] <http://www.cs.tut.fi/sgn/arg/dcase2017/>.
- [21] S. Park, S. Mun, Y. Lee, and H. Ko, "Score fusion of classification systems for acoustic scene classification," *DCASE2016 challenge technical report*, 2016.
- [22] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Aud. Spee. Lang. Proc.*, vol 19, no. 4, pp. 788-798, May 2011.

THE DETAILS THAT MATTER: FREQUENCY RESOLUTION OF SPECTROGRAMS IN ACOUSTIC SCENE CLASSIFICATION

Karol J. Piczak

Institute of Computer Science
Warsaw University of Technology

ABSTRACT

This study describes a convolutional neural network model submitted to the acoustic scene classification task of the DCASE 2017 challenge. The performance of this model is evaluated with different frequency resolutions of the input spectrogram showing that a higher number of mel bands improves accuracy with negligible impact on the learning time. Additionally, apart from the convolutional model focusing solely on the ambient characteristics of the audio scene, a proposed extension with pretrained event detectors shows potential for further exploration.

Index Terms— acoustic scene classification, spectrogram, frequency resolution, convolutional neural network, DCASE 2017

1. INTRODUCTION

The area of environmental sound classification has recently experienced a significant increase in the quantity of performed studies. One of the main driving factors in 2016 was the organization of the first DCASE workshop [1], complemented by an open challenge focusing on the detection and classification of acoustic scenes and events. This unique opportunity enabled researchers to exchange ideas and evaluate various approaches on a common set of tasks and datasets, a valuable initiative which continues in 2017 with a second installment of the workshop [2].

Looking at previous submissions to this challenge, a clear picture emerges on how diverse the methods employed to tackle these tasks can be. In 2013, when the very first DCASE challenge [3] was organized, although most approaches used a support vector machine (*SVM*) classifier, the input frames spanned a vast range of features:

- mel-frequency cepstral coefficients (*MFCC*) [4, 5, 6],
- mel spectrograms processed through a sparse RBM extractor [7],
- statistics from a cochleogram based on a tone-fit algorithm [8],
- responses of modulation tuned filters (2D Gabors) [9],
- visual features (*HOG*) computed on a constant-Q transform [10].

At the same time, other teams evaluated the usefulness of hidden Markov models (*HMM*) [11], an i-vector approach combined with MFCCs [12], bagging of decision trees with MFCCs and wavelets [13] and a random forest classifier working on an embedding through dissimilarity representation [14].

In contrast, the DCASE 2016 challenge saw an emergence of deep learning techniques with numerous systems shifting to deep neural networks (*DNN*) [15, 16, 17, 18], convolutional neural networks [19, 20, 21, 22, 23, 24], recurrent models [25, 26, 27, 28, 29] and their fusions with other approaches like the i-vector [30]. Although MFCCs were still widely encountered as input features in

The source code for this study can be found at:
<https://github.com/karoldvl/paper-2017-DCASE>

sound event detection, more low-level representations such as mel band energy and various forms of spectrograms were much more common in the acoustic scene classification task.

When developing models relying on spectrograms as their input, one of the decisions that has to be made is the resolution of the data generated in the preprocessing step. What should it be? One obvious response is: “the higher, the better”. As visualized by Figure 1, choosing a more fine-grained representation means less information is being lost at the very beginning and this should hopefully allow for more nuanced differentiation between similar training examples in later stages. However, there are three countervailing issues that we have to take into consideration here.

First of all, although increasing the time and frequency resolution of the employed representation may be desirable, the uncertainty principle imposes a theoretical limit on how these two can be combined. It is always a trade-off. Wide windows give good frequency resolution, but their temporal resolution is affected for the worse. Narrow windows behave in the opposite way.

One can counter this claim by stating that, theoretical limits notwithstanding, in most cases it is still possible to maintain a temporal resolution sufficient for an audio classification task while using wider windows. Even then, however, a practical aspect of resource constraints remains. Will the impact on memory and storage requirements introduced by a higher resolution be acceptable in a given application? Is a longer computation time, both in the preprocessing and learning phase, really worth it? Especially in scenarios combining real-time processing with deployment on low-power devices these issues can become crucial.

Finally, dimensionality reduction of the input data is a proven way to facilitate learning. Looking from this perspective, a single audio frame of 10 milliseconds, sampled at 44.1 kHz, contains 441 datapoints in its raw form. On the contrary, MFCCs can succinctly describe it with only a dozen of coefficients. With longer frames the discrepancy will be even more pronounced. Therefore, a valid concern arises whether a high-resolution spectrogram with hundreds of frequency bands will not become an overkill that effectively impedes efficient learning.

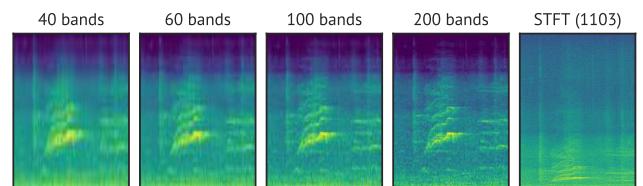


Figure 1: A visual comparison of 3-second-long fragments of spectrograms with different frequency resolutions (first four use a mel scale, the last one is a plain STFT).

Evaluating related works in this area, it seems indeed that the prevailing tendency is to limit the number of computed frequency bands to less than 100. Although greater values can be occasionally encountered (100 in [28], 128 in [31], 150 in [32] and even 1025 in [27]), 60 bands [20, 23] and 40 bands [17, 26, 29, 33] are the dominant option. This would imply either that the gains potentially achievable from a higher resolution are counterbalanced by other negative factors, or that the issue is deemed, so far at least, only tangential to the actual problem of model construction and has not received much attention of itself.

This specific research question is the main motivation behind this study. A thorough analysis of all the issues voiced in the introduction is not possible in a scope of a short paper, so it will be limited to an evaluation of a single submission to the acoustic scene classification task of the DCASE 2017 challenge [2]. Nevertheless, it will hopefully signal whether this problem could be worth investigating further in a more generalized manner.

2. EXPERIMENT SETUP

2.1. Task and dataset

The goal of the acoustic scene classification task proposed in the DCASE 2017 challenge is to determine the context of a given recording by choosing one appropriate label from a set of 15 predetermined acoustic scenes. For each scene, there are 312 audio segments in the development dataset with each segment having a length of 10 seconds and a sampling rate of 44.1 kHz. The challenge organizers prearranged the development dataset into 4 folds for comparable cross-validation in such a manner that segments originating from one physical location are contained in the same fold. The final scoring of submitted systems is based on the fraction of correctly classified segments from the evaluation dataset. Further information about the recording and annotation procedure can be found in the paper describing the dataset [34].

2.2. Data preprocessing

The first step of the proposed solution consists in converting all the provided recordings into spectrograms with *librosa v0.5.1* [35]. Mel spectrograms are created with an FFT window length of 50 ms (2205 samples), hop length of 20 ms (882 samples), and a number of bands that is either 40, 60, 100 or 200, in all cases covering a frequency range of up to 22050 Hz. Additionally, a plain STFT spectrogram (1103 bands) is created with the same window and hop length for comparison. Finally, the spectrograms are converted to a decibel scale and standardized by subtracting the mean and dividing by the standard deviation computed on a random batch of 1000 examples. In this manner, the resulting dimension of a 10-second-long segment representation is b rows and 500 columns, where b is the number of generated frequency bands.

During training, slight data augmentation is introduced by a uniformly distributed offset of the start time of up to 1 second. Moreover, in each case a randomly sized tail of the generated example is replaced with a different segment belonging to the same class, creating some additional variety in the training batches.

2.3. Model architecture

Most acoustic scenes can be conceptually described as an ensemble of two distinct elements. The ambience layer consists of a nondescript theme recurring in the background with little to no change (e.g. sound of a noisy street). Every now and then a more specific

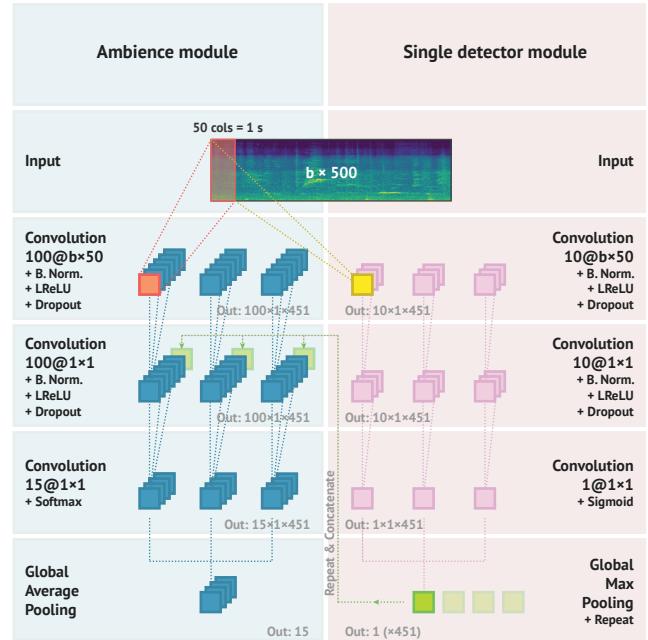


Figure 2: A schematic of the model with its ambience part and a possible extension with a detector module.

event of a short-lived nature occurs (e.g. a book page being flipped in a library). In many situations the background information alone is quite sufficient for establishing an actual context with little ambiguity. However, browsing through the provided dataset and trying to deduce how human perception copes with such a task, it seems that in some cases very subtle clues (as the aforementioned page flipping) are the key elements that drastically shift the expectations between similar contexts (e.g. *home* and *library*).

Based on this observation, a natural question is whether a machine learning model incorporating such an assumption would be advantageous. On the other hand, taking into consideration the high accuracy of the baseline solution and the results of Mafra et al. [36] where the authors indicate that good results can be obtained by representing each recording with only a single averaged frame, it is thus very likely that a good architecture should not be overly complicated in this case.

Therefore, the system described in this work has a very simple design, coming in two flavors depicted in Figure 2. The first variant is a three block convolutional network focusing on processing the ambience content. Its first layer takes the whole input spectrogram ($b \times 500$) and applies a convolution with a stride of 1, filter size of $b \times 50$ (i.e. over fragments of 1 second) and the number of filters set at 100. The response is batch normalized and processed through a LeakyReLU activation ($\alpha = 0.3$) combined with dropout ($p_{drop} = 0.25$). The second processing block is identical, except for the filter size which in this case is reduced to 1×1 , meaning that there is no spatial convolution but only aggregation across feature maps. The final layer consists of 15 convolutional filters of 1×1 and a softmax activation that is computed separately for each step. For training purposes, output probabilities are averaged with a global pooling layer. However, during the prediction step no pooling is performed, but instead these values are binarized with a threshold of 0.5 and only then averaged over the whole time span, which is equivalent to a majority vote.

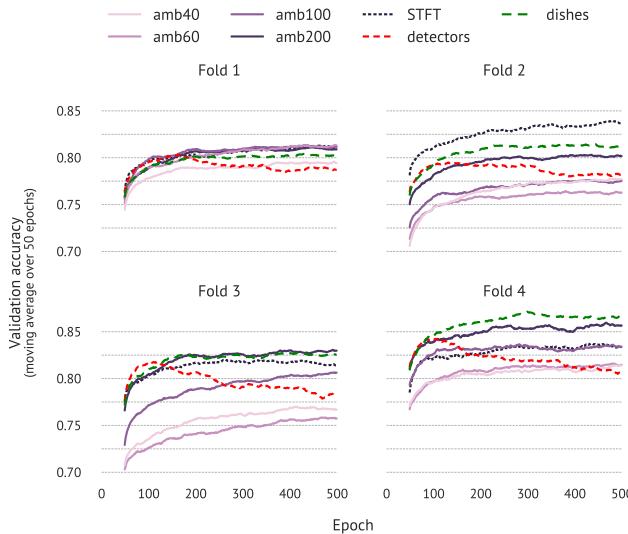


Figure 3: Comparison of validation accuracy for the evaluated systems achieved on the development dataset. Results are presented as a moving average over 50 epochs for better clarity.

The second variant extends this model with a module that we will further call a *detector*. An architecture of a detector is exactly the same as the already described ambient part with the difference that convolutional blocks use only 10 filters and the last layer consists of a single convolutional unit with sigmoid activation that is max-pooled over the whole time span. The rationale behind this is that the output of such a network should signal whether a given event (template match) has occurred anywhere in the whole recording. The whole variant then combines the ambient module with a predefined number of detectors by concatenating their output to the input of the last convolutional layer (same global event detection value is repeated for each step of the ambient model).

Two remarks about the implications of such an architecture. First of all, while we are using a “convolutional” designation for this model, were it not for some subtle differences coming from the use of normalization layers and joint training, it could be validly understood as a simple multi-layer perceptron that is being applied to consecutive frames of the input, an approach very similar to the baseline implementation.

Moreover, by filling the first layer with filters of a very large size, spanning the whole frequency range, we can limit the impact of higher resolutions to this layer only. This means that the increase in computation time is not that severe. The prospects here would be much worse with networks stacking multiple layers of small-sized filters, where such changes propagate in the output dimensions of deeper layers, a drawback which should not be overlooked.

2.4. Training procedure

Before training, all model weights are initialized with a *He uniform* [37] procedure. Training is performed for 500 epochs with an *Adam* optimizer (learning rate of 0.001, batch size of 32) and a categorical cross-entropy loss function. 400 segments are carved out from the training fold as an additional holdout batch. The best performer on the holdout batch is retained as the final model, whereas validation results are calculated on a completely separate fold as provided by the organizers. Separate models are trained for each

| Overall accuracy: 82.33% | | | | | | | | | | | | | | | | |
|--------------------------|------------------|-----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| True label | beach | 250 | 1 | 0 | 6 | 4 | 7 | 0 | 2 | 3 | 0 | 1 | 24 | 13 | 0 | 1 |
| | bus | 0 | 279 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 9 | 15 |
| | cafe/restaurant | 0 | 0 | 250 | 0 | 1 | 0 | 33 | 18 | 4 | 10 | 9 | 1 | 0 | 1 | 5 |
| | car | 0 | 3 | 0 | 301 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 5 | 96.5% |
| | city_center | 0 | 0 | 2 | 0 | 293 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 14 | 0 | 0 |
| | forest_path | 2 | 0 | 1 | 0 | 0 | 290 | 0 | 5 | 0 | 0 | 0 | 0 | 14 | 0 | 0 |
| | grocery_store | 1 | 0 | 22 | 0 | 0 | 0 | 269 | 2 | 2 | 14 | 0 | 0 | 2 | 0 | 0 |
| | home | 8 | 1 | 2 | 1 | 0 | 3 | 0 | 251 | 31 | 0 | 20 | 0 | 1 | 0 | 0 |
| | library | 13 | 15 | 5 | 0 | 2 | 5 | 5 | 29 | 207 | 11 | 12 | 0 | 2 | 4 | 2 |
| | metro_station | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 2 | 305 | 0 | 0 | 0 | 0 | 1 |
| | office | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 5 | 0 | 289 | 0 | 0 | 0 | 0 |
| | park | 27 | 1 | 4 | 0 | 5 | 5 | 0 | 3 | 4 | 0 | 0 | 198 | 61 | 0 | 4 |
| | residential_area | 10 | 0 | 1 | 0 | 15 | 27 | 0 | 0 | 4 | 0 | 0 | 49 | 205 | 0 | 1 |
| | train | 1 | 13 | 11 | 4 | 19 | 0 | 0 | 11 | 1 | 0 | 0 | 1 | 225 | 25 | 72.1% |
| | tram | 2 | 8 | 1 | 11 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 2 | 0 | 15 | 266 |
| | | bea | bus | caf | car | cit | for | gro | hom | lib | met | off | par | res | tra | |
| | | Predicted label | | | | | | | | | | | | | | |

Figure 4: Confusion matrix of the submitted *amb200* model (ambience only, 200 mel bands) combined over all folds of the development set. The rightmost column presents class-wise accuracies.

combination of training and validation folds. A hierarchical learning method similar to the one reported in [17] was tentatively evaluated, however the difference achieved with the employed architecture was not noticeable enough to warrant further investigation.

3. RESULTS

The main system presented in this work, codenamed *amb*, consisted solely of the ambience processing module (left part of Figure 2). Five variants of this model were evaluated, four using mel spectrograms with 40, 60, 100 and 200 frequency bands respectively and one working on STFT spectrograms with 1103 bands (denoted as *STFT* later on). Additionally, a model combining the *amb200* variant with 15 independent detector modules was created (*detectors*). The results of these models are depicted in Figure 3 and presented in a numerical way in Table 1, while Figure 4 more specifically details class-wise performance of the *amb200* model.

The analysis of these results indicates that a higher number of mel frequency bands quite uniformly improves the achieved validation accuracy. There is almost a 4 percentage point difference between *amb40* and *amb200* variants, showing that, in this setup at least, higher resolution models have a greater predictive capacity. The *STFT* variant is on average comparable to *amb200*, it is however underperforming in fold 4 and strongly outperforming in fold 2. Taking into consideration the processing overhead (approximate epoch processing time on a GTX 980 Ti card was 23 s for *amb40* up to 25 s for *amb200* and 52 s for *STFT*) the *amb200* variant is a clear winner.

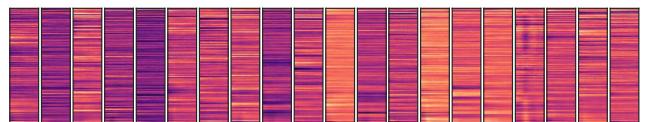


Figure 5: First 20 filters learned by the initial convolutional layer of the *amb200* model (ambience only, 200 mel bands).

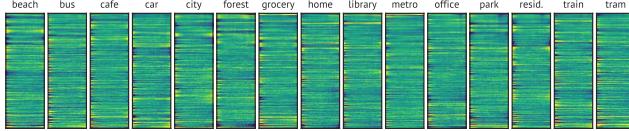


Figure 6: Synthetic examples of input patterns resulting in maximum output activation for a given class ($p_{\text{class}}(\mathbf{X}) = 1.0$). Slight contrast squashing was applied for presentation purposes.

On the other hand, a disappointing behavior of the *detectors* model combines poor validation accuracy with very high training time. It is quite evident that for this particular dataset the capacity of such an architecture is too high and after 100 epochs of training significant overfitting occurs. Therefore, an additional model *dishes* is proposed. What is peculiar about it, it extends the *amb200* model with only a single detector. Moreover, this detector module is separately pretrained on additional hand-annotations specifically created for this purpose indicating the occurrence of specific events in the *cafe/restaurant* scene that could be described as sounds involving cups, plates, kitchenware etc. Unfortunately, due to time constraints and the effort involved in creating a more complete annotation of the dataset, it was not possible to evaluate a model with a broader range of pretrained detectors. However, taking into consideration that initial results reported here hint at a possible improvement, it could be an interesting further avenue for research.

Finally, trying to understand a bit better on what is going on behind the scenes, we can see that the ambience model learns to be a strong frequency discriminator as seen both by the convolutional filters visualized in Figure 5 and examples of patterns that induce the strongest activation for a specific class (Figure 6). This would explain why a high frequency resolution of the input data might be so important for this task. However, especially looking at Figure 6, the perceptual differences are minuscule apart from some intricate patterns of narrow frequency bands, so a real question is on what is being learned. Is it actually the semantic differentiator between different types of scenes? In some cases, like interiors of vehicles, most probably yes. However, for classes such as *home* or *library* it is quite possible that these specific frequency patterns concentrate on what would be perceived by a human listener as recording artifacts. Further study would be required, but there is a high risk that the resulting model would be prone to adversarial attacks.

Table 1: Results of the proposed systems.

| System | Development | | | | Final | |
|-----------|-------------|------------|------------|------------|-------|------|
| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | | |
| amb40 | 79.4 (0.5) | 77.7 (0.8) | 76.7 (1.0) | 81.4 (1.0) | 78.8 | — |
| amb60 | 81.3 (0.6) | 76.3 (1.0) | 75.8 (0.9) | 81.5 (1.0) | 78.7 | 62.0 |
| amb100 | 81.1 (0.6) | 77.5 (0.9) | 80.6 (0.7) | 83.4 (1.3) | 80.7 | 67.7 |
| amb200 | 80.9 (0.8) | 80.2 (0.8) | 83.0 (0.9) | 85.6 (1.3) | 82.4 | 70.6 |
| STFT | 81.1 (0.9) | 83.6 (0.8) | 81.4 (0.9) | 83.4 (1.3) | 82.4 | — |
| detectors | 78.7 (0.9) | 78.1 (1.1) | 78.6 (1.3) | 80.8 (1.4) | 79.1 | — |
| dishes | 80.3 (0.9) | 81.4 (0.7) | 82.6 (0.6) | 86.6 (1.0) | 82.7 | 69.6 |

Mean (standard deviation) of validation accuracies across 50 final epochs of training on the development set and official evaluation results for submitted models. Values in percentages.

4. CONCLUSION

This paper described a submission to the acoustic scene classification task of the DCASE 2017 challenge based on a convolutional neural network model specifically limited to focusing on the ambient characteristic of auditory scenes by average pooling responses for consecutive fragments of the recording. Experiments completed in this study showed that a very important determinant of the final performance in this task is the frequency resolution of the input representation being used, most probably due to the fact that the network is learning a form of a frequency discriminating function. Therefore, increasing the number of mel bands up to 200, well above what is most commonly encountered in related works, proved to be most effective. At the same time, using plain STFT spectrograms with even higher number of bands did not provide additional gains, while considerably increasing the computation time.

It is hard to tell in the scope of this work whether these results could be generalized for other contexts (e.g. event detection), where apart from the frequency content, changes in time also play a crucial role. Concurrently, while the exact scope of the increase in the processing time when employing different types of models is not clear, a valid concern is whether the gains achieved will compensate for the longer training time, especially when using deep convolutional architectures with very small filters. This would have to be evaluated on a case-by-case basis. Nevertheless, the aim of this study is to underline that this hyperparameter should also be taken into consideration, even if only to squeeze some additional performance out of the very final model.

5. REFERENCES

- [1] T. Virtanen, *et al.*, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing, 2016.
- [2] A. Mesaros, *et al.*, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, Munich, 2017.
- [3] D. Stowell, *et al.*, “Detection and classification of acoustic scenes and events,” *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.
- [4] J. T. Geiger, B. Schuller, and G. Rigoll, “Recognising acoustic scenes with large-scale audio feature extraction and SVM,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [5] W. Nogueira, G. Roma, and P. Herrera, “Sound scene identification based on MFCC, binaural features and a support vector machine classifier,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [6] G. Roma, *et al.*, “Recurrence quantification analysis features for auditory scene classification,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [7] J. Nam, Z. Hyung, and K. Lee, “Acoustic scene classification using sparse feature learning and selective max-pooling by event detection,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [8] J. D. Krijnders and G. ten Holt, “A tone-fit feature representation for scene classification,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

- [9] K. Patil and M. Elhilali, "Multiresolution auditory representations for scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [10] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 1, pp. 142–153, 2015.
- [11] M. Chum, *et al.*, "IEEE AASP scene classification challenge using hidden Markov models and frame based classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [12] B. Elizalde, *et al.*, "An i-vector based approach for audio scene detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [13] D. Li, J. Tam, and D. Toub, "Auditory scene classification using machine learning techniques," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [14] E. Olivetti, "The wonders of the normalized compression dissimilarity representation," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.
- [15] S. Mun, *et al.*, "Deep neural network bottleneck feature for acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [16] G. Takahashi, *et al.*, "Acoustic scene classification using deep neural network and frame-concatenated acoustic feature," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [17] Y. Xu, *et al.*, "Hierarchical learning for DNN-based acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [18] I. Choi, *et al.*, "DNN-based sound event detection with exemplar-based approach for noise reduction," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [19] Y. Han and K. Lee, "Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [20] M. Valenti, *et al.*, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [21] T. Lidy and A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [22] H. Phan, *et al.*, "CNN-LTE: a class of 1-X pooling convolutional neural networks on label tree embeddings for audio scene recognition," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [23] D. Battaglino, *et al.*, "Acoustic scene classification using convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [24] E. Çakir, T. Heittola, and T. Virtanen, "Domestic audio tagging with convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [25] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [26] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [27] M. Zöhrer and F. Pernkopf, "Gated recurrent networks applied to acoustic scene classification and acoustic event detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [28] T. Hayashi, *et al.*, "Bidirectional LSTM-HMM hybrid system for polyphonic sound event detection," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [29] S. Adavanne, *et al.*, "Sound event detection in multichannel audio using spatial and harmonic features," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [30] H. Eghbal-Zadeh, *et al.*, "CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [31] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.
- [32] P. Giannoulis, *et al.*, "Improved dictionary selection and detection schemes in sparse-CNMF-based overlapping acoustic event detection," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [33] I. Sobieraj and M. Plumley, "Coupled sparse NMF vs. random forest classification for real life acoustic event detection," in *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Budapest, 2016.
- [34] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, 2016.
- [35] B. McFee *et al.*, "librosa 0.5.0," Feb. 2017. [Online]. Available: <https://doi.org/10.5281/zenodo.293021>
- [36] G. Mafra, *et al.*, "Acoustic scene classification: An evaluation of an extremely compact feature representation," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2016.
- [37] K. He, *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *IEEE International Conference on Computer Vision (ICCV 2015)*, Washington, DC, 2015.

WAVELETS REVISITED FOR THE CLASSIFICATION OF ACOUSTIC SCENES

Kun Qian^{1,2,3}, Zhao Ren^{2,3}, Vedhas Pandit^{2,3}, Zijiang Yang^{2,3}, Zixing Zhang³, Björn Schuller^{2,3,4}

¹ MISP Group, MMK, Technische Universität München, Germany

² Chair of Embedded Intelligence for Health Care & Wellbeing, Universität Augsburg, Germany

³ Chair of Complex & Intelligent Systems, Universität Passau, Germany

⁴ GLAM – Group on Language, Audio & Music, Imperial College London, UK

andykun.qian@tum.de, schuller@ieee.org

ABSTRACT

We investigate the effectiveness of wavelet features for acoustic scene classification as contribution to the subtask of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE2017). On the back-end side, gated recurrent neural networks (GRNNs) are compared against traditional support vector machines (SVMs). We observe that, the proposed wavelet features behave comparable to the typically-used temporal and spectral features in the classification of acoustic scenes. Further, a late fusion of trained models with wavelets and typical acoustic features reach the best averaged 4-fold cross validation accuracy of 83.2 %, and 82.6 % by SVMs, and GRNNs, respectively; both significantly outperform the baseline (74.8 %) of the official development set ($p < 0.001$, one-tailed z-test).

Index Terms— Acoustic Scene Classification, Wavelets, Support Vector Machines, Sequence Modelling, Gated Recurrent Neural Networks

1. INTRODUCTION

Acoustic scene classification (ASC) is defined as classification of the environment in which a recording has been made [1], which is a subfield of Computational Auditory Scene Analysis [2]. It is based on the assumption that, various acoustic scenes can be distinguished from one another by their general acoustic properties due to general characterisations of a location or situation [1]. In practice, ASC is a challenging task since a certain scene is usually similar to others, and shares commonalities of sound sources all across [3]. In recent years, there is an increasing interest in finding more robust and efficient ASC methods to be applied into multimedia searching [4], smart mobile devices [5], and intelligent monitoring systems [6, 7]. Previous DCASE Challenges in 2013 [8], and 2016 [9] attracted numerous teams from across the world working on this uprising topic. A recent overview on the ASC literature is found in [10]. The acoustic features used for ASC include mel-frequency cepstral coefficients [5, 11], histograms of sounds [12], and histogram of gradients learnt from time-frequency representations [13]. In terms of the classifiers, hidden Markov models (HMMs) [12], Gaussian mixture models (GMMs) [11], and support vector machines (SVMs) [13, 14] have been popular. More recently, a series of methods using deep learning are applied to ASC tasks [3, 15–18].

In this contribution, we investigate the effectiveness of wavelet features for the ASC task, which had been proven to be successful in our previous work in snore sound classification [19–21].

A large scale typical acoustic feature vector extracted by openSMILE [22] is compared with, and combined in a late fusion process with the proposed wavelet features. As to machine learning models, SVMs [23], and gated recurrent neural networks (GRNNs) [24] are implemented and compared. To train both models, we first extract *low level descriptors* (LLDs) on the frame level; then, we apply different functionals over *clips*. It is worth noting that the *clip* for the SVM model refers to a long segment, whereas for the GRNNs, it denotes a short episode which is *sequentially* segmented from a long segment in a fixed duration length. Both for the SVMs and the GRNNs, the models trained independently with wavelets and typical acoustic features, are fused later to make the final decision by a *margin sampling* strategy [25].

In comparison to the enormous focus on cepstral and other spectral features that do not optimise the Heisenberg-alike time-frequency trade-off, there is little attention on the effectiveness of wavelet features for the ASC task. This work thus explores avenues towards accordingly optimised novel features which are efficient in classification of acoustic scenes, and to investigate their performances by the popular classifiers such as SVMs, and state-of-the-art machine learning techniques like GRNNs.

This paper is organised as follows: Section 2 will give a description of the database and the methodology we used. The experimental results will be shown in Section 3 before a conclusion is made in Section 4.

2. METHODOLOGY

2.1. Database

To evaluate the proposed systems, we use the official dataset of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE2017) [1]. This dataset is accessible through the challenge website ¹. The development set contains 312 segments of 10 seconds in each of the 15 classes. The total duration of the development set is 13 hours. The fifteen acoustic scene classes needed to be recognised in this task are: *beach, bus, cafe/restaurant, car, city centre, forest path, grocery store, home, library, metro station, office, park, residential area, train, and tram*. The organisers split the data such that all segments from the same recording are put in either the train or the test partition, for all of the four folds they generated for cross-validation. This is done to evaluate robustness of the proposed systems. The correct recognition accuracy is used as the evaluation metric.

¹<http://www.cs.tut.fi/sgn/arg/dcase2017/>

Table 1: Parametres for wavelet features.

| Wavelet Function | J_{\max} | # of LLDs |
|------------------|------------|-----------|
| WPTE | 'rbio3.3' | 7 |
| WEF | 'db7' | 7 |

2.2. Wavelet Features

We earlier introduced wavelet features into the area of snore sound classification in [19]. Wavelets were found to be effective in localising different snore sounds, and performed better than the other widely-used spectral feature-types such as *mel-frequency cepstral coefficients*, *formants*, *fundamental frequency*, etc. One most recent work on deep wavelet features for ASC was presented in [26]. Firstly, we use the wavelet packet transform energy (WPTE) feature extracted by wavelet packet transformation (WPT) [27]. In contrast to the discrete wavelet transformation (DWT) [28], WPT further decomposes ‘detail’ components to obtain their own ‘approximation’. We use the normalised bank filter energy in [29] as our WPTE LLDs, which is defined as:

$$\mathbf{E}_{\Omega_{j,k}} = \log \sqrt{\frac{\sum_{n=1}^{N_{j,k}} (\mathbf{w}_{j,k,n})^2}{N_{j,k}}}, \quad (1)$$

where $\mathbf{w}_{j,k,n}$ are the coefficients calculated by WPT from the analysed signal at the subspace $\Omega_{j,k}$. $N_{j,k}$ is the total number of wavelet coefficients in the k -th subband at the j -th decomposition level. The scale of k is $0, 1, 2, \dots, 2^j - 1$. Totally, $2^{J_{\max}+1} - 1$ WPTE based LLDs are generated. The J_{\max} is the maximum level for wavelet decomposition by a certain *wavelet function*.

In addition, another wavelet feature set based on DWT is defined as:

$$\tilde{\mathbf{E}}_{\Omega_j} = \frac{(\mathbf{w}_j)^2}{\sum_{j=1}^{J_{\max}} (\mathbf{w}_j)^2} \times 100, \quad (2)$$

where \mathbf{w}_j are the coefficients generated by DWT at the j -th decomposition level. Furthermore, the *mean*, *variance*, *waveform length* (the sum of the absolute differences), and *entropy* are calculated from the vector (see Eq. 2) as LLDs. In total, for a j -th decomposition of DWT, this procedure generates $4 \times (J_{\max} + 1)$ LLDs.

We combine the features extracted according to Eq. 1 with Eq. 2, and refer to them as wavelet energy features (WEFs) as in [21]. Subsequently, four statistical *functionals*, i.e., *maximum*, *mean*, *minimum*, and *bias* of the estimated linear regression on the frame-level features are applied to the LLDs of WPTE, and WEF. These four selected *functionals* are shown to be efficient in [21]. The *wavelet function* was selected empirically based on initial experiments, which are shown in Table 1, where the J_{\max} and dimensions of LLDs of wavelets are included as well. The wavelet function names and the decomposition scripts are based on the Wavelet Toolbox² of Matlab by MathWorks.

2.3. Temporal and Spectral Features

We use our toolkit openSMILE [22] to extract the large scale temporal and spectral features, which has been proven to be efficient on the ASC task in DCASE2016 [3]. In this study, we chose the INTERSPEECH ComParE feature set [30]. This feature set contains the ‘usual suspects’ of most popular acoustic features like *mel-frequency cepstral coefficient* (MFCC), *root mean square* (RMS)

²<http://www.mathworks.com/products/wavelet/>

Table 2: COMPAR-E acoustic feature set: 65 low-level descriptors (LLDs). MFCC: Mel-Frequency Cepstral Coefficient; RASTA: Relative Spectral Transform; HNR: Harmonics to Noise Ratio; RMS: Root Mean Square. Refer to [31] for more details.

| 55 spectral LLDs | Group |
|--|---------------|
| MFCC 1–14 | Cepstral |
| Psychoacoustic sharpness, harmonicity | Spectral |
| RASTA-filt. aud. spect. bds. 1–26 (0–8 kHz) | Spectral |
| Spectral energy 250–650 Hz, 1 k–4 kHz | spectral |
| Spectral flux, centroid, entropy, slope | Spectral |
| Spectral Roll-Off Pt. 0.25, 0.5, 0.75, 0.9 | Spectral |
| Spectral variance, skewness, kurtosis | spectral |
| 6 voicing related LLDs | Group |
| F_0 (SHS and Viterbi smoothing) | Prosodic |
| Probability of voicing | Voice quality |
| $\log HNR$, jitter (local and δ), shimmer (local) | Voice quality |
| 4 energy related LLDs | Group |
| RMS energy, zero-crossing rate | Prosodic |
| Sum of auditory spectrum (loudness) | Prosodic |
| Sum of RASTA-filtered auditory spectrum | Prosodic |

energy, *harmonics to noise ratio* (HNR), and others. The LLDs are presented in Table 2, and some statistical *functionals* are applied to these LLDs, which generate 6 373 features in total from one audio sample. Details of the features can be found in [31].

2.4. Support Vector Machines

As a popular classifier, SVMs [23] are chosen as the baseline learning models in our experiments. Both the wavelet features and the ComParE features are fed into SVM models in the format as *functionals*. The original feature values are standardised before the training phase, and the information of the *training sets* are applied to the *test sets*.

2.5. Gated Recurrent Neural Networks

Zöhrer et al. introduced gated recurrent neural networks (GRNNs) for the ASC task in [32]. GRNNs are built with blocks of gated recurrent units (GRUs, see Figure 1) [24], which is a simple alternative to long short term memory networks (LSTMs) [33]. GRNNs and LSTMs share the common characteristic of learning temporal information from the sequence as recurrent neural networks (RNNs) [34]. In particular, GRNNs need fewer parameters than LSTMs, when reaching a comparable performance. Figure 1 shows the flow diagram of one GRU in which the z , and r are *update*, and *reset* gates, governing the network to learn temporal information from an input sequence. Details on GRNNs can be found, e.g., in [24].

When feeding features to GRNNs, our first step is to segment the audio file (of 10 seconds) into *episodes* (of 1 second) sequenced by time steps of 0.5 seconds. Then, the features are extracted from the *episodes* in the format of *functionals*. Finally, features (standardised) are fed into GRNNs as the same index in the sequence of *episodes*.

2.6. Decision Fusion

To improve the efficiency and robustness of our proposed systems, we use a decision fusion process (see Figure 2) for combining mod-

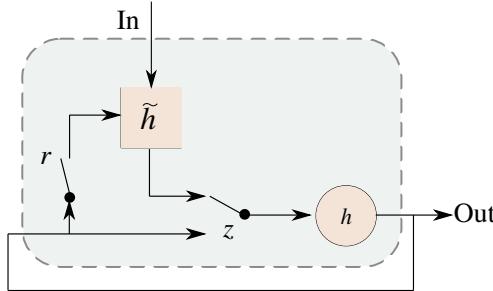


Figure 1: Diagram of a Grated Recurrent Unit (refer to [24] for more details).

els trained independently on varied feature sets. A *margin sampling value* (MSV) is defined as the difference between the first and second largest *posteriori probabilities* estimated by the trained classifier for the given test sample [25]. The final label will be given by the model which has the maximum *MSV* for the given test sample, which means this model is more ‘confident’ than others when making this decision.

3. EXPERIMENTS

3.1. Setup

The SVM models are implemented by the toolkit LIBSVM [35]. We select the SVMs with a *linear kernel*, and the complexity value C is optimised by searching within the grids spanned by $10^{-5}, 10^{-4}, 10^3, \dots, 10^3, 10^4, 10^5$. The GRNNs models are implemented by TensorFlow³. We use a two-layer (120-60) GRNNs structure, and empirically set the *learning rate*, the *drop out rate*, and the *epoch* as 0.0002, 0.1, and 50 respectively. The LLDs are extracted from the frame-level of the audio signals within a frame length of 40 ms, and an overlap of 20 ms as the set in the official baseline [1]. We combine predictions given independently using different features; the final decision is made by considering the *margin sampling* strategy mentioned in Section 2.6.

3.2. Results

We can see from both Tables 3 and 4 that, wavelet features (WPTE, WEF) are comparable to ComParE features for the ASC task in this study. Among wavelets, WEFs perform slightly better than WPTE (77.8 % vs 75.7 % on SVMs, and 76.0 % vs 72.6 % on GRNNs). Furthermore, by combining the models trained by wavelets, the final performance can be improved. In particular, by a late fusion of ComParE features with wavelet features, both models (SVMs and GRNNs) can reach an averaged accuracy of more than 81.0 %, which significantly ($p < 0.05$, one-tailed z-test [36]) outperforms the best performance (77.9 %) achieved by the model (SVMs) trained with a single feature set (ComParE). SVMs lead to a slightly better performance than GRNNs considering the overall best performance (83.2 % vs 82.6 %). Both methods considerably outperform the official baseline (74.8 %) at a significance level of $p < 0.001$ in a one-tailed z-test.

Table 5, and Table 6 show the confusion matrices of the two best-performing systems using SVMs, and GRNNs respectively. It

Table 3: Performance comparison obtained by different feature sets applied to the original segments (of 10 seconds). Classifier: Support Vector Machines (SVMs) with *linear kernel*. C -value is set to 0.01, 10 and 0.1 for ComParE, WPTE, and WEF, respectively. All the models are trained independently, and combined to make the final decision by *margin sampling values* generated by each model.

| accuracy [%] | Fold1 | Fold2 | Fold3 | Fold4 | Mean |
|------------------|-------|-------|-------|-------|-------------|
| ComParE | 76.8 | 76.8 | 75.7 | 82.5 | 77.9 |
| WPTE | 76.1 | 75.9 | 72.8 | 78.3 | 75.7 |
| WEF | 79.9 | 79.0 | 75.2 | 77.1 | 77.8 |
| ComParE+WPTE | 80.6 | 82.3 | 79.9 | 85.5 | 82.1 |
| ComParE+WEF | 82.3 | 83.9 | 81.7 | 83.7 | 82.9 |
| WPTE+WEF | 80.1 | 79.8 | 76.4 | 80.0 | 79.1 |
| ComParE+WPTE+WEF | 82.4 | 83.9 | 81.7 | 84.7 | 83.2 |

Table 4: Performance comparison between different feature sets (*Sequentially Learnt*). Classifier: Gated Recurrent Neural Networks (GRNNs). The GRNNs are structured as two-layer (120-60) topology, *learning rate*: 0.0002, *drop out rate*: 0.1, *epoch*: 50 for both ComParE, WPTE, and WEF. All models are trained independently, and combined to make the final decision by *margin sampling values* generated by each model.

| accuracy [%] | Fold1 | Fold2 | Fold3 | Fold4 | Mean |
|------------------|-------|-------|-------|-------|-------------|
| ComParE | 79.3 | 74.8 | 77.0 | 81.0 | 78.0 |
| WPTE | 73.6 | 71.8 | 71.1 | 74.1 | 72.6 |
| WEF | 77.7 | 76.6 | 73.1 | 76.8 | 76.0 |
| ComParE+WPTE | 82.1 | 79.0 | 80.1 | 84.8 | 81.5 |
| ComParE+WEF | 83.2 | 81.2 | 81.3 | 84.7 | 82.6 |
| WPTE+WEF | 78.5 | 77.2 | 74.3 | 77.6 | 76.9 |
| ComParE+WPTE+WEF | 82.6 | 81.8 | 81.0 | 85.0 | 82.6 |

is common for both SVMs and GRNNs that, some acoustic scenes like *office* and *metro station* are recognised with a high accuracy, while others like *park*, *residential area*, and *train* are difficult to be distinguished. The SVMs considerably outperform the GRNNs in classifying *city centre*, *park*, *residential area*, and *train* while the GRNNs perform better at classifying *beach* than the SVMs.

Note that, in our experiments, we intended to combine the best two models, i.e., SVMs and GRNNs trained with wavelets and ComParE features. However, the result (73.1 %) is below the achieved best performances – in fact even lower than the official baseline. One of the possible future directions is to find an efficient way to fuse the various well-trained models preserving their strengths more efficiently.

Overall, the proposed wavelet features can help improve the final recognition performance of the trained models. Both the two learning models (SVMs and GRNNs) were found to be efficient on the ASC task.

4. CONCLUSION

We found our proposed wavelet features can perform well, and help to improve the performance of typical acoustic features in classification of different acoustic scenes. Popular SVMs, and the state-of-the-art GRNNs were compared as learning algorithms. In this work, SVMs slightly outperformed GRNNs by measuring the best averaged accuracy of 4-fold cross validation on the DCASE 2017 development set (83.2 % vs 82.6 %). Both the best models significantly outperformed the official baseline (an averaged accuracy of

³<https://www.tensorflow.org/>

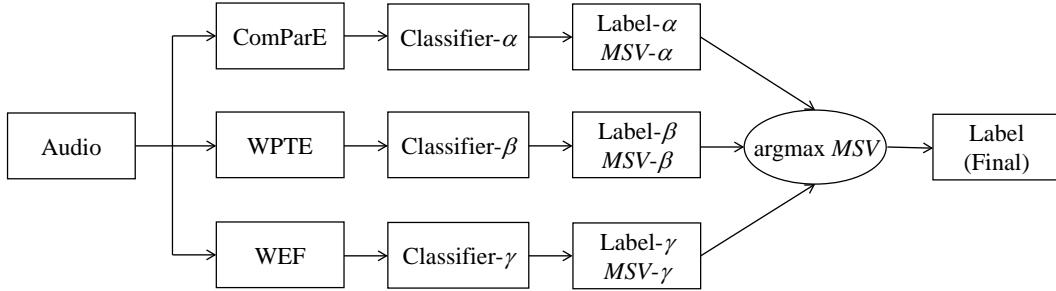


Figure 2: The diagram of a decision fusion process. The input audio files fed to SVMs, and GRNNs are original segments (of 10 seconds), and episodes (of 1 second) rowed as a sequence with time step of 0.5 second, respectively.

Table 5: Confusion matrix of the development set by decision fusion of SVMs models trained independently on the ComParE, WPTE, and WEF feature sets. Values are averaged by 4-fold cross validation on development sets.

| <i>Predicted -></i> | beach | bus | cafe/rest. | car | city cent. | forest path | groc. store | home | library | metro st. | office | park | resid. area | train | tram |
|------------------------|-----------|-----------|------------|-----------|------------|-------------|-------------|-----------|-----------|-----------|-----------|-----------|-------------|-----------|------|
| beach | 61 | 0 | 0 | 1 | 4 | 2 | 1 | 1 | 0 | 0 | 0 | 3 | 6 | 0 | 1 |
| bus | 0 | 71 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 3 | |
| cafe/rest. | 0 | 0 | 63 | 0 | 0 | 0 | 3 | 5 | 1 | 2 | 2 | 1 | 1 | 1 | |
| car | 0 | 1 | 0 | 71 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 5 | |
| city cent. | 0 | 0 | 1 | 0 | 73 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | |
| forest path | 1 | 0 | 1 | 0 | 0 | 66 | 0 | 2 | 0 | 1 | 2 | 1 | 5 | 1 | 0 |
| groc. store | 1 | 0 | 3 | 0 | 0 | 0 | 65 | 3 | 0 | 6 | 0 | 0 | 1 | 0 | 0 |
| home | 2 | 0 | 1 | 0 | 0 | 2 | 0 | 57 | 14 | 1 | 4 | 0 | 0 | 0 | 0 |
| library | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 5 | 61 | 2 | 3 | 0 | 3 | 3 | 0 |
| metro st. | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 2 | 73 | 1 | 0 | 0 | 0 | 0 |
| office | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 75 | 0 | 0 | 0 | 0 |
| park | 2 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 57 | 13 | 1 | 0 |
| resid. area | 3 | 0 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 11 | 59 | 0 | 0 |
| train | 2 | 3 | 3 | 1 | 3 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 57 | 8 |
| tram | 2 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 | 67 | |

74.8 %, $p < 0.001$, one-tailed z-test). Some acoustic scenes, e.g., *park*, *residential area*, and *train* are difficult to be distinguished in our study. In future works, we will focus on feature selection and enhancement.

5. ACKNOWLEDGMENT

This work was partially supported by the China Scholarship Council (CSC), the European Union's Seventh Framework under grant agreements No. 338164 (ERC StG iHEARu), and the EU's Horizon 2020 Programme through the Innovation Action No. 645094 (SEWA).



6. REFERENCES

- [1] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge

Table 6: Confusion matrix of the development set by decision fusion of GRNN models trained independently on ComParE, WPTE, and WEF feature sets. Values are averaged by 4-fold cross validation on development sets.

| <i>Predicted -></i> | beach | bus | cafe/rest. | car | city cent. | forest path | groc. store | home | library | metro st. | office | park | resid. area | train | tram |
|------------------------|-----------|-----------|------------|-----------|------------|-------------|-------------|-----------|-----------|-----------|-----------|-----------|-------------|-----------|------|
| beach | 68 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 4 | 0 | 1 |
| bus | 0 | 74 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
| cafe/rest. | 0 | 0 | 59 | 0 | 1 | 0 | 6 | 5 | 1 | 1 | 2 | 0 | 0 | 0 | 3 |
| car | 0 | 1 | 0 | 74 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 |
| city cent. | 0 | 0 | 0 | 0 | 66 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 8 | 0 | 0 |
| forest path | 1 | 0 | 1 | 0 | 0 | 66 | 0 | 2 | 1 | 5 | 1 | 0 | 0 | 0 | 0 |
| groc. store | 1 | 0 | 3 | 0 | 0 | 0 | 65 | 3 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| home | 2 | 0 | 1 | 0 | 0 | 2 | 0 | 57 | 14 | 1 | 4 | 0 | 0 | 0 | 0 |
| library | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 5 | 61 | 2 | 3 | 0 | 3 | 3 | 0 |
| metro st. | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 2 | 73 | 1 | 0 | 0 | 0 | 0 |
| office | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 75 | 0 | 0 | 0 | 0 |
| park | 2 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 57 | 13 | 1 | 0 |
| resid. area | 3 | 0 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 11 | 59 | 0 | 0 |
| train | 2 | 3 | 3 | 1 | 3 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 57 | 8 | |
| tram | 2 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 | 67 | |

setup: Tasks, datasets and baseline system,” in *Proc. DCASE Workshop*, Munich, Germany, 2017, in press.

- [2] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE Press, 2006.
- [3] 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. DCASE Workshop*, Budapest, Hungary, 2016, pp. 65–69.
- [4] M. Bugalho, J. Portelo, I. Trancoso, T. Pellegrini, and A. Abad, “Detecting audio events for semantic video search,” in *Proc. INTERSPEECH*, Brighton, UK, 2009, pp. 1151–1154.
- [5] 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,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

- [6] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *Proc. WASPAA*, New Paltz, NY, US, 2015, no pagination, 5 pages.
- [7] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.
- [8] 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. WASPAA*, New Paltz, NY, US, 2013, no pagination, 4 pages.
- [9] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. EUSIPCO*, Budapest, Hungary, 2016, pp. 1128–1132.
- [10] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.
- [11] 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," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.
- [12] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. EUSIPCO*, Aalborg, Denmark, 2010, pp. 1272–1276.
- [13] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 23, no. 1, pp. 142–153, 2015.
- [14] B. Elizalde, A. Kumar, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, "Experiments on the DCASE Challenge 2016: Acoustic scene classification and sound event detection in real life recording," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 20–24.
- [15] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, "Deep neural network baseline for DCASE Challenge 2016," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 50–54.
- [16] Y. Xu, Q. Huang, W. Wang, and M. D. Plumbley, "Hierarchical learning for DNN-based acoustic scene classification," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 105–109.
- [17] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 95–99.
- [18] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 11–15.
- [19] K. Qian, C. Janott, Z. Zhang, C. Heiser, and B. Schuller, "Wavelet features for classification of vote snore sounds," in *Proc. ICASSP*, Shanghai, China, 2016, pp. 221–225.
- [20] K. Qian, C. Janott, J. Deng, C. Heiser, W. Hohenhorst, M. Herzog, N. Cummins, and B. Schuller, "Snore sound recognition: on wavelets and classifiers from deep nets to kernels," in *Proc. EMBC*, Jeju, Korea, 2017, pp. 3737–3740.
- [21] K. Qian, C. Janott, V. Pandit, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "Classification of the excitation location of snore sounds in the upper airway by acoustic multi-feature analysis," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 8, pp. 1731–1741, 2017.
- [22] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in opensmile, the munich open-source multimedia feature extractor," in *Proc. ACM MM*, Barcelona, Catalunya, Spain, 2013, pp. 835–838.
- [23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014, 9 pages.
- [25] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," in *Proc. IDA*, Cascais, Portugal, 2001, pp. 309–318.
- [26] Z. Ren, V. Pandit, K. Qian, Z. Yang, Z. Zhang, and B. Schuller, "Deep sequential image features on acoustic scene classification," in *Proc. DCASE Workshop*, Munich, Germany, 2017, in press.
- [27] R. R. Coifman, Y. Meyer, and V. Wickerhauser, "Wavelet analysis and signal processing," in *Wavelets and their Applications*. Sudbury: MA: Jones and Barlett, 1992, pp. 153–178.
- [28] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Burlington, MA, USA: Elsevier, 2009.
- [29] R. N. Khushaba, S. Kodagoda, S. Lal, and G. Dissanayake, "Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 121–131, 2011.
- [30] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism," in *Proc. INTERSPEECH*, Lyon, France, 2013, pp. 148–152.
- [31] F. Eyben, *Real-time speech and music classification by large audio feature space extraction*. Switzerland: Springer International Publishing, 2015.
- [32] M. Zöhrer and F. Pernkopf, "Gated recurrent networks applied to acoustic scene classification," in *Proc. DCASE Workshop*, Budapest, Hungary, 2016, pp. 115–119.
- [33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [34] D. P. Mandic and J. A. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. West Sussex, PO19 1UD, England: John Wiley & Sons, Ltd, 2001.
- [35] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [36] M. R. Spiegel, J. J. Schiller, R. A. Srinivasan, and M. LeVan, *Probability and Statistics*. New York, NY, USA: McGraw-Hill, 2009.

DEEP SEQUENTIAL IMAGE FEATURES FOR ACOUSTIC SCENE CLASSIFICATION

Zhao Ren^{1,2}, Vedhas Pandit^{1,2}, Kun Qian^{1,2,3}, Zijiang Yang^{1,2}, Zixing Zhang², Björn Schuller^{1,2,4}

¹Chair of Embedded Intelligence for Health Care & Wellbeing, Universität Augsburg, Germany

²Chair of Complex & Intelligent Systems, Universität Passau, Germany

³MISP Group, MMK, Technische Universität München, Germany

⁴GLAM – Group on Language, Audio & Music, Imperial College London, UK

Zhao.Ren@informatik.uni-augsburg.de, schuller@ieee.org

ABSTRACT

For the Acoustic Scene Classification task of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE2017), we propose a novel method to classify 15 different acoustic scenes using deep sequential learning, based on features extracted from Short-Time Fourier Transform and scalogram of the audio scenes using Convolutional Neural Networks. It is the first time to investigate the performance of *bump* and *morse* scalograms for acoustic scene classification in an according context. First, segmented audio waves are transformed into a spectrogram and two types of scalograms; then, ‘deep features’ are extracted from these using the pre-trained VGG16 model by probing at the fully connected layer. These representations are then fed into Gated Recurrent Neural Networks for classification separately. Predictions from the three systems are finally combined by a margin sampling value strategy. On the official development set of the challenge, the best accuracy on a four-fold cross-validation setup is 80.9%, which increases by 6.1% when compared with the official baseline ($p < .001$ by one-tailed z-test).

Index Terms— Audio Scene Classification, Deep Sequential Learning, Scalogram, Convolutional Neural Networks, Gated Recurrent Neural Networks

1. INTRODUCTION

As a sub-field of computational auditory scene analysis (CASA) [1], acoustic scene classification attempts to identify the acoustic environment. It has been used in several applications such as context-aware computing [2], mobile robots [3], serious games [4] and many more. This year’s scene classification task of the IEEE AASP Challenge – DCASE2017 [5] – provides a unique opportunity to present models and audio feature representations customised for this task. The challenge requires the participants to classify the audio data into fifteen classes based on the acoustic scene they represent. The corpus has been divided into a non-public evaluation set and four-folds, each featuring training set and development set.

Different from the representations extracted from 1D audio samples directly, such as energy, frequency, and voice-based features [6], features extracted from 2D spectrograms recently show significant improvement in music [7], snore sound [8], and acoustic scene classification [9]. Those methods mostly extract Short-Time Fourier Transformation (STFT) spectrograms from audio waves. In contrast, wavelet transformation incorporates multiple scales and localisation as an extension of Fourier transform to reach the optimum of the time-frequency resolution trade-off. Wavelet fea-

tures have been shown to be efficient in snore sound classification [10–12] and acoustic scene classification [13] recently. Motivated by this success, we additionally investigate and present the capability of the deep feature representations of two types of scalograms in this study for the first time to our best knowledge in combination with pre-trained deep nets for image classification.

In the recent years, Convolutional Neural Networks (CNNs) became popular in deep learning for visual recognition tasks [14] thanks to their capability of highly nonlinear mapping of input images to output labels. Several CNN structures have been presented in succession, such as AlexNet [15], VGG [16], GoogLeNet [17], and ResNet [18]. It is also worth to design CNNs for processing spectrograms in acoustic tasks [7, 9, 19]. But as CNNs trained on a large scale data set are more robust and stable than those neural networks trained on relatively smaller number of (audio) samples, it might be worthwhile to reuse such nets to extract features from the spectrogram or scalogram for acoustic or other acoustic tasks through transfer learning [20].

In the transfer learning context, feeding powerful representations from CNNs into a classifier, such as a support vector machine (SVM), could achieve good prediction results [8]. However, as acoustic scene audio waves are relatively longer than speech signals which happen in a short time, there is a limitation in learning sequence by sequence using SVM. To break this bottleneck, several models for sequential learning have been proposed, including recurrent neural networks (RNNs) [21], long short-term memory (LSTM) RNNs [22], or Gated Recurrent Neural Networks (GRNNs) [23]. LSTM RNNs and CNNs are combined in [24] to improve the classification performance. Hence, sequential learning methods will be helpful to achieve a higher performance.

The contribution of our approach for acoustic scene classification is described as follows. First, we propose to extract the sequential features from a spectrogram (STFT) and two types of scalograms, namely (*bump* wavelet [25] and *morse* wavelets [26]), by the VGG16 model [16]. Second, we connect CNNs with a sequential learning method by feeding the features into GRNNs. Finally, the predictions from the three models are fused by the margin sampling value method. To our best knowledge, very little research has been undertaken exploring deep CNN feature representations of scalograms on sequential learning in audio analysis, let alone for acoustic scene classification.

In the following, our work aims at proposing a novel approach that makes use of CNNs and GRNNs by transfer learning, as well as presenting the experimental results obtained on DCASE2017.

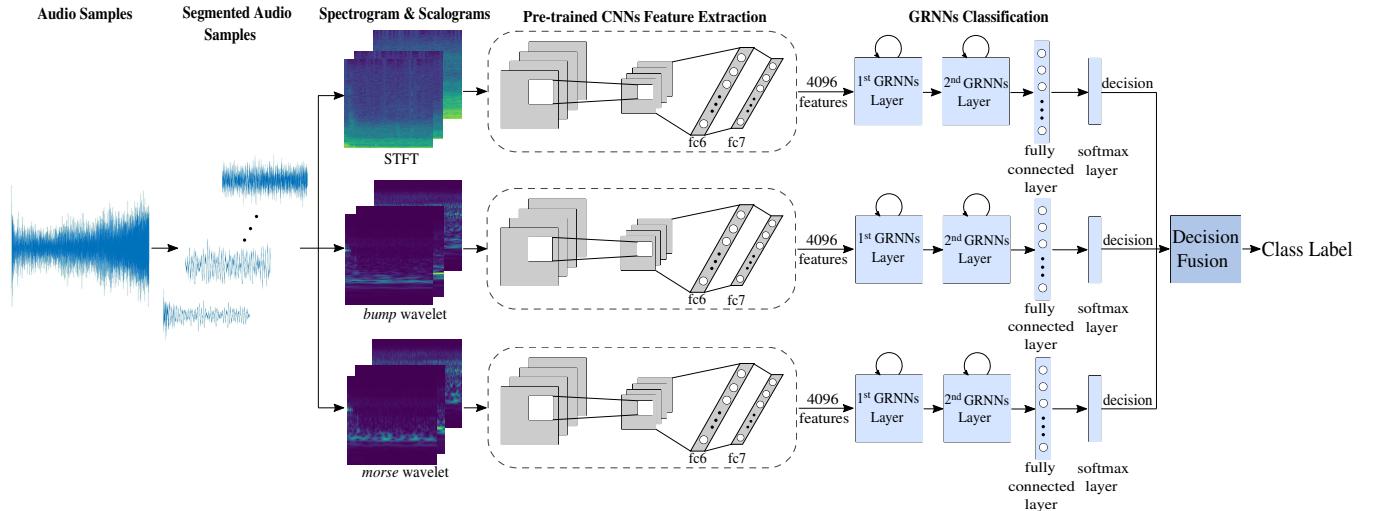


Figure 1: Framework of our proposed system. One type of spectrogram (STFT) and two types of scalograms (*bump* and *morse* wavelet) are generated from the segment audio files. Next, we use pre-trained CNNs to extract features from these images at the first fully connected layer *fc6*. After that, the features are fed into GRNNs to be trained and the final predictions are combined by a decision fusion strategy.

2. METHODOLOGY

An overview of our system can be seen in Figure 1. It mainly includes three components: deep sequential feature extraction, GRNN classification, and decision fusion, which will be introduced in this section after presenting the task.

2.1. Database

We evaluate our proposed system on the dataset of the acoustic scene classification task in the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events [5]. Each recording is split into several independent 10 s segments. The dataset contains 15 classes, including *beach*, *bus*, *cafe/restaurant*, *car*, *city center*, *forest path*, *grocery store*, *home*, *library*, *metro station*, *office*, *park*, *residential area*, *train*, and *tram*. The database is divided into an unlabelled evaluation set and four folds, each of which contains a training set and a development set. For each class, the development set contains 312 segments of 10 s from 52 minutes of audio recordings.

2.2. Deep Sequential Image Feature Extraction

2.2.1. Spectrograms

As written, we generate a STFT spectrogram and two types of scalograms, which are now described in more detail as follows.

a) STFT spectrogram. We use the STFT algorithm [27] with a Hamming window, a frame time of 40 ms and overlap of 20 ms, to compute the power spectral density by the dB power scale. At the time t , for a signal $x(t)$ with window function $\omega(t)$ and time index τ , the STFT is defined by

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t)\omega(t - \tau)e^{-j\omega t}. \quad (1)$$

b) Bump scalogram. As a special type of continuous wavelet transform (CWT), the *bump* wavelet [28] with the scale s and the window ω is defined in the Fourier domain as

$$\Psi(sw) = e^{\left(1 - \frac{1}{1 - (sw - \mu)^2/\sigma^2}\right)} \mathbf{1}_{[(\mu - \sigma)/s, (\mu + \sigma)/s]}, \quad (2)$$

where μ and σ are two constant parameters.

c) Morse scalogram. The *morse* wavelet generation is proposed in [26], in which it is defined by

$$\Psi_{P,\gamma}(\omega) = u(\omega)\alpha_{P,\gamma}\omega^{\frac{P^2}{\gamma}}e^{-\omega^{\gamma}}, \quad (3)$$

where $u(\omega)$ means the unit step, ω is the window, $\alpha_{P,\gamma}$ stands for a normalising constant, P and γ are the time-bandwidth product and the symmetry.

The spectrogram and scalograms are plotted by MATLAB using the *viridis* colour map, which was shown to be better suited than other colour maps or a gray image in [8]. It is a uniform colour map varying from blue to green to yellow. Moreover, the plots are (obviously) made to have no axes or margins, and are scaled to squared images with 224×224 pixels for VGG16-based feature extraction, as shown in Figure 2.

2.2.2. Convolutional Neural Networks

With the spectrograms and scalograms for acoustic scenes, the pre-trained CNNs are employed to extract our deep spectrum features. We use the VGG16 model provided by MatConvNet [29] as it worked successfully in the ImageNet Challenge 2014¹. VGG16 consists of 16 layers, including 13 convolutional layers, and 3 fully connected layers. The convolutional layers are split into five stacks with maxpooling layers, which use the same kernel size 3×3 and different numbers of channels [64, 128, 256, 512, 512]. The final fully connected layer is followed by a softmax function to generate a 1000-label classification on ImageNet data set. A framework of the VGG16 architecture is described in Table 1.

For our deep sequential feature extraction, the images are fed into the pre-trained VGG16 as input and the features are extracted

¹<http://image-net.org/challenges/LSVRC/2014/results>

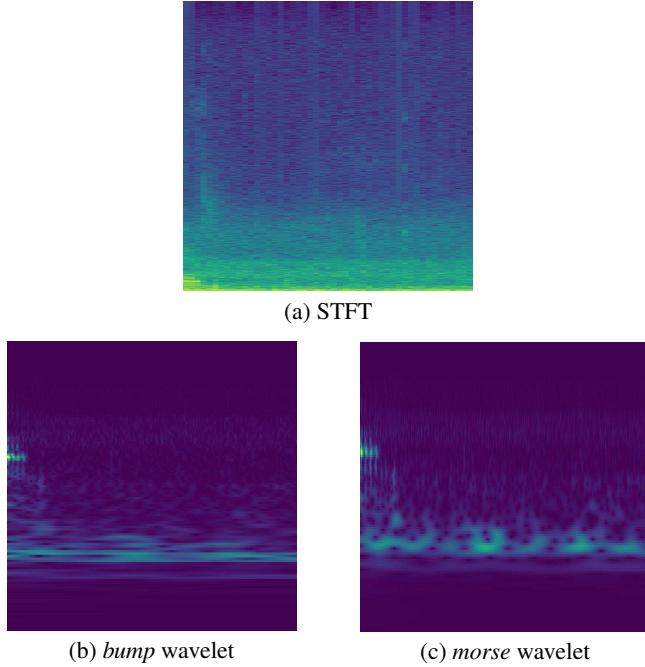


Figure 2: The STFT spectrogram and two types of scalograms for the acoustic scenes. All of the images are extracted from the first audio sequence of DCASE2017’s “a001_0_10.wav” with a label *residential area*.

from the activations on the first fully connected layer *fc6*, which includes 4096 neurons. Therefore, we extract deep features with 4096 attributes from all segmented audios.

2.3. Gated Recurrent Neural Networks

Similar to LSTM-RNNs, GRNNs are able to learn sequence information as a special type of RNN. The GRNNs contain a gated recurrent unit (GRU) [23], which consists of two gating units (reset gate r and update gate z), an activation h , and a candidate activation \tilde{h} , as shown in Figure 3. Different from LSTM, the information flows inside the GRU without separate memory cells so that a GRU needs less parameters. Hence, GRNNs converge faster than LSTM, i.e., they need less epochs during training iterations to obtain the best model [23].

Based on the deep sequence features extracted by pre-trained CNNs, we design a two-layer GRNN, which is followed by a fully connected layer and a softmax layer (see Figure 1). Therefore, we obtain the classification predictions from the softmax layer for the three different feature sets.

2.4. Decision Fusion

To improve the performance of our system, we apply a decision fusion method on the three classification results from different feature sets. The Margin Sampling Value (MSV) method is introduced in [30] as the difference of the first and second highest posterior probabilities for each predicted label of the test sample. We obtain the final label by selecting the model which has the maximum MSV, which is the most confident among the three models.

Table 1: Configurations of the VGG16 convolutional neural networks. ‘conv’ denotes convolutional layers, size means receptive field size, and ‘ch’ stands number of channels.

| | |
|---|------------|
| Input: 224×224 RGB image | |
| 2×conv size: 3; ch: 64 | Maxpooling |
| 2×conv size: 3; ch: 128 | Maxpooling |
| 3×conv size: 3; ch: 256 | Maxpooling |
| 3×conv size: 3; ch: 512 | Maxpooling |
| 3×conv size: 3; ch: 512 | Maxpooling |
| Fully connected layer <i>fc6</i> with 4096 neurons | |
| Fully connected layer <i>fc7</i> with 4096 neurons | |
| Fully connected layer with 1000 neurons | |
| Output: softmax layer of probabilities for 1000 classes | |

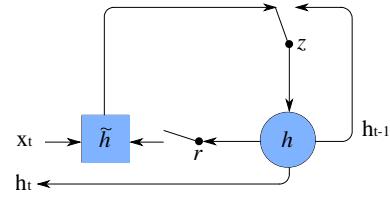


Figure 3: Illustration of a Gated Recurrent Unit (GRU) [23]. r and z represent the reset and update gates separately, and h and \tilde{h} are the activation and the candidate activation.

3. EXPERIMENTS

3.1. Setup

Each audio file is first segmented into a sequence of 19 audio samples with 50 % overlap and 1 000 msec duration. Next, we apply the pre-trained VGG16 model provided by the MATLAB toolbox MatConvNet [29] on the STFT spectrogram, *bump* and *morse* wavelet scalograms, and features are extracted from activations in the first layer *fc6* of the VGG16 model. Then, we use two-layer GRNNs (120–60), followed by a fully connected layer and a softmax layer. We implement this architecture in TensorFlow² and TFLearn³ with a fixed *learning rate* of 0.0002 (the optimiser is ‘rmsprop’), and train it for 30, 50, and 70 *epochs*. Finally, the margin sampling value decision fusion method described in Section 2.4 is selected to combine the three neural networks to obtain the final predictions.

3.2. Results

We train the three models in parallel but end at different epochs. Table 2 presents the performances of the 4-fold evaluation on the development set and their mean accuracies. We can see that, all

²<https://github.com/tensorflow/tensorflow>

³<https://github.com/tflearn/tflearn>

Table 2: Performance comparison of different epochs ($epoch \in \{30, 50, 70\}$) of GRNNs on features extracted by CNNs from STFT spectrogram, *bump*, and *morse* scalograms. The GRNNs are implemented in two layers with 120 and 60 GRU cells in each layer and a *learning rate*=0.0002.

| accuracy [%] | Fold1 | Fold2 | Fold3 | Fold4 | Mean |
|--------------------------|--------------|--------------|--------------|--------------|-------------|
| (a) STFT | | | | | |
| epoch 30 | 77.9 | 72.5 | 73.1 | 79.3 | 75.7 |
| epoch 50 | 79.2 | 74.7 | 74.3 | 77.7 | 76.5 |
| epoch 70 | 77.1 | 75.8 | 72.9 | 77.4 | 75.8 |
| (b) bump wavelet | | | | | |
| epoch 30 | 74.5 | 75.4 | 73.9 | 77.2 | 75.2 |
| epoch 50 | 73.6 | 72.9 | 73.6 | 73.2 | 73.3 |
| epoch 70 | 69.7 | 73.4 | 72.6 | 72.1 | 72.0 |
| (c) morse wavelet | | | | | |
| epoch 30 | 74.5 | 75.4 | 73.9 | 77.2 | 75.2 |
| epoch 50 | 73.6 | 72.9 | 73.6 | 73.2 | 73.3 |
| epoch 70 | 69.7 | 73.4 | 72.6 | 72.1 | 72.0 |

performances from the three models are comparable with the baseline of the DCASE2017 challenge. The results of STFT and *bump* wavelets perform slightly better than the baseline. We find that the best accuracy of each model is obtained at different epochs. For the STFT spectrogram, we observe the best performance (76.5%) at *epoch 50*, but both for *bump* and *morse* wavelet (75.2% and 72.6%) at *epoch 30*.

Therefore, we apply late-fusion to the three GRNNs results to obtain the final results, as shown in Table 3. We observe that epochs affect the performances substantially. The similarity, however, is that, the best performances in all epochs are achieved when combining results of all three models, corroborating our assumption that scalograms are efficient for acoustic scene classification. A further improvement is observed for the combination of the best epoch from each feature representation (STFT: 50, *bump*: 30, *morse*: 30): up to 80.9% accuracy with a significant improvement of 6.1% accuracy over the baseline of the DCASE2017 challenge ($p < .001$ in a one-tailed z-test [31]). Table 4 shows the confusion matrix for the best performance, combining the best epoch for the neural network of each model. Some classes, such as *beach* and *car*, are classified with high accuracies, while others, such as *park* and *residential area*, are not easy to be recognised.

To sum up, our proposed scalograms are helpful to improve the performance on acoustic scene classification, and the presented approach which connects sequence learning with pre-trained CNNs can increase the accuracy on this task.

4. CONCLUSIONS

We proposed a method for classifying acoustic scenes that relies on the ability of deep pre-trained CNNs to extract useful features from STFT and wavelet representations. Using our deep image spectrum features on GRNNs as a sequential learning method, we were able to improve the performance significantly on the official development set of the DCASE2017 challenge in a 4-fold cross validation, achieving an accuracy of 80.9% ($p < .001$ in a one-tailed z-test). In our experiments, we found that wavelet features are helpful to increase the accuracy when combining with STFT spectrogram representations. In future works, we will investigate which CNNs infer

Table 3: Performance comparison of different combinations of the three feature sets by decision fusion on the multi-class classifier GRNNs. The GRNNs are implemented in two-layers (120-60), *learning rate*=0.0002, $epoch \in \{30, 50, 70, \text{the best epoch}\}$ (STFT: 50, *bump*: 30, *morse*: 30)}. All of the models are first trained independently, and then combined to make a final decision by the MSV method.

| accuracy [%] | Fold1 | Fold2 | Fold3 | Fold4 | Mean |
|--------------|-----------------|--------------|--------------|--------------|-------------|
| epoch 30 | STFT+bump | 80.9 | 79.9 | 77.5 | 82.2 |
| | STFT+morse | 79.8 | 79.4 | 76.8 | 81.5 |
| | bump+morse | 76.7 | 77.5 | 76.0 | 77.5 |
| | STFT+bump+morse | 80.9 | 80.1 | 78.7 | 81.7 |
| epoch 50 | STFT+bump | 81.9 | 78.4 | 77.6 | 80.9 |
| | STFT+morse | 81.5 | 80.4 | 76.4 | 79.7 |
| | bump+morse | 76.7 | 77.5 | 76.0 | 77.5 |
| | STFT+bump+morse | 82.1 | 79.9 | 78.4 | 80.0 |
| epoch 70 | STFT+bump | 80.3 | 78.8 | 76.6 | 81.7 |
| | STFT+morse | 80.3 | 78.3 | 76.1 | 80.9 |
| | bump+morse | 72.8 | 75.9 | 72.5 | 76.1 |
| | STFT+bump+morse | 80.2 | 79.1 | 77.2 | 82.7 |
| best epoch | STFT+bump | 82.6 | 79.5 | 77.5 | 80.9 |
| | STFT+morse | 81.1 | 80.0 | 76.5 | 81.5 |
| | bump+morse | 76.7 | 77.5 | 76.0 | 77.5 |
| | STFT+bump+morse | 82.6 | 80.7 | 78.7 | 81.5 |

Table 4: Confusion matrix of the development set for the proposed method, in which the values are averaged in the 4-fold cross validation. Our proposed approach achieves an accuracy of 80.9%.

| | | Prediction | | | | | | | | | | | | | | |
|-------------|--|------------|-----------|-----------|-----------|------------|-------------|-------------|-----------|-----------|-----------|-----------|-----------|-------------|-----------|-----------|
| | | beach | bus | cafe/rest | car | city cent. | forest path | groc. store | home | library | metro st. | office | park | resid. area | train | |
| Actual | | | | | | | | | | | | | | | | |
| beach | | 68 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 4 | 0 | 1 |
| bus | | 0 | 74 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 1 |
| cafe/rest. | | 0 | 0 | 59 | 0 | 1 | 0 | 6 | 5 | 1 | 1 | 2 | 0 | 0 | 0 | 3 |
| car | | 0 | 1 | 0 | 74 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 |
| city cent. | | 0 | 0 | 0 | 0 | 66 | 0 | 1 | 0 | 1 | 0 | 2 | 8 | 0 | 0 | 0 |
| forest path | | 1 | 0 | 1 | 0 | 3 | 71 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| groc. store | | 1 | 0 | 5 | 0 | 0 | 0 | 61 | 0 | 2 | 6 | 1 | 1 | 0 | 0 | 1 |
| home | | 0 | 2 | 2 | 0 | 0 | 0 | 1 | 61 | 5 | 0 | 9 | 0 | 0 | 0 | 0 |
| library | | 1 | 0 | 1 | 0 | 0 | 3 | 2 | 3 | 58 | 4 | 3 | 0 | 1 | 2 | 0 |
| metro st. | | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 | 4 | 71 | 0 | 0 | 0 | 0 | 0 |
| office | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 73 | 0 | 0 | 0 | 0 |
| park | | 4 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 2 | 0 | 1 | 48 | 19 | 0 | 0 |
| resid. area | | 2 | 0 | 0 | 0 | 7 | 3 | 0 | 1 | 1 | 1 | 1 | 13 | 50 | 1 | 0 |
| train | | 0 | 7 | 3 | 2 | 8 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 3 | 45 | 8 |
| tram | | 0 | 0 | 1 | 2 | 1 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 3 | 69 |

the best representations from our audio representations, and experiment with data augmentations of the training data.

5. ACKNOWLEDGEMENTS

 This work was partially supported by the European Union’s Seventh Framework under grant agreements No. 338164 (ERC StG iHEARu), and the China Scholarship Council (CSC).

6. REFERENCES

- [1] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, “Computational auditory scene recognition,” in *Proc. of ICASSP*, vol. 2. IEEE, 2002, pp. II–1941.
- [2] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in *Proc. of WMCSA*. IEEE, 1994, pp. 85–90.
- [3] 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 ICME*. IEEE, 2006, pp. 885–888.
- [4] F. Eyben, F. Weninger, F. Groß, and B. Schuller, “Recent developments in opensmile, the munich open-source multimedia feature extractor,” in *Proc. of the ACM MM*, Barcelona, Catalonia, Spain, 2013, pp. 835–838.
- [5] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, in press.
- [6] F. Eyben, F. Weninger, F. Gross, and B. Schuller, “Recent developments in opensmile, the munich open-source multimedia feature extractor,” in *Proc. of ACM MM*. ACM, 2013, pp. 835–838.
- [7] G. Gwardys and D. Grzywczak, “Deep image features in music information retrieval,” *International Journal of Electronics and Telecommunications*, vol. 60, no. 4, pp. 321–326, 2014.
- [8] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, M. Freitag, S. Pugachevskiy, A. Baird, and B. Schuller, “Snore Sound Classification Using Image-based Deep Spectrum Features,” in *Proc. of INTERSPEECH*. Stockholm, SE: ISCA, 2017, 5 pages.
- [9] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, “Dcase 2016 acoustic scene classification using convolutional neural networks,” in *Proc. of DCASE Workshop*, 2016, pp. 95–99.
- [10] K. Qian, C. Janott, Z. Zhang, C. Heiser, and B. Schuller, “Wavelet features for classification of vote snore sounds,” in *Proc. of ICASSP*, Shanghai, China, 2016, pp. 221–225.
- [11] K. Qian, C. Janott, J. Deng, C. Heiser, W. Hohenhorst, M. Herzog, N. Cummins, and B. Schuller, “Snore sound recognition: on wavelets and classifiers from deep nets to kernels,” in *Proc. of EMBC*, 2017, pp. 3737–3740.
- [12] K. Qian, C. Janott, V. Pandit, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, “Classification of the excitation location of snore sounds in the upper airway by acoustic multi-feature analysis,” *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 8, pp. 1731–1741, 2017.
- [13] K. Qian, Z. Ren, V. Pandit, Z. Yang, Z. Zhang, and B. Schuller, “Wavelets revisited for the classification of acoustic scenes,” in *Proc. DCASE Workshop*, Munich, Germany, 2017, in press.
- [14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. of NIPS*, 2012, pp. 1097–1105.
- [16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proc. of CVPR*, 2015, pp. 1–9.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. of CVPR*, 2016, pp. 770–778.
- [19] J. Schluter and S. Bock, “Improved musical onset detection with convolutional neural networks,” in *Proc. of ICASSP*. IEEE, 2014, pp. 6979–6983.
- [20] J. Deng, N. Cummins, J. Han, X. Xu, Z. Ren, V. Pandit, Z. Zhang, and B. Schuller, “The university of passau open emotion recognition system for the multimodal emotion challenge,” in *Proc. of CCPR*. Springer, 2016, pp. 652–666.
- [21] D. P. Mandic, J. A. Chambers, et al., *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley Online Library, 2001.
- [22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [24] S. H. Bae, I. Choi, and N. S. Kim, “Acoustic scene classification using parallel combination of LSTM and CNN,” in *Proc. of DCASE Workshop*, 2016.
- [25] S. Abdullah, J. Choi, J. Giacomin, and J. Yates, “Bump extraction algorithm for variable amplitude fatigue loading,” *International Journal of Fatigue*, vol. 28, no. 7, pp. 675–691, 2006.
- [26] S. C. Olhede and A. T. Walden, “Generalized morse wavelets,” *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2661–2670, 2002.
- [27] E. Sejdić, I. Djurović, and J. Jiang, “Time–frequency feature representation using energy concentration: An overview of recent advances,” *Digital Signal Processing*, vol. 19, no. 1, pp. 153–183, 2009.
- [28] I. Daubechies, *Ten lectures on wavelets*. SIAM, 1992.
- [29] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in *Proc. of ACM MM*. ACM, 2015, pp. 689–692.
- [30] T. Scheffer, C. Decomain, and S. Wrobel, “Active hidden markov models for information extraction,” in *Proc. of IDA*. Springer, 2001, pp. 309–318.
- [31] M. R. Spiegel, J. J. Schiller, R. A. Srinivasan, and M. LeVan, “Probability and statistics,” 2009.

MULTI-TEMPORAL RESOLUTION CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION

Alexander Schindler

Austrian Institute of Technology
 Center for Digital Safety and Security
 Vienna, Austria
 alexander.schindler@ait.ac.at

Thomas Lidy, Andreas Rauber

Vienna University of Technology
 Institute of Software Technology
 Vienna, Austria
 lidy.rauber@ifs.tuwien.ac.at

ABSTRACT

In this paper we present a Deep Neural Network architecture for the task of acoustic scene classification which harnesses information from increasing temporal resolutions of Mel-Spectrogram segments. This architecture is composed of separated parallel Convolutional Neural Networks which learn spectral and temporal representations for each input resolution. The resolutions are chosen to cover fine-grained characteristics of a scene's spectral texture as well as its distribution of acoustic events. The proposed model shows a 3.56% absolute improvement of the best performing single resolution model and 12.49% of the DCASE 2017 Acoustic Scenes Classification task baseline [1].

Index Terms— Deep Learning, Convolutional Neural Networks, Acoustic Scene Classification, Audio Analysis

1. INTRODUCTION

Convolutional Neural Networks (CNN) [2] have become a popular choice in computer vision due to their ability to capture nonlinear spatial relationships which is in favor of tasks such as visual object recognition [3]. Their success has fueled interest also in audio-based tasks such as speech recognition and music information retrieval. An interesting sub-task in the audio domain is the detection and classification of acoustic sound events and scenes, such as the recognition of urban city sounds, vehicles, or life forms, such as birds [4]. The IEEE AASP Challenge DCASE is a benchmarking challenge for the “Detection and Classification of Acoustic Scenes and Events”. Acoustic Scene Classification (ASC) in urban environments (task 1) is one of four tasks of the 2016 and 2017 competition. The goal of this task is to classify test recordings into one of predefined classes that characterizes the environment in which it was recorded, for example “metro station”, “beach”, “bus”, etc. [1].

The presented approach attempts to circumvent various limitations of Convolutional Neural Networks (CNN) concerning audio classification tasks. The tasks performed by a CNN are more related to the visual computing domain. A common approach is to use Short-Term Fourier Transform (STFT) to retrieve a Spectrogram representation which is in the following interpreted as a gray-scale image. Commonly a Mel-Transform is applied to scale the Spectrogram to a desired input size. In previous work we have introduced a CNN architecture to learn timbral and temporal representations at once. This architecture takes a Mel-Spectrogram as input and reduces this information in two parallel CNN stacks towards the spectral and the temporal dimension. The combined representations are input to a fully connected layer to learn the concept

relevant dependencies. The challenge is how to choose the length of the input analysis window. Acoustic events can be single sounds or compositions of multiple sounds. Acoustic scenes could be described by the presence of a single significant acoustic event such as ship horns for harbors or by combinations of different events. The temporal pattern of such combinations varies distinctively across and within the acoustic scenes (see Figure 1 for examples of acoustic scenes). Choosing the wrong size of the analysis window can either prevent from having sufficient timbral resolution or to fail to recognize acoustic events with longer patterns.

Thus, we propose an architecture that trains on multiple temporal resolutions to harness relationships between spectral sound characteristics of an acoustic scene, and its patterns of acoustic events. This would facilitate to learn more precise representations on a high temporal scale to discriminate timbral differences such as diesel engines from trucks and petrol based engines from private cars. On the other hand, low level temporal resolutions with ranges from several seconds can optimize on different patterns of acoustic events such as a speech, steps or passing cars. Finally, the representations of the different temporal resolutions, learned by the parallel CNN stacks, are combined to form an input for a fully connected layer which learns the relationships between them to predict the acoustic scenes annotated in the dataset.

In Section 2 we will give a brief overview of related work. In Section 3 and 4 our method and the applied data augmentation methods are described in detail. Section 5 describes the evaluation setup and results while results are presented and discussed in Section 6. Finally, Section 7 summarizes the paper and provides conclusions.

2. RELATED WORK

The presented approach is based on our DCASE 2016 contribution [6] and the modified deeper parallel architecture presented in [7]. Approaches to apply CNNs and Neural Network (NN) architectures to audio analysis tasks were evaluated in [8]. The authors conclude that DNNs are not yet outperforming crafted feature-based approaches and that best performing results can be achieved through hybrid combinations. Also the leading contributions to the DCASE 2016 ASC task were not based on DNNs [9, 10, 11]. A similar data augmentation method of mixing audio files of the same class to generate new instances was applied in [12] and similar perturbations and noise induction was reported in [13]. Approaches to ASC using CNN based models were reported in [14, 15]. Combinations of CNNs with Recurrent Neural Networks (RNN) [16] have also shown promising results.

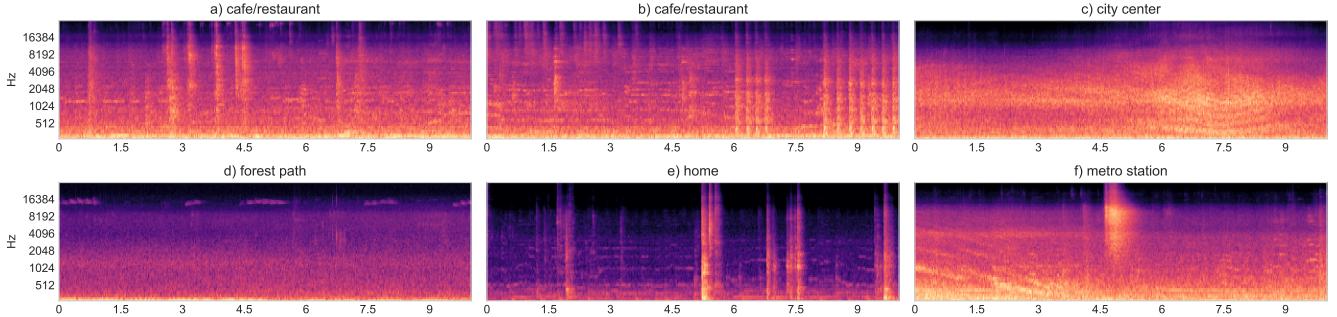


Figure 1: Example Mel-Spectrograms to visualize variances in length and shape of different acoustic events. a) dropping coins into the cash-box, b) beating coffee grounds out of the strainer, c) Doppler effect with Lloyd's mirror effect [5] of a passing car, d) chirping bird, e) opening and closing of cupboards and drawers in the kitchen, f) arriving subway with pneumatic exhaust.

3. METHOD

The presented approach analyses multiple temporal resolutions simultaneously. The design of this architecture is based on the hypothesis that acoustic scenes are composed of the spectral texture or timbre of a scene such as the low-frequent humming of refrigeration units in supermarkets as well as a sequence of acoustic events. These events can be unique for certain acoustic scenes such as the sound of breaking waves at the beach, but usually the characteristics of a scene is described by mixtures of multiple events or sounds. Spectral texture or timbre analysis requires high temporal resolutions. To distinguish the trembling fluctuations of a truck's diesel engine from a private car an analysis window of several milliseconds is required. Acoustic events, as exemplified in Figure 1, happen on a much broader temporal scale. The pattern of beating the coffee grounds out of the strainer of an espresso machine in a coffee (see Figure 1 b) requires an analysis window of 0.5 to 1 seconds. Up to 5 seconds are required for the very significant dropping sound of a decelerating Metro engine with the pneumatic exhaust of the breaks at full halt (see Figure 1 f).

Figure 2 visualizes different spectral resolutions at a fixed start-offset from audio content recorded in a residential area. Figure 2 a) visualizes the low-frequent urban background hum at a very high temporal resolution. At this level a CNN can learn a good timbre representation for acoustic scenes, but it is not able to recognize acoustic events that are longer than 476 milliseconds. Patterns such as speech (see Figure 2 c) or combinations of patterns such as people talking while a car is passing (see Figure 2 e) require much longer analysis windows, up to several seconds. The problem with single-resolution CNNs is, that a decision has to be made concerning the length and precision of the analysis window. A high temporal resolution prevents from recognizing long events while a low resolution is not able to effectively describe timbre. Increasing the size of the input segment to widen the analysis window would also increase the size of the model, its number of trainable parameters and the number of required training instances to avoid overfitting. If pooling-layers are extensively used to reduce the size of the model, the advantage of the high temporal resolution gets lost in these data-reduction steps.

Thus, we propose to use multiple inputs at different temporal resolutions to have separate CNN models learn acoustic scene representations at different scales which are finally combined to learn the categorical concepts of the acoustic scene classification dataset.

3.1. Deep Neural Network Architecture

The presented architecture consists of identical but not shared Convolutional Neural Network (CNN) stacks - one for each temporal resolution. These stacks are based on the parallel architectures initially described in [17] and further developed in [6, 7, 4]. The fully connected output layers of each parallel CNN stack, which is considered to contain the learned representation for the corresponding temporal resolution, are combined to the multi-resolution model.

The Parallel Architecture: This architecture uses a parallel arrangement of CNN layers with rectangular shaped filters and Max-Pooling windows to capture spectral and temporal relationships at once [17]. The parallel CNN stacks use the same input - a log-amplitude transformed Mel-Spectrogram with 80 Mel-bands spectral and 80 STFT frames temporal resolution. The variant used in this paper (see Figure 3b) is based on the deep architecture presented in [7]. The first level of the parallel layers are similar to the original approach [6]. They use filter kernel sizes of 10×23 and 21×10 to capture frequency and temporal relationships. To retain these characteristics the sizes of the convolutional filter kernels as well as the feature maps are sub-sequentially divided in halves by the second and third layers. The filter and Max Pooling sizes of the fourth layer are slightly adapted to have the same rectangular shapes with one part being rotated by 90° . Thus, each parallel layer subsequently reduces the input shape to 2×10 dimensions - one layer reduces the spectral while preserving the temporal information, the other performs the same reduction on the temporal axis. The final equal dimensions of the final feature maps of the parallel model paths balances their influences on the following fully connected layer with 200 units.

Multi-Temporal Resolutions CNN: The proposed architecture instantiates one parallel architecture for each temporal resolution (see Figure 3b). Their fully connected output layers are concatenated. To learn the dependencies between the sequences of spectral and temporal representations of the different temporal resolutions an intermediate fully connected layer with 512 units is added before the Softmax output layer.

Dropping out Resolution Layers: To support the final fully connected layer in learning relations between the different resolutions, a layer that has been added that drops out entire resolutions of the concatenated intermediate layer of the multi-resolution architecture.

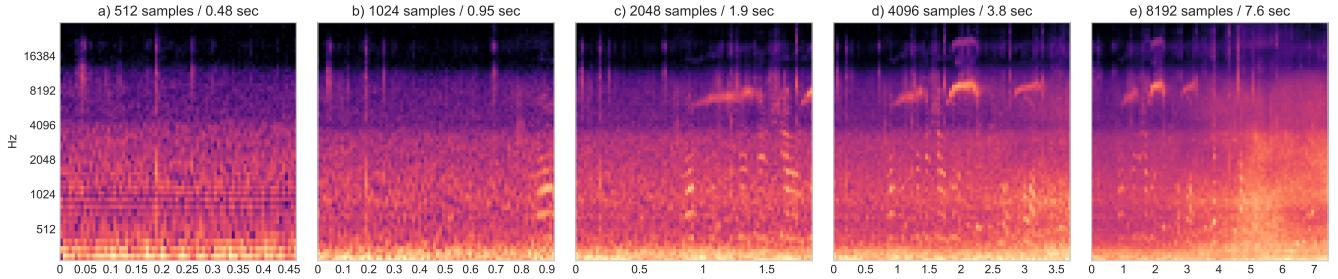


Figure 2: Input Segments for the Convolutional Neural Networks with 80 Mels spectral and five different temporal resolutions with fixed start-offset. a) spectral texture of residential area background noise, b) person saying a word (vertical wave-line), c) person talking, tweet of a bird (horizontal arc), d) person talking, bird tweeting, e) person talking, bird tweeting, car passing (light cloud to the right)

4. DATA AUGMENTATION

The most challenging characteristics of the provided dataset is its low variance. Table 1 depicts that for each class audio content of 3120 seconds length is provided. Nevertheless, this content originates from only 13 to 18 different locations per class. To create more data instances these recordings have been split into 10 seconds long audio files, but this does not introduce more variance due to very high self-similarity within a location. This low variance leads complex neural networks with a large number of trainable parameters to over-fit on the training data. Further, the limitation of 10 seconds per file prevents from using larger analysis windows. To circumvent these shortcomings data augmentation using the following methods is applied:

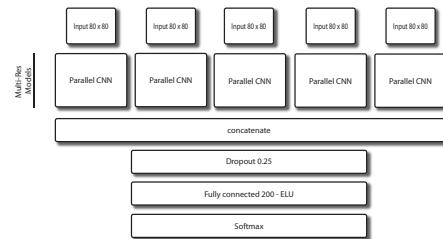
Split-Shuffle-Remix of audio files: To create additional audio content by increasing the length of an audio file its content is segmented by non-silent intervals. To create approximately 10 segments the Decibel-threshold is iteratively increased until the desired quantity is reached. These segments are duplicated to retrieve four identical copies which corresponds to 40 seconds of audio. All segments are then randomly reordered and remixed into a final combined audio file.

Remixing Places: To introduce more variance in the provided data, additional training examples are created by mixing files of the same class. Based on the assumption that classes are composed of a certain spectral texture and a set of acoustic events, mixing files of the same class would generate new recordings of this class. For each possible pairwise combination of locations within a class, a random file for each location is selected. The recordings are mixed by averaging both signals.

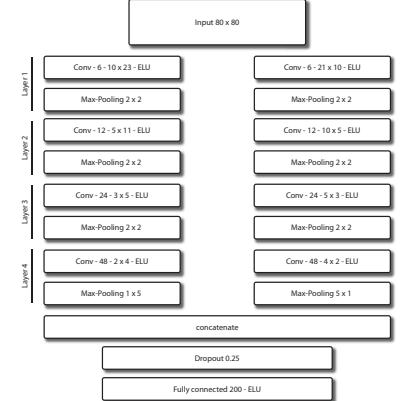
Pitch-Shifting: The pitch of the audio signal is increased or decreased within a range of 10% of its original frequency while keeping its tempo the same. The 10% range has been subjectively assessed. Larger perturbations sounded unnaturally.

Time-Stretching: The audio signal is speed up or slowed down randomly within a range of 10% at maximum of the original tempo while keeping its pitch unchanged.

Noise Layers: A data-independent augmentation method to increase the model's robustness. The input data is corrupted by adding Gaussian noise with a probability of $\sigma = 0.1$ is to the Mel-Spectrograms. The probability σ has been empirically evaluated in preceding experiments using different single-resolutions models. From the tested values [0.05, 0.1, 0.2, 0.3] a σ of 0.1 improved the model's accuracies most.



(a) Multi-Resolution Model



(b) Parallel CNN Architecture

Figure 3: The *Multi-Resolution Model* (a) which consists of one *Parallel CNN Architecture* (b) per temporal resolution.

5. EVALUATION

The presented approach was evaluated on the *development dataset* of the *TUT Acoustic Scenes 2017 dataset* [1]. The dataset consists of 15 classes representing typical urban and rural acoustic scenes (see Table 1). 4-fold cross-validation was applied using grouped stratification which preserved the class distribution of the original ground-truth assignment in the train/test splits as well as ensured that files of the same location are not split across them. The performance was measured in classification accuracy on a per-instance-level (*raw*) for every extracted Mel-Spectrogram as well as on a per-file-level (*grouped*) by calculating the average Softmax response for all Mel-Spectrograms of a file. For each audio file 10 log-

Table 1: Per class dataset Overview. Number of different locations, complete length as well as min/max/mean length of audio content.

| Label | num diff locations | Audio length (in seconds) | | | |
|------------------|--------------------|---------------------------|-----|-----|-------|
| | | sum | min | max | mean |
| beach | 17 | 3120 | 120 | 210 | 183.5 |
| bus | 18 | 3120 | 60 | 300 | 173.3 |
| cafe/restaurant | 16 | 3120 | 120 | 300 | 195.0 |
| car | 17 | 3120 | 90 | 270 | 183.5 |
| city_center | 15 | 3120 | 150 | 270 | 208.0 |
| forest_path | 18 | 3120 | 60 | 300 | 173.3 |
| grocery_store | 17 | 3120 | 120 | 270 | 183.5 |
| home | 16 | 3120 | 90 | 300 | 195.0 |
| library | 16 | 3120 | 150 | 240 | 195.0 |
| metro_station | 17 | 3120 | 90 | 300 | 183.5 |
| office | 13 | 3120 | 150 | 300 | 240.0 |
| park | 17 | 3120 | 120 | 210 | 183.5 |
| residential_area | 17 | 3120 | 120 | 240 | 183.5 |
| train | 17 | 3120 | 90 | 270 | 183.5 |
| tram | 17 | 3120 | 60 | 300 | 183.5 |

amplitude scaled Mel-Spectrograms with 80 Mels times 80 frames are extracted from the normalized input signal using random offsets and increasing FFT window sizes of 512, 1024, 2048, 4096, 8192 samples with 50% overlap. To augment the data, additional 10 random input segments were extracted for *time-stretched, pitch-shifted place-wise remixed* audio content. *Split-Shuffle-Remix* augmentation preceded all feature extraction processes. The neural networks were trained using Nadam optimization [18] with *categorical crossentropy loss* at 10^{-5} learning rate and a batch-size of 32. The learning rate was reduced by 10% if the validation loss did not improve over 3 epochs maintaining a minimum rate of $5 * 10^{-6}$.

The evaluation is divided into single- and multi-resolution experiments. First, for each of the combined model's resolutions a separate *parallel CNN model*, and second, the full multi-resolution model is evaluated. Both experiments are performed using un-augmented (*raw*) and augmented input data.

6. RESULTS AND DISCUSSION

As shown in Table 2 and Figure 4 the proposed multi-resolution model clearly outperforms the best performing single-resolution models by 3.56%. Especially the classes *train*, *metro_station*, *residential_area* and *cafe/resaturant* indicate that the model harnesses dependencies between the temporal resolutions. Although an improvement can already be observed on un-augmented (*raw*) data, the high complexity of the model especially gains from the added variance of augmented data. An interesting observation though is that the augmentation had no or a slightly degrading effect on the classes *car*, *grocery_store* and *city_center*, which seem to be unaffected or distorted by timbral and temporal perturbations or by mutual remixing. Grouping and averaging the predictions for a file of all single-resolution models (see '*grouped single*' in Table 2) does not increase the performance of these models, nor is it comparable to the multi-resolution model. It was further observed that lower temporal resolutions perform better than higher. This could indicate that the higher contrast of peaking spikes in the spectrograms makes it easier for algorithms to learn better and more discriminative representations than from the noise-like pattern of higher temporal resolutions. As already reported in preceding studies [6, 19, 7] the grouped accuracy outperforms instance based (*raw*) prediction. Averaging over multiple predicted segments of a test file balances outliers in the classification results. The custom dropout which dropped the output of two random resolution CNN stacks showed little effect on the general performance of a model. Conventional

Table 2: Experimental results (classification accuracy with standard deviation over cross-validation folds). Single-resolution model results provided on top, multi-resolution models at the bottom.

| fft win size | instance raw | grouped raw | instance augmented | grouped augmented |
|----------------|--------------|--------------|--------------------|-------------------|
| 512 | 64.14 (2.84) | 70.32 (2.96) | 69.06 (4.33) | 76.63 (4.44) |
| 1024 | 66.32 (2.58) | 71.27 (3.06) | 71.70 (5.46) | 77.06 (5.46) |
| 2048 | 66.83 (1.52) | 70.23 (1.99) | 76.24 (2.53) | 80.46 (3.30) |
| 4096 | 69.50 (2.83) | 71.92 (3.23) | 79.20 (3.03) | 81.66 (3.29) |
| 8192 | 69.66 (2.58) | 71.47 (2.95) | 82.26 (2.40) | 83.73 (2.63) |
| grouped single | | 73.12 | | 83.19 |
| multi-res | 72.23 (4.15) | 74.30 (4.81) | 85.22 (2.11) | 87.29 (2.02) |
| multi-res do | 69.39 (2.77) | 72.05 (3.26) | 82.51 (2.37) | 86.04 (3.03) |

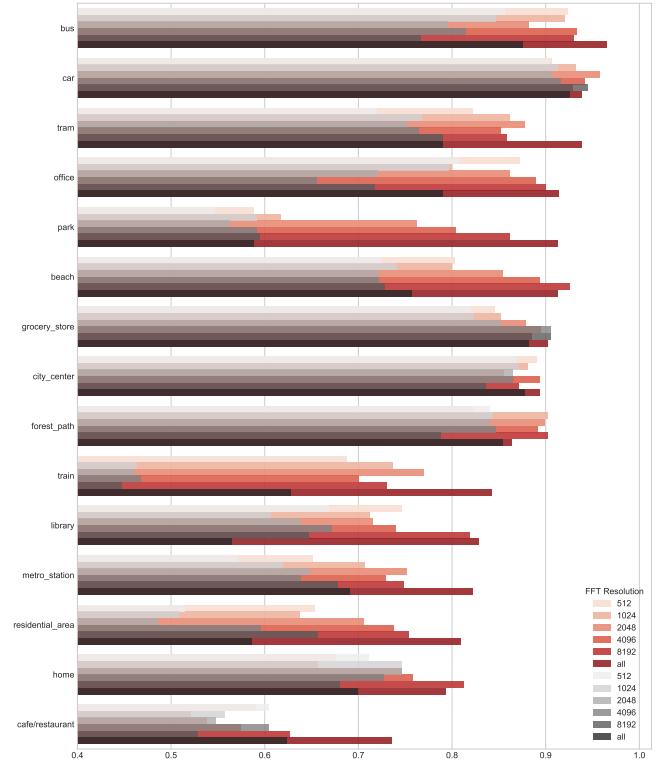


Figure 4: Results per class and FFT window size with ascending temporal resolutions. Multi-resolution results at last. Grayed bars represent un-augmented data, red bars augmented.

Dropout with a probability of $\sigma = 0.25$ seemed sufficient.

7. CONCLUSIONS AND FUTURE WORK

The presented study introduced a Convolutional Neural Network (CNN) architecture which harnesses multiple temporal resolutions to learn dependencies between timbral properties of an acoustic scene as well as its temporal pattern of acoustic events. The experimental results showed that the proposed multi-resolution model outperforms the all single-resolution and combined models by at least 3.56%. Future work will concentrate on improved data augmentation models, including evaluations on which augmentation methods have an improving/degrading effect on the classes (e.g. grocery store) and which methods can be applied to make the lower performing classes more discriminative.

8. REFERENCES

- [1] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.
- [2] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
- [3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [4] B. Fazekas, A. Schindler, T. Lidy, and A. Rauber, “A multi-modal deep neural network approach to bird-song identification,” *Working Notes of CLEF*, vol. 2017, 2017.
- [5] K. W. Lo, S. W. Perry, and B. G. Ferguson, “Aircraft flight parameter estimation using acoustical lloyd’s mirror effect,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 137–151, 2002.
- [6] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, September 2016, pp. 60–64.
- [7] A. Schindler, T. Lidy, and A. Rauber, “Comparing shallow versus deep neural network architectures for automatic music genre classification,” in *9th Forum Media Technology (FMT2016)*, vol. 1734. CEUR, 2016, pp. 17–21.
- [8] Y. M. Costa, L. S. Oliveira, and C. N. S. Jr., “An evaluation of convolutional neural networks for music classification using spectrograms,” *Applied Soft Computing*, vol. 52, pp. 28 – 38, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1568494616306421>
- [9] 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,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [10] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Supervised nonnegative matrix factorization for acoustic scene classification,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [11] S. Park, S. Mun, Y. Lee, and H. Ko, “Score fusion of classification systems for acoustic scene classification,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [12] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, “Deep convolutional neural networks and data augmentation for acoustic event detection,” *arXiv preprint arXiv:1604.07160*, 2016.
- [13] J. Schlüter and T. Grill, “Exploring data augmentation for improved singing voice detection with neural networks.” in *ISMIR*, 2015, pp. 121–126.
- [14] A. Santoso, C.-Y. Wang, and J.-C. Wang, “Acoustic scene classification using network-in-network based convolutional neural network,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [15] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, “DCASE 2016 acoustic scene classification using convolutional neural networks,” DCASE2016 Challenge, Tech. Rep., September 2016.
- [16] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, “Convolutional gated recurrent neural network incorporating spatial features for audio tagging,” *arXiv preprint arXiv:1702.07787*, 2017.
- [17] J. Pons, T. Lidy, and X. Serra, “Experimenting with musically motivated convolutional neural networks,” in *Proceedings of the 14th International Workshop on Content-based Multimedia Indexing (CBMI 2016)*, Bucharest, Romania, June 2016.
- [18] T. Dozat, “Incorporating nesterov momentum into adam,” 2016.
- [19] T. Lidy and A. Schindler, “Parallel convolutional neural networks for music genre and mood classification,” Music Information Retrieval Evaluation eXchange (MIREX 2016), Tech. Rep., August 2016.

ACOUSTIC SCENE CLASSIFICATION: FROM A HYBRID CLASSIFIER TO DEEP LEARNING

Anastasios Vafeiadis¹, Dimitrios Kalatzis¹, Konstantinos Votis¹, Dimitrios Giakoumis¹, Dimitrios Tzovaras¹, Liming Chen², Raouf Hamzaoui²

¹ Information Technologies Institute,
Center for Research & Technology Hellas, Thessaloniki, Greece
 {anasvaf, dkal, kvotis, dgiakoum, tzovaras}@iti.gr

² Faculty of Technology, De Montfort University, Leicester, UK
 {liming.chen, rhamzaoui}@dmu.ac.uk

ABSTRACT

This report describes our contribution to the 2017 Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. We investigated two approaches for the acoustic scene classification task. Firstly, we used a combination of features in the time and frequency domain and a hybrid Support Vector Machines - Hidden Markov Model (SVM-HMM) classifier to achieve an average accuracy over 4-folds of 80.9% on the development dataset and 61.0% on the evaluation dataset. Secondly, by exploiting data-augmentation techniques and using the whole segment (as opposed to splitting into sub-sequences) as an input, the accuracy of our CNN system was boosted to 95.9%. However, due to the small number of kernels used for the CNN and a failure of capturing the global information of the audio signals, it achieved an accuracy of 49.5% on the evaluation dataset. Our two approaches outperformed the DCASE baseline method, which uses log-mel band energies for feature extraction and a Multi-Layer Perceptron (MLP) to achieve an average accuracy over 4-folds of 74.8%.

Index Terms— Acoustic scene classification, feature extraction, deep learning, spectral features, data augmentation

1. INTRODUCTION

Environmental sounds hold a large amount of information from our everyday environment. Sounds can be captured unobtrusively with the help of mobile phones (MEMS microphones) or microphones (Soundman OKM II Klassik/studio A3) [1].

The process of acoustic scene classification involves the extraction of features from sound and the use of these features to identify the class of the scene.

Over the last few years, many researchers have worked on acoustic scene classification, by recognizing single events in monophonic recordings [2] and multiple concurrent events in polyphonic recordings [3]. Different approaches to feature extraction have been introduced [4], data augmentation techniques [5], use of hybrid classifiers [6] and neural networks [7] and finally comparisons between well-known classifiers and deep learning models using public datasets [8]. However, it must be noted that the problem of audio-based event recognition remains a hard task. This is because features and classifiers that work extremely well for a specific dataset may fail for another.

In this report we present two approaches for acoustic scene classification using the DCASE 2017 development dataset for training

and validation and the unlabeled DCASE 2017 evaluation dataset for testing. Our first approach combines time and frequency domain features, applies statistical analysis for dimensionality reduction, and uses a hybrid SVM-HMM for classification. Our second approach uses a CNN for classification and exploits data augmentation techniques. It differs from other CNN-based methods [9, 10] first, in that we feed the whole segment as input to the network (as opposed to splitting it in sub-sequences) and second, in that we apply max pooling to both dimensions of the input (i.e. both time and frequency). By doing that, we reduce the dimensionality of the input in a more uniform manner, thus preserving more of the segment's spatio-temporal structure, yielding more salient features with each consecutive convolutional-max pooling operation.

The remainder of the report is organized as follows. Chapter 2 describes the steps in acoustic scene classification. Chapter 3 presents the first approach using the SVM-HMM classifier and the results obtained. Chapter 4 describes the CNN model and its performance. Finally, chapter 5 concludes the report.

2. ACOUSTIC SCENE CLASSIFICATION FRAMEWORK

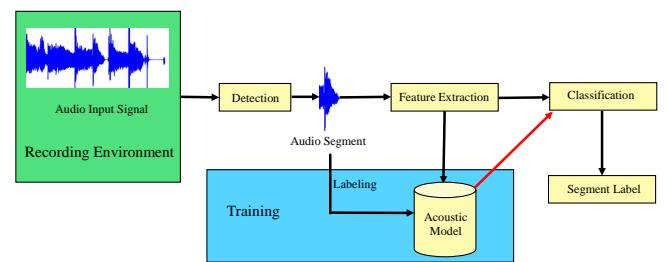


Figure 1: Typical Acoustic Scene Classification system.

Fig. 1 shows a typical Acoustic Scene Classification (ASC) system and its main components. The detection module first segments of the sound events from the continuous audio signal. Then features are extracted to characterize the acoustic information. Finally, classification matches the unknown features with an acoustic model, learnt during a training phase, to output a label for the segmented sound event.

The **Audio Input Signal** collection is the first step in the process. This step depends on the corresponding classification task.

For instance, in handwriting recognition, this step involves splitting each sentence into separate words and letters and performing other initial tasks. For sound recognition, this step involves capturing a sound from the environment and loading it into a computer. This task is typically performed using a microphone. In addition, a computer converts the analog signal to the digital format via sampling and quantization.

Feature Extraction is the second step in the process. Feature extraction involves selecting pieces of the input data that uniquely characterize that information. The choice of features depends on the application and it is based on the belief of which feature most accurately characterizes the sound.

All these levels of understanding should be combined to produce a system that is able to extract the best features. For example, a speech recognition system could use statistical techniques to identify when speech is passed into a microphone (speech/non-speech detection). Syntactical techniques could then split the speech into separate words. Each word could then be recognized and then a semantic technique could be used to interpret each word using a dictionary.

Classification is the third step in the process. For sound recognition, many techniques have been used, including Hidden Markov Models, Neural Networks and Reference Model Databases (as used with Dynamic Time Wrapping) [11]. All of these techniques use a training/testing paradigm. Training gives the system a series of examples of a particular item, so the system can learn the general characteristics of this item. Then, during testing, the system can identify the class of the item being tested.

However, classification faces one challenge. It is important to ensure that the testing and the training sets are recorded in the same conditions in order to get optimum results. In an analysis of training and testing techniques for speech recognition, Murthy, et al. [12] explains how training data must be collected from within a variety of different environments to make sure that a representative set of training data is stored in the database. They use of a filter bank to remove erroneous environmental sounds from the sound sample to ensure that these do not affect classification. Hence, robust recognition techniques are most useful if noise and other factors affect the training data.

3. PROPOSED SVM-HMM SYSTEM

In this section we describe the hybrid SVM-HMM system that was implemented using the baseline code that was provided by the organizers. We have used well-known features from the field of speech recognition and previous works in environmental sound classification.

3.1. Feature Extraction

In the feature extraction phase all audio files are transformed into the frequency domain through a 2048-sample Short-Time Fourier Transform (STFT) with 50% overlap, in order to avoid loss of information. Each frame has a window size of 40 ms with a 20 ms hop size from the next one. In our approach, we convert the 24-bit depth stereo audio recordings to mono, then the spectrum is divided into 40 mel-spaced bands, and the following features are extracted for each band: *Spectral Rolloff (SR)*, *Spectral Centroid (SC)*, *Mel-Frequency Cepstral Coefficients (MFCC)* (static, first and second order derivatives) and *Zero-Crossing Rate (ZCR)*.

For each mel band there are 12 cepstral coefficients + 1 energy coefficient, 12 delta cepstral coefficients + 1 delta energy coefficient and 12 double delta cepstral coefficients + 1 double delta energy coefficient; making a total of 39 MFCC features.

Taking the average ZCR gives a reasonable way to estimate the frequency of a sine wave. ZCR was important in recordings such as the cafe/restaurant, grocery store, metro station, tram and train, in order to separate the speech from the non-speech components.

SC and SR are defined based on the magnitude spectrum of the STFT. They measure the average frequency weighted by amplitude of a spectrum as well as the frequency below which 90% (in our case) of the magnitude distribution is concentrated.

Statistics such as the mean, variance, skewness, first and second derivatives are computed to aggregate all time frames into a smaller set of values representing each of features for every mel-band. One of the main problems is that whenever there is a large dataset, using a large number of features can slow down the training process [13]. We used the Sequential Backward Selection (SBS) [14], which sequentially constructs classifiers for each subset of features by removing one feature at a time from the previous set and finally outputs the classification error rate. The combination of all the features along with SBS increased the classification accuracy in 4-folds from 77.1% to 80.9%

Table 1 shows a comparison between our hybrid SVM-HMM approach, the DCASE2017 baseline based on Gaussian Mixture Model (GMM), using the development dataset, and the performance of our SVM-HMM system with the evaluation dataset.

Table 1: Performance comparison (averaged over 4-folds) between the DCASE2017 baseline based on GMM and our hybrid SVM-HMM approach

| Class | Baseline GMM w/ MFCC features (%) (development dataset) | Our approach SVM-HMM w/ MFCC, ZCR, SR, SC features (%) (development dataset) | Our approach SVM-HMM w/ MFCC, ZCR, SR, SC features (%) (evaluation dataset) |
|------------------|---|--|---|
| Beach | 75.0 | 78.8 | 23.1 |
| Bus | 84.3 | 90.1 | 42.6 |
| Cafe/Restaurant | 81.7 | 68.3 | 58.3 |
| Car | 91.0 | 94.2 | 66.7 |
| City center | 91.0 | 91.3 | 77.8 |
| Forest path | 73.4 | 85.6 | 86.1 |
| Grocery store | 67.9 | 80.8 | 64.8 |
| Home | 71.4 | 74.5 | 94.4 |
| Library | 63.5 | 65.7 | 39.8 |
| Metro station | 81.4 | 89.1 | 92.6 |
| Office | 97.1 | 99.0 | 54.6 |
| Park | 39.1 | 59.0 | 20.4 |
| Residential area | 74.7 | 79.8 | 72.2 |
| Train | 41.0 | 63.8 | 81.5 |
| Tram | 79.2 | 85.6 | 39.8 |
| Average | 74.1 | 80.9 | 61.0 |

3.2. Classification

The development dataset is split by the organizers in 4-folds each containing 3510 training recordings and 1170 testing recordings (75/25 split). For the training, we use the features that were mentioned in the previous section as an input to the HMM. Then, the most probable model is associated with every sequence which needs to be classified. The HMM output, which can be considered as a further refinement of the HMM input features is in turn fed to the SVM classifier in the testing phase, as it was originally proposed by Bisio et al. [16] for gender-driven emotion recognition. For the SVM, we used the Radial-Basis Function (RBF) kernel and after performing grid search, we found that the best parameters were $\sigma = 0.1$ and $C = 100$. The parameter σ of the RBF kernel handles the non-linear

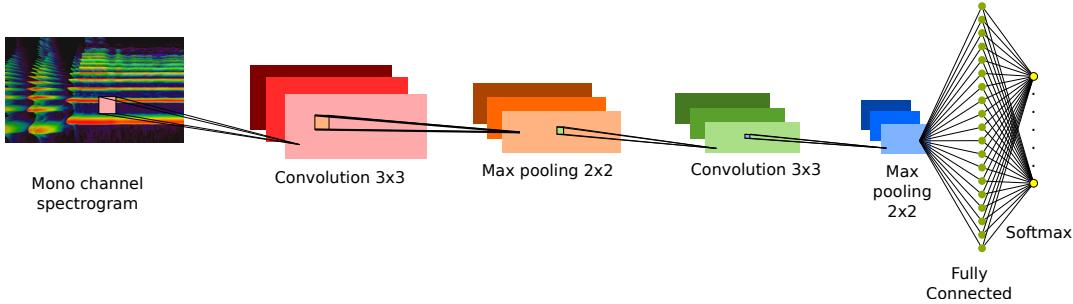


Figure 2: Block diagram of a Convolutional Neural Network.

classification and is considered to be a similarity measure between two points. C is the cost of classification.

Fig.3 shows the Receiver Operating Characteristics (ROC) curves of the SVM-HMM model. The system was not able to create a good model for classes such as: library, park, train and cafe/restaurant.

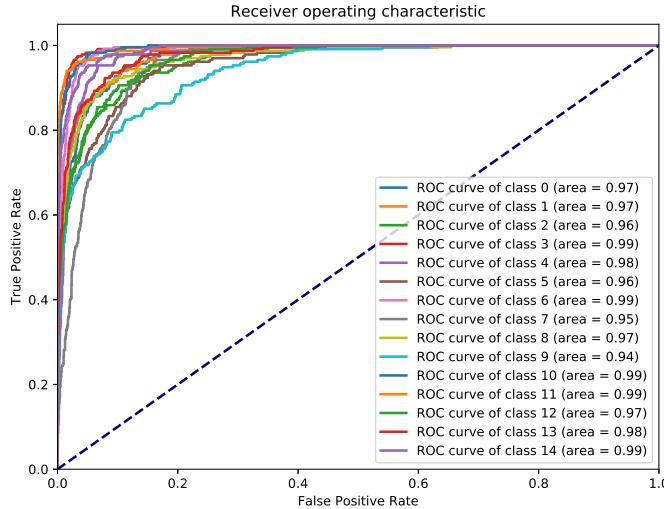


Figure 3: ROC curves of the SVM-HMM model. Classes 0-14 represent the alphabetical order of the classes from the challenge.

4. PROPOSED CNN SYSTEM

In this section we describe the CNN system that was implemented in Python using Librosa [17] for feature extraction and Keras [18] for the development of the model. The network was trained on NVIDIA GeForce GTX 1080 Ti and Tesla K40M GPUs.

4.1. Data augmentation

Environmental audio recordings have different temporal properties. Therefore, we need to make sure that we have captured all the significant information of the signal in both the time and frequency domain. Most environmental audio signals have non-stationary noise, which is often time-varying correlated and non-Gaussian.

Based on previous research [5, 19], data augmentation proved to significantly improve the total performance of the classification

system. In our approach we produced two additional augmented recordings from the original ones. Hence the total training audio files of each fold were increased from 3510 to 10530 and the testing from 1170 to 3510. For the first recording we added Gaussian noise over the 10 seconds of the recording; hence it has an average time domain value of zero. This allowed us to train our system better, since the evaluation recordings would also introduce various noises (e.g. kids playing on the beach). For the second recording we resampled the original signal from 44.1 kHz to 16 kHz. We kept the same length as the original recording and padded with zeros where necessary. We found that a lot of information at around 11 kHz was necessary for classes such as "beach" where there was a lot of noise from the wind and the sea waves.

4.2. Feature Extraction

All the recordings were converted into mono channels. In this approach, we use the mel-spectrogram with 128 bins which is a sufficient size to keep spectral characteristics while greatly reduces the feature dimension. Each frame has a window size of 40 ms with a 20 ms hop size from the next one. We normalized the values before using them as an input into the CNN network by subtracting the mean and dividing by the standard deviation.

4.3. CNN description

Our network architecture consists of 4 convolutional layers (Fig.2). In detail, the first layer performs convolutions over the spectrogram of the input segment, using 3x3 kernels. The output is fed to a second convolutional layer which is identical to the first. A 2x2 max pooling operation, then, follows the second layer and the subsampled feature maps are fed to two consecutive convolutional layers, each followed by max pooling operations. Each convolution operation is followed by batch normalization [20] of its outputs, before the element-wise application of the ELU activation function [21] to facilitate training and improve convergence time. After each max pooling operation, we apply dropout [22] with an input dropout rate of 0.2. The number of kernels in all convolutional layers is 5.

The resulting feature maps of the consecutive convolution-max pooling operations are then fed as input to a fully-connected layer with 128 logistic sigmoid units to which we also apply dropout with a rate of 0.2, followed by the output layer which computes the softmax function. Classification is, then, obtained through hard assignment of the normalized output of the softmax function. I.e.:

$$c = \arg \max_i y_i, \quad \text{for } i = 1, \dots, n \quad (1)$$

Table 2: Comparison of recognition accuracy between the proposed system and the second baseline system based on Log-mel band energies and MLP for the DCASE 2017 dataset averaged over 4-folds

| Class | Baseline Log-mel band energies | Our System (with data augmentation) Log-mel spectrogram | Our System (with data augmentation) Log-mel spectrogram |
|------------------|-------------------------------------|--|--|
| | MLP (%) (development dataset) | CNN (%) (development dataset) | CNN (%) (evaluation dataset) |
| | | | |
| Beach | 75.3 | 97.8 | 35.2 |
| Bus | 71.8 | 92.3 | 23.1 |
| Cafe/Restaurant | 57.7 | 96.2 | 58.3 |
| Car | 97.1 | 97.4 | 63.0 |
| City center | 90.7 | 99.6 | 90.7 |
| Forest path | 79.5 | 100.0 | 90.7 |
| Grocery store | 58.7 | 99.6 | 57.4 |
| Home | 68.6 | 98.3 | 61.1 |
| Library | 57.1 | 95.3 | 20.4 |
| Metro station | 91.7 | 92.3 | 38.0 |
| Office | 99.7 | 100.0 | 53.7 |
| Park | 70.2 | 90.6 | 25.9 |
| Residential area | 64.1 | 90.2 | 45.4 |
| Train | 58.0 | 93.2 | 59.3 |
| Tram | 81.7 | 97.0 | 48.1 |
| Average | 74.8 | 95.9 | 49.5 |

$$y_i = \frac{\exp x_i}{\sum_{j=1}^N \exp x_j} \quad (2)$$

where, c is the argmax-index position of each row (class) i in the set $1, \dots, N$ for which y_i is maximum and x is the net input.

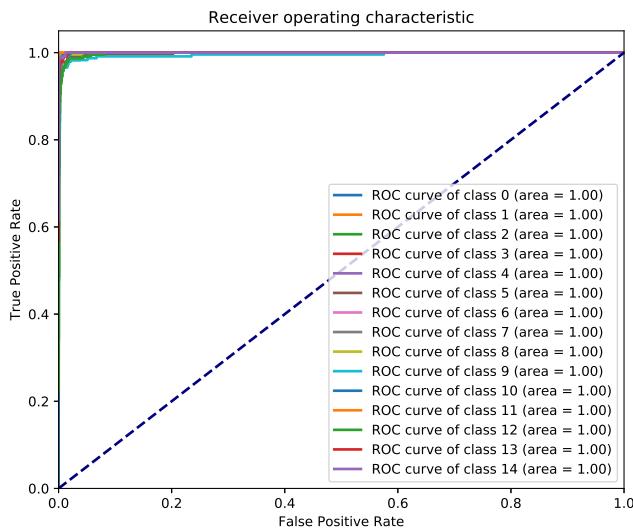


Figure 4: ROC curves of the final CNN model. Classes 0-14 represent the alphabetical order of the classes from the challenge.

Fig. 4 shows the ROC curves of our CNN model. It proves that we have a good model, as the area under the ROC curve (AUC) is approximately 0.99. Table 2 compares the classification accuracies between the baseline model and the proposed CNN model.

5. CONCLUSIONS

We presented two systems that use environmental sounds for event detection in an indoor or an outdoor environment. In order to further evaluate the performance of the proposed systems we have to test it extensively with more public datasets (i.e. UrbanSounds 8K, ESC-50, Chime Home, etc.)

Our system severely underperformed in the evaluation set, with performance dropping by almost 50%. We attribute this to a combination of inadequate feature extraction and model capacity. While our extracted features were adequate enough to encode information present in the development set (and thus lead to good development held out performance) they seem to have captured mostly local information, or at least failed to encapsulate the global structure hidden in the data. This, coupled with the relatively small capacity of our model (only 5 convolutional kernels) played a significant role in the worsening of the model's performance in the evaluation set.

We plan to explore statistical feature selection with Analysis Of Variance(ANOVA) and SBS for the CNN and compare the performance with the addition of bidirectional Long Short-Term Memory (LSTM) layers. The data augmentation technique used for the CNN will be tested with well-known classifiers. Furthermore, we will use a Variational Auto-Encoder data augmentation method, since it has proven to create robust models in the field of speech recognition [23]. Finally, tests with binaural recordings will be conducted to evaluate the performance.

6. ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 676157, project ACROSS-ING.

7. REFERENCES

- [1] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “Dcase 2017 challenge setup: tasks, datasets and baseline system.”
- [2] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, “Acoustic scene classification: Classifying environments from the sounds they produce,” *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, May 2015.
- [3] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, June 2017.
- [4] 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,” DCASE2016 Challenge, Tech. Rep., Sept. 2016.
- [5] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, Mar. 2017.
- [6] J. Liu, X. Yu, W. Wan, and C. Li, “Multi-classification of audio signal based on modified svm,” in *IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009)*, Dec. 2009, pp. 331–334.
- [7] Y. Xu, Q. Huang, W. Wang, P. Foster, S. Sigtia, P. J. B. Jackson, and M. D. Plumbley, “Unsupervised feature learning based on deep models for environmental audio tagging,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1230–1241, June 2017.
- [8] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, “A comparison of deep learning methods for environmental sound detection,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 126–130.
- [9] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Sept. 2016, pp. 60–64.
- [10] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, *DCASE 2016 Acoustic Scene Classification Using Convolutional Neural Networks*. Tampere University of Technology. Department of Signal Processing, 9 2016.
- [11] P. Khunarsal, C. Lursinsap, and T. Raicharoen, “Very short time environmental sound classification based on spectrogram pattern matching,” *Information Sciences*, vol. 243, pp. 57 – 74, 2013.
- [12] H. A. Murthy, F. Beaufays, L. Heck, and M. Weintraub, “Robust text-independent speaker identification over telephone channels,” *IEEE Transactions on Speech and Audio Processing*, vol. 7/5, Sept. 1999.
- [13] R. Murata, Y. Mishina, Y. Yamauchi, T. Yamashita, and H. Fujiyoshi, “Efficient feature selection method using contribution ratio by random forest,” in *2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, Jan. 2015, pp. 1–6.
- [14] S. Visalakshi and V. Radha, “A literature review of feature selection techniques and applications: Review of feature selection in data mining,” in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, Dec. 2014, pp. 1–6.
- [15] A. Kumar, B. Elizalde, A. Shah, R. Badlani, E. Vincent, B. Raj, and I. Lane, “DCASE challenge task 1,” DCASE2016 Challenge, Tech. Rep., Sept. 2016.
- [16] I. Bisio, A. Delfino, F. Lavagetto, M. Marchese, and A. Sciarrone, “Gender-driven emotion recognition through speech signals for ambient intelligence applications,” *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 2, pp. 244–257, Dec. 2013.
- [17] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, “librosa: Audio and Music Signal Analysis in Python,” in *Proceedings of the 14th Python in Science Conference*, Kathryn Huff and James Bergstra, Eds., 2015, pp. 18 – 25.
- [18] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [19] B. McFee, E. Humphrey, and J. Bello, “A software framework for musical data augmentation,” in *16th International Society for Music Information Retrieval Conference*, ser. ISMIR, 2015.
- [20] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.
- [21] D. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *CoRR*, vol. abs/1511.07289, 2015.
- [22] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [23] W.-N. Hsu, Y. Zhang, and J. Glass, “Learning latent representations for speech generation and transformation,” in *Interspeech*, 2017, pp. 1273–1277.

Audio Event Detection and classification using extended R-FCN Approach

Kaiwu Wang, Liping Yang, Bin Yang

Key Laboratory of Optoelectronic Technology and Systems(Chongqing University), Ministry of Education,
ChongQing University, China
 {amsturdy, yanglp, Braun}@cqu.edu.cn

ABSTRACT

In this study, we present a new audio event detection and classification approach based on R-FCN—a state-of-the-art fully convolutional network framework for visual object detection. Spectrogram features of audio signals are used as the input of the approach. The proposed approach consists of two stages like R-FCN network. In the first stage, we detect whether there are audio events by sliding convolutional kernel in time axis, and then proposals which possibly contain audio events are generated by RPN (Region Proposal Networks). In the second stage, time and frequency domain information are integrated to classify these proposals and refine their boundaries. Our approach can output the positions of audio events directly which can input a two-dimensional representation of arbitrary length sound without any size regularization.

Index Terms—audio event detection, Convolutional Neural Network, spectrogram feature

1. INTRODUCTION

Intelligent surveillance system is becoming increasingly ubiquitous in our living environment. At present, most of the video-based surveillance system is lack of robustness and reliability in practical application. For example, video-based surveillance doesn't work well in some specific scenarios, such as the night or cloudy circumstances. Audio surveillance has been alone or in combination with video surveillance to solve this problem. The work in [1] described a framework for scene analysis in a typical surveillance scenario through integrating audio and visual information. Generally, audio stream is much less onerous than video stream and the audio devices are more inexpensive. Audio events detection has been one of the important components to intelligent surveillance of security.

Unlike the static or changing slowly backgrounds in video surveillance backgrounds, there may be some impulsive sounds in audio backgrounds. Moreover, the audio signal is more versatile when audio events are superimposed on one or more backgrounds with different signal-to-noise ratio (SNR). Thus, it is a challenge work to detect audio events correctly from an audio segment.

Early works mainly concentrated on extracting different types of hand-crafted features, and training effective classifiers for recognition with traditional machine learning algorithms. The most typical classifier is Support Vector Machines (SVM). For instance, a method aimed at recognizing environmental sounds

for surveillance and security applications is presented in [2], which applies one-class support vector machines together with a sophisticated measure. Another approach is to utilize a Gaussian Mixture Model (GMM) for sounds recognition. The proposed approach in [3] first classifies a given audio frame into vocal and non-vocal events, and then performs further classification into normal and target events using GMM. The non-stationary (time-frequency) techniques are applied to sound classification and produce a good result. In [4], Jonathan Dennis et al. extract image feature from the SPD image—a novel two-dimensional representation that characterizes the spectral power distribution over time in each frequency sub-band. In [5], [6], features are extracted from the spectrogram image of sound signals for automatic sound recognition. More recently, methods based on Deep Neural Networks (DNNs) have achieved good performance for sound event classification and detection. In [12], the authors outline a sound event classification framework that compares auditory image front end features with spectrogram image-based front end features, using deep neural network classifiers. The work in [13] employs an ensemble learning framework—a stack of ensemble classifiers named multi-resolution stacking (MRS). A concatenation of lower building blocks' predictions and the expansion of the raw acoustic feature is fed into each classifier in MRS. The lower building blocks describe a base classifier in MRS, named boosted deep neural network. In [14], the authors record a database of sounds occurring in subway trains in real conditions of exploitation and use DNNs to classify the sounds into screams, shouts and other categories. In these audio events detection methods, Deep Neural Networks (DNNs) mostly work as a classifier which achieve better performance than other traditional classifiers, but can't directly detect audio events in real-time when the audio events occur.

In this paper, we present a new audio event detection and classification approach by extending the R-FCN framework [11] which is a fully convolutional neural network used in visual object detection. Inspired by the framework, the proposed approach also consists of three parts (not include the extracting of feature maps). The first part is RPN Network [10] for generating a list of proposals which possibly contain audio events. However, unlike the proposals with different widths and heights in [11], the proposals in our extended R-FCN framework have the same height and just vary in width. We do this just because the sound is one dimension signal. The second part refines the boundaries of above proposals. The third part classifies above proposals into audio events or backgrounds. In the second and third parts, we not only use the information in time domain but also use the information in different frequency ranges for classification and boundary regression. Spectrogram features of audio signals are

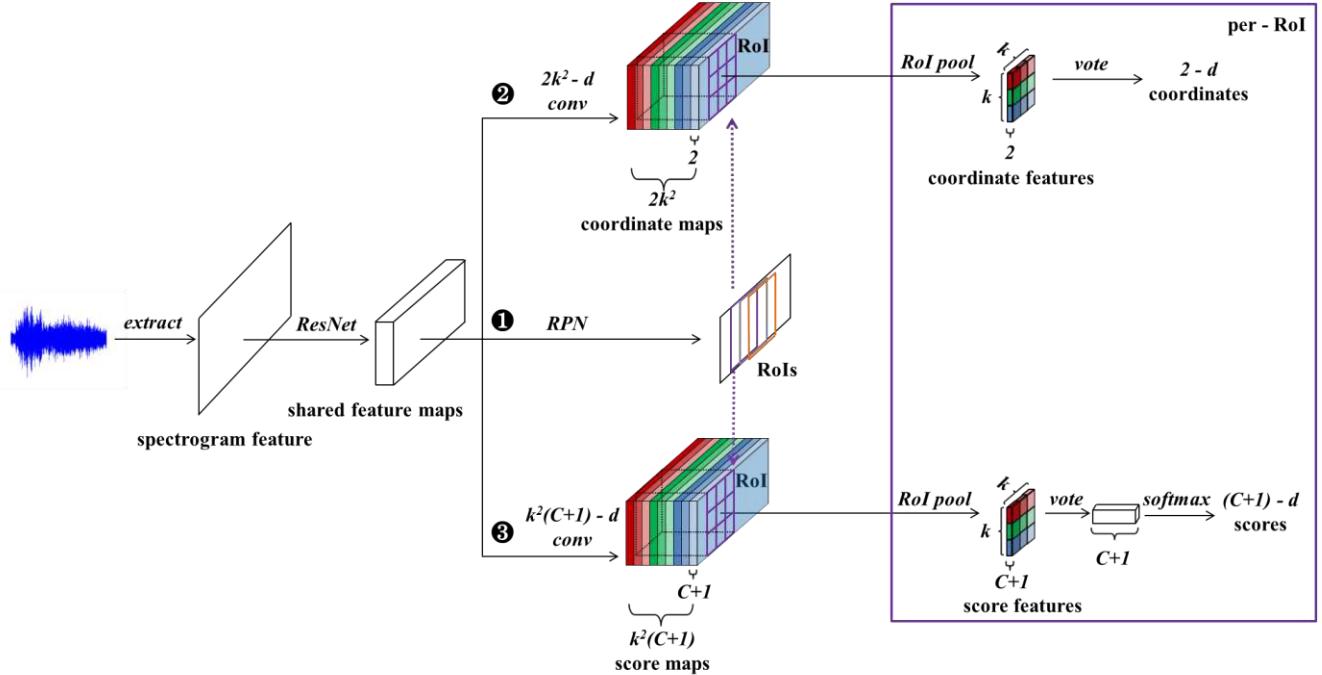


Figure 1: Overview of the system architecture for audio detection and classification

fed into the architecture as inputs. Based on the capacity of ResNet network [8] for extracting features and solving the gradient degradation problem when training a deep neural network, feature maps are produced by the modified ResNet50 network fed with these spectrogram feature. These feature maps are shared by above three parts.

The rest of this paper is organized as follows. Section 2 presents the proposed deep learning approach for audio detection and classification. Section 3 discusses our experiments and results. Section 4 concludes this work.

2. PROPOSED APPROACH

Our approach follows the deep learning framework of R-FCN which is a fully convolutional neural network for visual object detection [11]. We improve the Region Proposal Network (RPN) to make the framework suitable for audio detection. To make the overall system more simply and generate real-time result, we substitute ResNet50 for ResNet101, and modify ResNet50 network. The overall system architecture essentially consists of two stages through three parts in Figure 1. Firstly, we detect the approximate position (proposal) of audio event and don't care the audio event class by our first part. So, this can be treated as a binary classification problem. Secondly, we classify the proposal into audio events or background and refine the boundaries in our second and third parts. These proposals are also called as regions of interest (RoIs).

2.1. The Modified ResNet50 Network

Based on the capacity of ResNet50 [8] network for extracting features and solving the degradation problem when training

a deep neural network, we extract the shared feature maps from spectrogram feature by the modified ResNet50 network. Experiments show that the performance will be reduced because of the reduction of feature maps' resolution. So, we remove the last *fc* layer, pooling layer and three building blocks in ResNet50. The modified ResNet50 begins with a 7×7 convolutional layer which has 64 kernels and a stride of 2, then follows a 3×3 max pooling layer with a stride of 2, the rest of the modified ResNet50 are series of building blocks. The last convolutional layer has 1024 kernels, so the generated shared feature maps has 1024 channels.

2.2. The Improved RPN Network

For audio events detection, the improved RPN network outputs the start time, the length of proposals with scores estimating the probability that these proposals belong to audio events or background. We model this process followed a fully convolutional network [10]. The shared feature maps produced by the modified ResNet50 are fed into RPN network.

A strip window slides over the shared feature maps to generate m base regions for proposals at each position on the time axis of feature map. These m base regions are different in length but start at the same time of audio signal corresponded to the current position of sliding window. The sliding window has a size of $n \times 1$, n is the height of the shared feature maps. At each position of sliding window, 256-dimensional vector representing these m base regions simultaneously is generated, which is used to produce scores and coordinates of proposals based on these m base regions. The coordinates represent the difference of start time and length between a base region and proposal. The scores estimate the probability that these m proposals belong to audio

events or background. So, $2m$ scores and $2m$ coordinates are generated for each position of sliding window. In another word, we generate m base regions at every few frames of audio signal, and extract 256-d feature vector from front part of these m base regions to represent all m base regions simultaneously. Then, the 256-d feature vector is used for generating scores and refining the position.

Taking inspiration from the character of convolutional layer, the improved RPN network is implemented by three normal convolutional layers. The 256-d feature vector is produced by a $n \times 1$ convolutional layer which has 256 kernels and works as a sliding window. Then the 256-d vector is simultaneously fed into two sibling 1×1 convolutional layers—a classification layer (*cls* layer) and a regression layer (*reg* layer). The *cls* layer and *reg* layer have $2m$ convolutional kernels respectively. The *cls* layer outputs 2 scores that estimate the probability that the proposal belongs to audio events or background. So the *cls* layer has $2m$ outputs. The *reg* layer has $2m$ outputs representing the difference between base regions and proposals in start time and length. The key ideal of RPN network for audio detection is illustrated in the Figure 2. We use m lines with different lengths represent the m base regions in the Figure 2.

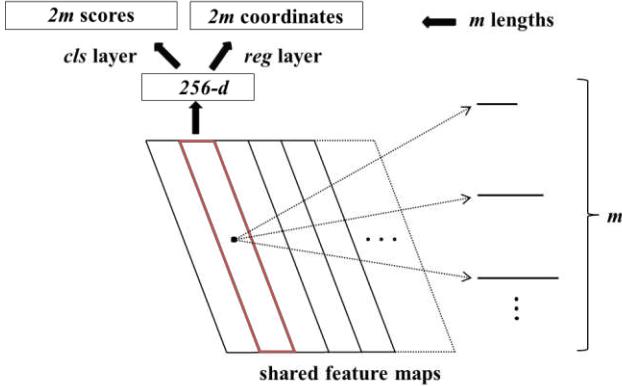


Figure 2: Key ideal of RPN network for audio detection

We use two 2-d vectors (B_s, B_l) and (G_s, G_l) denote the position of a base region and ground truth respectively. We are more concerned with the time that audio event occurs, so the 2-d position vector denotes the start and length of an audio event. We need to define an operation F to produce position vector of proposal from a base region:

$$(P_s, P_l) = F(B_s, B_l) \quad (1)$$

Here, (P_s, P_l) denote the position vector of a proposal. When the base region near to the ground truth, we take shift and scale transformation into consideration from a base region. The operation F can be simply defined as:

$$P_s = B_l * t_s + B_s, \quad P_l = B_l * \exp(t_l) \quad (2)$$

t_s, t_l is the shift factor and scale factor need to be learned, and they are the *reg* layer's output for a base region. So, the corresponding labels are defined as:

$$t_s^* = (G_s - B_s) / B_l, \quad t_l^* = \log(G_l / B_l) \quad (3)$$

When training the improved RPN network, our loss function for audio detection followed the multi-task loss in [10] which can

solve the classification and coordinates refining simultaneously, defined as:

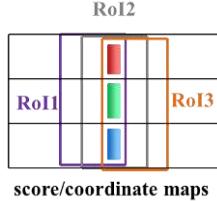
$$L(\{p_i\}, \{t_i\}) = \frac{1}{2} \sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{2} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (4)$$

Here, i is the index of base regions, P_i represents the probability that base region i is predicted as an audio event or background. We set the balance weight $\lambda=1$. The classification loss L_{cls} is log loss over two classes (audio event or background). The ground-truth label P_i^* is 1 if the base region i is an audio event, otherwise is 0. This means the regression loss L_{reg} (smooth L_1 loss function, defined in [9]) is activated only for the base region near to ground truth. t_i is a two-dimensional vector (t_s, t_l) for base region i , which is produced by *reg* layer. t_i^* is a two-dimensional vector (t_s^*, t_l^*) for base region i .

2.3. The classification and refining of proposals

After the first stage, we select 6000 proposals which are most likely to be audio events from the translations of all base regions. We do this by checking their probability to be audio event because the first stage is a binary problem. In the second stage, $k^2(C+1)$ -channel score maps are produced by a convolutional layer which has $k^2(C+1)$ kernels and $2k^2$ -channel coordinate maps are produced by a convolutional layer which has $2k^2$ kernels as seen in Figure 1. C is the categories of audio events (+1 for background), k equals 3, but which is a hyper parameter related to the parameter of the following *RoI pooling* layer. Given the 6000 proposals (RoIs), each RoI is divided into $k \times k$ parts, and $k \times k$ score / coordinate features are produced through *RoI pooling* layer from score / coordinate maps respectively. The details of the *RoI pooling* layer defined in [11]. After the *RoI pooling* layer, a vote principle (implemented by an average pooling layer) is applied to every channel of scores / coordinates features. Then the 2-d coordinates are generated, and the $(C+1)$ -dimensional scores for each category are generated after a *softmax* layer.

In the score or coordinate maps, the three red, green, blue maps represent the high, middle, low frequency components respectively, and each of these three maps in every color (red, green, blue) maps represents sequentially one third segment of a proposal in time axis correspond to the color depth, the deepest color score map represents first segment of a proposal and the lightest color score map represents the last segment. Each component of score / coordinate features is from average pooling on only one of $k \times k$ score / coordinate maps correspond to their same color. This can be understood that not only components at different time periods but also the different frequency components at same time periods have different response to the classifying or refining of a RoI. For example, there are three RoIs which have same size and one third overlap as showed in Figure 3. However, the same one third part has different response to the classifying or refining of these three RoIs showed by the lighter of every color. Moreover, different frequency components have different response to the classifying or refining of each RoI showed by the different color (red, green, blue). What's more, there are always $k \times k$ score / coordinate features regardless of the size of RoI. In another word, the sound with arbitrary length can be processed.

Figure 3: Illustration of the *RoI pooling* layer’s function

3. EXPERIMENTS

We perform experiments on the dataset of IEEE DCASE Challenge 2017 Task 2, which consists of isolated sound events for three target class (babycry, glassbreak, gunshot) and recordings of everyday acoustic scenes to serve as background. The background audio material consists of recordings from 15 different audio scenes, and is part of TUT Acoustic Scenes 2016 dataset [7]. We regularize audio signals to same sample frequency of 44.1 KHz and synthesize our training data according to the event-to-background ratio (EBR, -6.0, 0, 6.0). Our training data has 15000 mixtures (5000 per target class, each mixture only contains a target class event). Then we generate the grayscale spectrogram through short-time Fast Fourier transform. A hamming window with a length of 1024 samples and an overlap of 441 samples is applied. The generated spectrogram feature has a height of 512. We evaluate our approach on the development dataset. The mainly evaluation metrics are Error Rate (ER) and F-Score (F) calculated using event-based onset-only condition with a collar of 500ms [15].

Table 1: The results for each class on development dataset

| | <i>Class</i> | <i>F (%)</i> | <i>ER</i> | <i>DR</i> | <i>IR</i> |
|--------------|--------------|--------------|-----------|-----------|-----------|
| Baseline | babycry | 69.5 | 0.73 | 0.18 | 0.55 |
| | glassbreak | 88.1 | 0.23 | 0.17 | 0.06 |
| | gunshot | 51.2 | 0.85 | 0.56 | 0.29 |
| | All | 70.0 | 0.60 | 0.30 | 0.30 |
| Our approach | babycry | 97.2 | 0.06 | 0.03 | 0.03 |
| | glassbreak | 94.6 | 0.10 | 0.09 | 0.00 |
| | gunshot | 81.4 | 0.32 | 0.28 | 0.04 |
| | All | 91.4 | 0.16 | 0.13 | 0.02 |

Table 1 shows the result on Event-based overall metrics for each class on development dataset compared with Baseline system for DCASE Challenge 2017 Task 2. *DR*, *IR* are the Deletion Rate, Insertion Rate respectively [15]. The implementation of Baseline is based on a multilayer perceptron architecture (MLP) and uses log mel-band energies as features. The features are calculated in frames of 40 ms with a 50% overlap, using 40 mel

bands covering the frequency range 0 to 22050 Hz. The feature vector was constructed using a 5-frame context, resulting in a feature vector length of 200. The MLP consists of two dense layers of 50 hidden units each, with 20% dropout. For each of the target classes, there is a separate binary classifier with one output neuron with sigmoid activation, indicating the activity of the target class. Our approach has an *ER* value of 0.16, F-Scores of 91.4% for all classes, which outperforms the Baseline system (*ER*: 0.60, *F*: 70.0%). The *ER* values of babycry and gunshot are improved by 0.67 and 0.53 respectively. It shows great improvement for babycry and gunshot.

The result on Event-based overall metrics for each EBR on development dataset is shown in Table 2. F-Scores and *ER* for all EBRs (not includes the mixture which doesn’t contain target event) are 91.7% and 0.15. The difference of *ER* value between each *EBR* is lower than that of Baseline system, which proves our approach is more robust to noise. Our approach has lower *IR* value for each class and *EBR*, which indicates our approach is more reliable.

Table 2: The results for each EBR on development dataset

| | <i>Class</i> | <i>F (%)</i> | <i>ER</i> | <i>DR</i> | <i>IR</i> |
|--------------|--------------|--------------|-----------|-----------|-----------|
| Baseline | -6.0 | 60.0 | 0.72 | 0.45 | 0.27 |
| | 0 | 77.6 | 0.43 | 0.24 | 0.19 |
| | 6.0 | 82.7 | 0.34 | 0.20 | 0.14 |
| | All | 73.5 | 0.50 | 0.30 | 0.20 |
| Our approach | -6.0 | 86.8 | 0.23 | 0.20 | 0.02 |
| | 0 | 93.3 | 0.13 | 0.12 | 0.01 |
| | 6 | 95.2 | 0.09 | 0.08 | 0.01 |
| | All | 91.7 | 0.15 | 0.13 | 0.01 |

On the evaluation dataset, F-score and ER on segment-based metrics are 0.3173 and 82.0%, respectively. Our approach generally produced better result because of the two stage strategy. After the first stage, there will be a global summary for an audio event compared with Baseline’s strategy. The 5-frame context of Baseline system only has a local summary for an audio event. At the same time, the using of deeper network and the integration of time and frequency domain information is another reason for the better performance.

4. CONCLUSION

We present an audio event detection and classification approach which can directly output the positions of audio events from an audio signals with arbitrary length. We improved RPN network for generating proposals which possibly contain audio events. Time and frequency domain information are fully utilized to classify these proposals and refine their boundaries.

5. REFERENCES

- [1] Cristani M, Bicego M, Murino V, et al. Audio-Visual Event Recognition in Surveillance Video Sequences[J]. IEEE Transactions on Multimedia, 2007, 9(2): 257-267.
- [2] Rabaoui A, Davy M, Rossignol S, et al. Using One-Class SVMs and Wavelets for Audio Surveillance[J]. IEEE Transactions on Information Forensics and Security, 2008, 3(4): 763-775.
- [3] Atrey P K, Maddage N C, Kankanhalli M S, et al. Audio Based Event Detection for Multimedia Surveillance[C]. international conference on acoustics, speech, and signal processing, 2006: 813-816.
- [4] Dennis J, Tran H D, Chng E S, et al. Image Feature Representation of the Subband Power Distribution for Robust Sound Event Classification[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2013, 21(2): 367-377.
- [5] Sharan R V, Moir T J. Noise robust audio surveillance using reduced spectrogram image feature and one-against-all SVM[J]. Neurocomputing, 2015: 90-99.
- [6] Sharan R V, Moir T J. Subband Time-Frequency Image Texture Features for Robust Audio Surveillance[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(12): 2605-2615.
- [7] Mesaros A, Heittola T, Virtanen T. TUT database for acoustic scene classification and sound event detection[C]// European Signal Processing Conference. 2016:1128-1132.
- [8] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]. computer vision and pattern recognition, 2015: 770-778.
- [9] Girshick R. Fast R-CNN[C]. international conference on computer vision, 2015: 1440-1448.
- [10] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015: 1-1.
- [11] Dai J, Li Y, He K, et al. R-FCN: Object Detection via Region-based Fully Convolutional Networks[C]. neural information processing systems, 2016: 379-387.
- [12] Mcoughlin I V, Zhang H, Xie Z, et al. Robust sound event classification using deep neural networks[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2015, 23(3): 540-552.
- [13] Zhang X, Wang D. Boosting contextual information for deep neural network based voice activity detection[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2016, 24(2): 252-264.
- [14] Laffitte P, Sodoyer D, Tatkeu C, et al. Deep neural networks for automatic detection of screams and shouted speech in subway trains[C]. international conference on acoustics, speech, and signal processing, 2016: 6460-6464.
- [15] Mesaros A, Heittola T, Virtanen T. Metrics for polyphonic sound event detection[J]. Applied Sciences, 2016, 6(6): 162.
- [16] Waibel A, Hanazawa T, Hinton G, et al. Phoneme recognition using time-delay neural networks[J]. IEEE transactions on acoustics, speech, and signal processing, 1989, 37(3): 328-339.

ACOUSTIC SCENE CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORK AND MULTIPLE SPECTROGRAMS FUSION

Zheng Weiping¹, Yi Jiantao¹, Xing Xiaotao^{1*}, Liu Xiangtao², Peng Shaohu³

¹School of Computer, South China Normal University
Guangzhou, China

250145025@qq.com, Yijiantao@hotmail.com, 1299670261@qq.com

²Shenzhen Chinasfan Information Technology Co., Ltd.
Shenzhen, China
liuxt@12366.net

³School of Mechanical and Electrical Engineering, Guangzhou University
Guangzhou, China
pengsh@gzhu.edu.cn

ABSTRACT

Making sense of the environment by sounds is an important research in machine learning community. In this work, a Deep Convolutional Neural Network (DCNN) model is presented to classify acoustic scenes along with a multiple spectrograms fusion method. Firstly, the generations of standard spectrogram and CQT spectrogram are introduced separately. Corresponding features can then be extracted by feeding these spectrogram data into the proposed DCNN model. To fuse these multiple spectrogram features, two fusing mechanisms, namely the voting and the SVM methods, are designed. By fusing DCNN features of the standard and CQT spectrograms, the accuracy is significantly improved in our experiments, comparing with the single spectrogram schemes. This proves the effectiveness of the proposed multi-spectrograms fusion method.

Index Terms— Deep convolutional neural network, spectrogram, feature fusion, acoustic scene classification

1. INTRODUCTION

Environmental sound is a combination of sounds from many sources. It carries a lot of information that can help human to sense the surrounding environment. Acoustic scene classification (ASC) has been attracting the attention of researchers in machine learning communities and has been applied into surveillance, robotic navigation and context-aware services, etc.

Deep learning based solutions have been receiving great attentions from ASC researches. CNN[1][2], RNN[3], LSTM[1], DNN[4] and their combinations[1][3] have been applied to propose solutions. CNN has once again proved its powerful potential. In the DCASE2016 ASC challenge, a deep CNN solution[5] was proposed and won the rank first in the challenge task.

In our DCASE2017 ASC submission, we also use a deep convolutional neural network (DCNN) based method to classify the acoustic scenes. Specifically, we produce multiple spectrograms from audio files which are used to train a DCNN model. We have explored two different productions of spectrogram: standard spectrogram and Constant-Q-Transform (CQT) spectrogram[6]. According to the sliding window width and shift step length, multiple standard spectrograms with different resolutions are generated. The classification performances of the DCNN model with multi-resolution standard spectrogram and CQT spectrograms are compared respectively. Next, we use the DCNN model to extract features, instead of classifying directly. A feature fusion method is applied in our submission. We have tried the fusion of features extracted from standard spectrograms with different resolutions, as well as the fusion of CQT spectrograms combined with standard spectrograms. Among our experiments, the CQT plus standard spectrograms fusion has achieved the best performance.

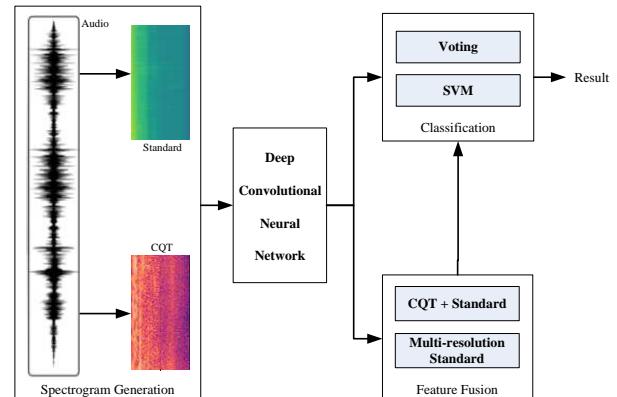


Figure 1: Flowchart of our method.

* Corresponding author

This work is partially supported by the Guangdong Provincial Scientific and Technological Projects (Grant Nos. 2016B010109005) and Characteristic Innovation Projects of the Educational Commission of Guangdong Province (Grant Nos. 2016KTSCX025)

The remainder of this paper is organized as follows. Section 2 introduce the generations of spectrograms. In Section 3, the detail of the DCNN model is given and the fusing algorithms used are described. Next, the experiment results are represented in Section 4. The submitted results are briefly explained in Section 5. Finally, we conclude the report in Section 6.

2. DATA PREPARATION

2.1. Standard Spectrogram

Instead of producing spectrograms from MFCC features[7], we generate spectrogram directly by performing Short Time Fourier Transform (STFT) on raw audio frames. Consequently, this spectrogram is referred to as “standard spectrogram”. According to different sliding window widths and shift lengths, multi-resolution standard spectrograms can be generated; related setting parameters are shown in Table 1.

Table 1: Multi-Resolution parameters

| Resolution Name | Sliding Window width | STFT | | | Bins × Freq | Num of samples |
|------------------------|----------------------|------|------|---------|-------------|----------------|
| | | NFFT | Pad | Overlap | | |
| R₅₂₉ | 24 | 529 | | 176 | 1249 × 512 | 12 × 2 |
| R₇₀₆ | 32 | 706 | 1024 | 276 | 1025 × 512 | 8 × 2 |
| R₈₈₂ | 40 | 882 | | 176 | 625 × 512 | 6 × 2 |

Once the spectrogram has been generated, we split it into several smaller patches with fixed width and shift length. Finally, we resize every patches into 143 × 143. Then these patches are used as the training/test samples for the DCNN model.

2.2. CQT Spectrogram

The CQT spectrogram is generated on the CQT features which are computed from the raw audio frames by using the python library Librosa 0.5.0. When invoking the cqt function in the library, the sampling rate is set as 44100 and the other parameters are set as default, namely the number of bins per octave is 12 and the hop length is 512, etc. For each audio file, we generate two CQT spectrograms (size 832 × 143), each for a channel. Once again, we split the spectrogram into patches and feed them into a DCNN model as training/test samples. The patch width is 143 pixels and the shift step is 80 pixels. For each CQT spectrogram, 10 patches can be generated. As a result, we can generate 20 segments from a single audio file.

As mentioned in [6], as for CQT, its frequency resolution is better for low and mid-to-low frequencies. Hence, we generate two versions of CQT spectrograms: one uses all the 84 bands; the other uses 80 bands (the 4 bands related to high frequencies are discarded). For convenience of distinguish, they are mentioned as CQT₈₄ and CQT₈₀ respectively in the rest of this paper.

3. METHODOLOGY

3.1. Deep Convolutional Neural Network

Inspired by [5], we have adopted a DCNN model similar to the one proposed in [5]. The model follows a VGG style network for object recognition. As shown in Table 2, the input size of our model is 143 × 143. We have removed the global average pooling layer from the model, compared to the DCNN model in [5]. The removal of the global average pooling improves the performance in our experiments. The outputs of the next-to-last layer in Table 2 have fifteen feature maps of size 7 × 7. We flatten them into a 735-dimensions vector and feed it to the SoftMax layer.

The optimization setting is as follows. The batch size is 96. The initial learning rate is set as 0.1 and is decreased with 0.998 times every 10 epoches. We also use the L2-Regularization with a weight decay of 0.0001.

Table 2: DCNN model

| Input 1 × 143 × 143 |
|---|
| 5 × 5 Conv(pad-2, stride-2)-32-BN-ReLu |
| 3 × 3 Conv(pad-1, stride-1)-32-BN-ReLu |
| 2 × 2 MaxPooling + DropOut(0.3) |
| 3 × 3 Conv(pad-1, stride-1)-64-BN-ReLu |
| 3 × 3 Conv(pad-1, stride-1)-64-BN-ReLu |
| 2 × 2 MaxPooling + DropOut(0.3) |
| 3 × 3 Conv(pad-1, stride-1)-128-BN-ReLu |
| 2 × 2 MaxPooling + DropOut(0.3) |
| 3 × 3 Conv(pad-0, stride-1)-512-BN-ReLu |
| DropOut(0.5) |
| 1 × 1 Conv(pad-0, stride-1)-512-BN-ReLu |
| DropOut(0.5) |
| 1 × 1 Conv(pad-0, stride-1)-15-BN-ReLu |
| Flatten |
| 15-way SoftMax |

3.2. Fusing Methods

The DCNN can be used to classify acoustic scenes directly on an image sample. However, multiple samples have been generated from an audio file. To make good use of these samples, we further consider the fusing algorithms here.

3.2.1. Voting

Voting is a straightforward method in this situation. Each sample produces one vote and the class which wins the most votes is considered as the final result. For example, when standard spectrum is used and the revolution is R₅₂₉ (as shown in Table 1), there are 24 samples for an audio file. They vote to decide the “correct” class.

By using voting, feature fusion can be easily implemented as well. If it is decided to use standard spectrograms (R₅₂₉, for

instance) and CQT spectrograms together for classification of the scenes, 44 votes are responsible for the result.

3.2.2. SVM

Instead of using the result given by DCNN directly, we can also use the next-to-last layer in the DCNN model to extract features for each sample. By concatenating all features of the samples from the same audio file sequentially, we can obtain a very long feature. Considering the risk of overfitting, a PCA dimensionality reduction operation is applied to the long features. As a result, a new feature has been generated which is encoded by all the samples. This new feature can be referred to as the aggregated feature.

By using the method described above, one aggregated feature can be generated for an audio file, according to a specific preparation of spectrograms. In other words, we can produce one CQT aggregated feature for an audio file, as well as another R_{529} aggregated feature, and so on (R_{706} etc.). When feature fusion is required, these aggregated features can be concatenated again into another feature. Note that PCA is not performed this time.

Finally, a SVM model is used to tell the final result by using these features as training/test samples. The linear kernel is applied in our experiments.

In our fusion experiments, SVM generally works better than voting mechanism. The reason for this is that the concatenating inputs of SVM provide sequential information which makes it possible for SVM to extract more comprehensive features for understanding the auditory scenes.

4. EXPERIMENTS AND RESULTS

In this section, we will demonstrate the experiments using the data and methods mentioned above. The experiments use the TUT Acoustic Scenes 2017 dataset (the part of acoustic scene classification). The results are conducted on the 4-fold cross validation set exactly the same to the baseline system in [8].

4.1. Classifying with Standard Spectrograms

Firstly, we try to explore the classification performances of standard spectrogram with different resolutions. The setting parameters about the resolutions involved here can refer to Table 1. For each resolution (R_{529} , for example), we will provide 3 types of accuracies: R_{529} (DCNN) is the accuracy computed by the DCNN model (see Table 2 in Section 3.1) on the patch sample as a unit; R_{529} (Voting) uses the voting algorithm to ensemble the baseline results from DCNN model; and R_{529} (SVM) uses SVM method instead. The accuracy results are shown as follows.

Table 3: Accuracies of standard spectrograms based solutions

| | Folder 1 | Folder 2 | Folder 3 | Folder 4 | Average |
|-------------------------------------|----------|----------|----------|----------|---------------|
| R_{529}(DCNN) | 0.7749 | 0.7779 | 0.6948 | 0.7557 | 0.7509 |
| R_{529}(Voting) | 0.8598 | 0.8789 | 0.7656 | 0.8632 | 0.8419 |
| R_{529}(SVM) | 0.8615 | 0.8721 | 0.7732 | 0.8684 | 0.8438 |
| R_{706}(DCNN) | 0.775 | 0.7892 | 0.7065 | 0.752 | 0.7557 |
| R_{706}(Voting) | 0.8496 | 0.873 | 0.7928 | 0.85 | 0.8451 |
| R_{706}(SVM) | 0.853 | 0.8679 | 0.8227 | 0.8709 | 0.8536 |

| | | | | | |
|-------------------------------------|--------|--------|--------|--------|--------|
| R_{882}(DCNN) | 0.772 | 0.7836 | 0.6532 | 0.7489 | 0.7394 |
| R_{882}(Voting) | 0.8513 | 0.8508 | 0.7573 | 0.8602 | 0.8299 |
| R_{882}(SVM) | 0.8581 | 0.8687 | 0.7622 | 0.8635 | 0.8381 |

Looking at Table 3, we find that both voting and SVM can significantly improve the baseline accuracies of DCNN model for all the resolutions. Specifically, the SVM method is slightly better than voting algorithm. In this group of experiments, the best average accuracy is 0.8536 which is achieved by the R_{706} (SVM) solution.

4.2. Classifying with CQT Spectrograms

Using the same DCNN model, we conduct several CQT spectrogram based DCNN experiments. The accuracy results are presented in Table 4.

Table 4: Accuracies of CQT spectrograms based solutions

| | Folder 1 | Folder 2 | Folder 3 | Folder 4 | Average |
|---------------------------------------|----------|----------|----------|----------|---------------|
| CQT_{84} (DCNN) | 0.7278 | 0.6946 | 0.6958 | 0.7067 | 0.7062 |
| CQT_{84} (Voting) | 0.8154 | 0.7928 | 0.7937 | 0.8188 | 0.8052 |
| CQT_{84} (SVM) | 0.8231 | 0.7715 | 0.8005 | 0.8188 | 0.8035 |
| CQT_{80} (DCNN) | 0.6972 | 0.6878 | 0.6885 | 0.6896 | 0.6908 |
| CQT_{80} (Voting) | 0.7701 | 0.7715 | 0.7809 | 0.7872 | 0.7774 |
| CQT_{80} (SVM) | 0.7846 | 0.7519 | 0.7801 | 0.7889 | 0.7764 |

Generally, the accuracies of CQT spectrogram based solutions are unsatisfactory, compared with the standard spectrogram. Furthermore, the accuracies of CQT_{80} are worse than the ones of CQT_{84} , which is different with our original expectation [6].

4.3. Classifying with Standard and CQT Spectrograms

Although the accuracies of CQT spectrogram are not very competitive, significant improvements can be achieved when fused with standard spectrograms in our experiments. We have tried several feature combinations and have presented their results in Table 5.

Table 5: Accuracies of multiple spectrograms fusion solutions

| | Folder 1 | Folder 2 | Folder 3 | Folder 4 | Average |
|---|----------|----------|----------|----------|---------------|
| $R_{529} + CQT_{84}$ (Voting) | 0.8769 | 0.9088 | 0.8406 | 0.8889 | 0.8788 |
| $R_{529} + CQT_{84}$ (SVM) | 0.8684 | 0.919 | 0.8764 | 0.9162 | 0.895 |
| $R_{529} + CQT_{80}$ (Voting) | 0.8752 | 0.902 | 0.8465 | 0.8949 | 0.8796 |
| $R_{529} + CQT_{80}$ (SVM) | 0.8641 | 0.9173 | 0.8764 | 0.9265 | 0.896 |
| $R_{706} + CQT_{84}$ (Voting) | 0.8547 | 0.8917 | 0.861 | 0.8983 | 0.8764 |
| $R_{706} + CQT_{84}$ (SVM) | 0.865 | 0.9037 | 0.896 | 0.9299 | 0.8986 |
| $R_{706} + CQT_{80}$ (Voting) | 0.8504 | 0.8832 | 0.8576 | 0.9043 | 0.8739 |
| $R_{706} + CQT_{80}$ (SVM) | 0.8556 | 0.902 | 0.89 | 0.9282 | 0.8939 |

As we can see, the four fusion solutions using SVM method have achieved satisfactory results. All of the four accuracies are greater than 0.89. Actually, the highest one is 0.8986 and the lowest one is 0.8939. It is easy to find that the differences of accuracies among these four are very slight. However, compared to the best results of standard spectrogram and CQT spectrogram solutions (0.8536 and 0.8052 respectively), the improvements in accuracies of these fusion solutions are still significant, which proves the effectiveness of our multiple spectrograms fusion. Similarly, Table 5 shows the accuracy superiority of SVM method over the voting in the fusion scenarios. To better understand the fusion performance, the class-wise accuracies of the best result, namely $R_{706} + \text{CQT}_{84}$ (SVM), are further given in Table 6.

Table 6: Class-wise accuracies of the best fusion solution

| | Folder 1 | Folder 2 | Folder 3 | Folder 4 | Average | Baseline |
|------------------------------|-----------------|-----------------|-----------------|-----------------|----------------|-----------------|
| beach | 0.8718 | 0.7564 | 1.0 | 0.7949 | 0.8558 | 0.753 |
| bus | 0.9872 | 0.9615 | 0.8462 | 0.9615 | 0.9391 | 0.718 |
| cafe/rest- aurant | 0.3333 | 0.7051 | 0.7564 | 0.8462 | 0.6603 | 0.577 |
| car | 0.9744 | 0.9615 | 0.9744 | 1.0 | 0.9776 | 0.971 |
| city _center | 0.8718 | 0.8333 | 0.9231 | 0.9103 | 0.8846 | 0.907 |
| forest _path | 0.9615 | 1.0 | 0.9615 | 1.0 | 0.9808 | 0.795 |
| grocery _store | 1.0 | 1.0 | 0.8718 | 0.9359 | 0.9519 | 0.587 |
| home | 0.9744 | 0.8889 | 0.9753 | 0.8077 | 0.9116 | 0.686 |
| library | 0.6282 | 1.0 | 0.9359 | 0.9487 | 0.8782 | 0.571 |
| metro _station | 1.0 | 1.0 | 0.9872 | 1.0 | 0.9968 | 0.917 |
| office | 0.9872 | 1.0 | 0.9359 | 1.0 | 0.9808 | 0.997 |
| park | 0.6923 | 0.8462 | 0.6923 | 0.8974 | 0.7821 | 0.702 |
| residential _area | 0.8846 | 0.9231 | 0.8718 | 0.8718 | 0.8878 | 0.641 |
| train | 0.8077 | 0.9487 | 0.7179 | 0.9744 | 0.8622 | 0.580 |
| tram | 1.0 | 0.7308 | 0.9872 | 1.0 | 0.9295 | 0.817 |
| total | 0.865 | 0.9037 | 0.896 | 0.9299 | 0.8986 | 0.748 |

The last column in Table 6 presents the performance of the baseline system provided along with the TUT Acoustic Scenes 2017 dataset in [8]. As we can see, the average accuracy of our best fusion system outperforms the one of baseline system by 20.13 percent.

5. SUBMISSION RESULTS

All the development data are utilized to train the model, and the submitted results are tested on this final model. According to the fusion methods, two systems are included in our submission to the DCASE2017 challenge (task 1). The first one is DCNN based voting system, which fuses the standard (R_{706}) and CQT_{84} spectrograms by voting method (namely the $R_{706} + \text{CQT}_{84}$ (Voting) solution). The second one is DCNN based SVM system, which fuses the same data by SVM method (namely the $R_{706} + \text{CQT}_{84}$ (SVM) solution).

6. CONCLUSION

In the ASC research domain, CNN is becoming more and more popular[1][2][5][6]. In this work, a DCNN solution is proposed for the acoustic scene classification. The main contributions of this work lie in two aspects as follows. First, a deep CNN model is presented, which is originated from [5] and is improved to be more suitable for the problem. Second, a multi-spectrogram fusion method is proposed. Multiple spectrograms are fed into the same DCNN model and the corresponding features are fused to improve the accuracy of classification. In this work, the standard spectrogram and the CQT spectrogram are studied. The best accuracy of using the standard spectrograms is 0.8536; and the one of using CQT spectrograms is 0.8052. Although the accuracy of using CQT spectrograms is unsatisfactory, it can significantly improve the accuracy when fused with the standard spectrogram. The best result of the fusion scheme is 0.8986 and outperforms the best results of the single spectrogram schemes by more than 0.045. We believe the performance can be further improved by using some other skills, such as fine tuning of parameters, normalization of spectrograms in the training of DCNN, utilizing the temporal characteristics, etc.

In our experiments, the fusion of multi-resolution standard spectrograms is also explored. The accuracy is also improved slightly, compared to the single resolution schemes. In summary, using the multiple spectrograms can greatly augment the size of training samples, which will result in a better DCNN performance.

When generating standard spectrograms, the width of sliding window as well as the overlap amount are important parameters. In our opinion, they both impact the accuracies of classification. Owing to the time limit, we have not performed grid searching for their values. In our future work, we will further explore the correlations between these parameters and the accuracies. It would be beneficial for finding out the best resolution for the DCNN model.

In [6], it is recommended to remove high frequency bins when preparing CQT inputs for the proposed CNN architecture. However, in our experiment, CQT_{84} works better than CQT_{80} in all cases, which differs with the results in [6]. In fact, the generation of CQT feature in our method is slightly different with the one proposed by [6], for example, we produce CQT samples for left and right channels separately. However, we don't think this contributes much to the difference of the conclusions. Actually, the main difference lies in the architectures of the two CNN model. The CNN structure in [6] is much simpler than the one in this paper. We suppose that the DCNN model in this paper can more effectively utilize the high frequencies bins. This should be validated in our future work.

7. REFERENCES

- [1] Bae S H, Choi I, Kim N S. Acoustic Scene Classification using Parallel Combination of LSTM and CNN[J]. IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2016
- [2] Valenti M, Diment A, Parascandolo G, et al. DCASE 2016 Acoustic Scene Classification using Convolutional Neural

- Networks[C]//Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2016), Budapest, Hungary. 2016.
- [3] Dai Wei, Juncheng Li, Phuong Pham, et al. Acoustic Scene Recognition with Deep Neural Networks (DCASE challenge 2016)[R]. Robert Bosch Research and Technology Center, 3 September 2016.
 - [4] Rohit Patiyal, Padmanabhan Rajan. Acoustic Scene Classification using Deep Learning[J]. IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), 2016
 - [5] 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[C]//Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2016), Budapest, Hungary. 2016.
 - [6] T. Lidy and A. Schindler. CQT-based Convolutional Neural Networks for Audio Scene Classification[C]//Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2016), September 2016, pp. 60-64.
 - [7] A Gorin, N Makhazhanov, N Shmyrev. DCASE 2016 Sound Event Detection System Based on Convolutional Neural Network[C]//Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2016), Budapest, Hungary. 2016.
 - [8] Acoustic Scene Classification[R/OL] <http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification>

ROBUST SOUND EVENT DETECTION THROUGH NOISE ESTIMATION AND SOURCE SEPARATION USING NMF

Qing Zhou, Zuren Feng

Xi'an Jiaotong University

*School of Electronic and Information Engineering
No. 28 Xianning West Road, Xi'an, Shaanxi 710049, P. R. China
belief2012@stu.xjtu.edu.cn, fzs9910@mail.xjtu.edu.cn*

ABSTRACT

This paper addresses the problem of sound event detection under non-stationary noises and various real-world acoustic scenes. An effective noise reduction strategy is proposed in this paper which can automatically adapt to background variations. The proposed method is based on supervised non-negative matrix factorization (NMF) for separating target events from noise. The event dictionary is trained offline using the training data of the target event class while the noise dictionary is learned online from the input signal by sparse and low-rank decomposition. Incorporating the estimated noise bases, this method can produce accurate source separation results by reducing noise residue and signal distortion of the reconstructed event spectrogram. Experimental results on DCASE 2017 task 2 dataset show that the proposed method outperforms the baseline system based on multi-layer perceptron classifiers and also another NMF-based method which employs a semi-supervised strategy for noise reduction.

Index Terms— Sound event detection, non-negative matrix factorization, sparse and low-rank decomposition, source separation

1. INTRODUCTION

Sound events such as gunshots, screams, glass breaks, etc. are often associated with hazardous situations. Automatic detection and monitoring of these sound events can be very useful for security reasons. A key problem in sound event detection is the presence of the highly non-stationary and time-varying background noise in realistic applications [1]. Most existing methods using a well-trained classifier on environment-specific training data is designed for particular situations and is thus unable to handle unseen noise. In addition, even if it is possible to train a classifier with an enormous amount of data involving different types of sounds in different environments, it enables the flexibility in dealing with different noises but at the expense of sacrificing performance at specific environments [2]. The goal of this paper is to develop robust detection methods which can automatically adapt to background variations for practical applications.

Techniques of non-negative matrix factorization (NMF) [3] have been extensively studied and successfully applied in speech enhancement for separating speech from noise [4-6]. Recently, many sound event detection systems using NMF have been published with promising results [7-11]. NMF models the spectrogram of a sound signal with a dictionary of spectral bases and a

corresponding activation matrix. Since the activations may vary along time, this model can describe non-stationary signals to some extent. The key strategy for NMF to detect target events from noise is to express noise and target events by different sets of bases. The input noisy signal is first decomposed by this combined dictionary and then the target events can be reconstructed by only using the event components.

In sound event detection task, samples of the target event class are usually available and an event dictionary can be pre-trained and kept fixed during test. Strategies differ when dealing with noise. If a noise dictionary is also pre-trained and used in the decomposition, it is the supervised case. In contrast, in the semi-supervised case the noise dictionary is unknown and needs to be updated concurrently during test. For example, Gemmeke et al. [7] extracted bases for both the target event and the background noise and kept the dictionaries fixed during test. But this method can only be applied in simple and fixed noise conditions. To better handle unseen noise, Komatsu et al. [8] adopted the semi-supervised NMF strategy. A noise dictionary was introduced and learned during test with the aim of modeling unknown spectra which were not included in the training data. Their method can adapt to different noises but is not suitable for handling non-stationary noises. Since it lacks control over the noise bases, this method may not obtain accurate separation results, especially when there are many interference sound signals in the background which exhibit similar spectral profiles as the target event.

In order to enhance system robustness and reduce background interference, this paper proposes to first estimate a noise dictionary from the input test signal and then conducts the supervised source separation procedure. Noise dictionary learning is accomplished by the technique of sparse and low-rank decomposition or robust NMF [4, 12-14] which has been proved useful in foreground/background separation. The underlying idea is to decompose a matrix into the summation of a low-rank matrix and a sparse matrix. This model also applies to sound event detection, that is, the foreground events are sparse due to its rare occurrence while the background noise is usually more stable and also less spectrally diverse than the foreground events and thus can be modeled by the low-rank part [13]. The low-rank part is further expressed by a linear combination of a limited number of bases (referring to the noise dictionary) upon NMF. Guided by the additional knowledge of noise bases, the event spectrogram can be better reconstructed via supervised NMF with less noise residue and signal distortion. The proposed method shows its effectiveness of adaptive noise reduction and achieves better performance compared to the semi-supervised method especially when dealing with similar background interference.

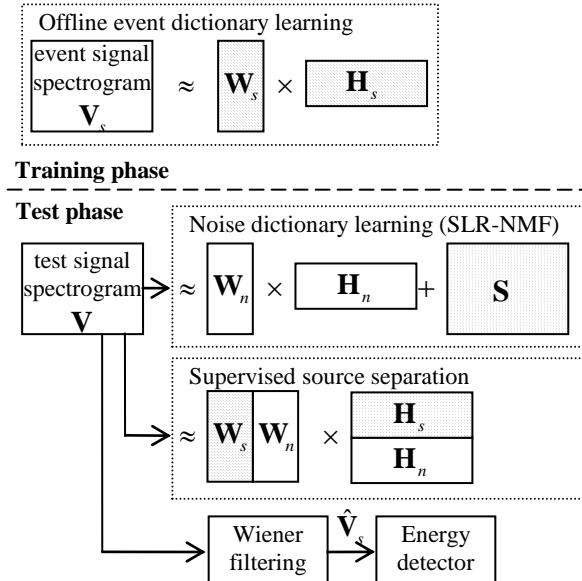


Figure 1: Framework of the proposed method

2. PROPOSED METHOD

The framework of the proposed method is presented in Fig. 1. Audio signals are all resampled to 44100 Hz and transformed via the STFT with a frame length of 40 ms and 50% overlap. Magnitude spectrograms are used as audio features and the dimension of the feature vector is 1025.

In the training phase, an event dictionary is trained by unsupervised NMF using the training set of clean event samples. The test phase mainly consists of three steps: noise dictionary learning, supervised source separation, and event detection. First, a noise dictionary is estimated from the input test signal by unsupervised sparse and low-rank non-negative matrix factorization (called SLR-NMF for short in this paper). Then, combining the estimated noise dictionary with the event dictionary, the input signal is decomposed again by NMF for supervised source separation. Thus the event spectrogram can be reconstructed by only using the event components. Finally, the estimated event spectrogram is further smoothed and then processed by an energy detector to generate the final onset/offset results.

2.1. Offline event dictionary learning

For each event class, an event dictionary is trained on all concatenated event spectrograms of the training data. Sparse NMF described in [15] is implemented using Kullback-Leibler (KL) divergence [16] as the distance measure and a sparsity parameter of 0.2. This produces an event dictionary denoted by

$\mathbf{W}_s \in \mathbb{R}_+^{N \times R_s}$, where N is the number of frequency bins, R_s is the number of event bases, and the subscript s refers to “event signal”.

2.2. Noise dictionary learning by SLR-NMF

Sparse and low-rank decomposition represents a matrix as the summation of a low-rank matrix and a sparse matrix. The inter-

pretation with respect to the low-rank part and the sparse part differs according to specific applications. For sound event detection, the foreground events rarely happen and occupy very limited entries of the input matrix and thus can be well expressed by the sparse part. In contrast, the background noise is usually more stable and also less spectrally diverse than the foreground events and thus can be modeled by the low-rank part.

Let $\mathbf{V} \in \mathbb{R}_+^{N \times T}$ denote the spectrogram of the input noisy signal with T time frames. The model of SLR-NMF is given by

$$\mathbf{V} \approx \mathbf{W}_n \mathbf{H}_n + \mathbf{S} \quad (1)$$

in which $\mathbf{S} \in \mathbb{R}_+^{N \times T}$ is the sparse part representing the foreground events, and the low-rank part dedicated to the background noise is further represented upon NMF as the product of a noise dictionary $\mathbf{W}_n \in \mathbb{R}_+^{N \times R_n}$ and an activation matrix $\mathbf{H}_n \in \mathbb{R}_+^{R_n \times T}$. R_n is the number of noise bases satisfying $R_n < \min\{N, T\}$ and the subscript n refers to “noise”.

The following optimization problem is constructed to solve the decomposition in (1):

$$\min_{\mathbf{W}_n, \mathbf{H}_n, \mathbf{S}} D(\mathbf{V} \| \mathbf{W}_n \mathbf{H}_n + \mathbf{S}) + \lambda \|\mathbf{S}\|_1 \quad (2)$$

in which the first term is the KL divergence between the input matrix and its approximation, and the second term is a sparsity constraint on \mathbf{S} which is measured by its L_1 -norm. λ controls the weight of the sparsity constraint in the cost function and its selection will be discussed later in Section 3.

The multiplicative update rules for (2) are given as follows:

$$\mathbf{W}_n \leftarrow \mathbf{W}_n \odot \left(\frac{\mathbf{V}}{\mathbf{W}_n \mathbf{H}_n + \mathbf{S}} \mathbf{H}_n^T \right) / (\mathbf{1} \mathbf{H}_n^T) \quad (3)$$

$$\mathbf{H}_n \leftarrow \mathbf{H}_n \odot \left(\mathbf{W}_n^T \frac{\mathbf{V}}{\mathbf{W}_n \mathbf{H}_n + \mathbf{S}} \right) / (\mathbf{W}_n^T \mathbf{1}) \quad (4)$$

$$\mathbf{S} \leftarrow \mathbf{S} \odot \left(\frac{\mathbf{V}}{\mathbf{W}_n \mathbf{H}_n + \mathbf{S}} \right) / (\mathbf{1} + \lambda) \quad (5)$$

where $\mathbf{1}$ is an all-1 matrix with the same dimension as \mathbf{V} and the superscript T means the transposition of a matrix. $\mathbf{A} \odot \mathbf{B}$ and \mathbf{A}/\mathbf{B} refer to the element-wise multiplication and division, respectively. Hence, by solving (2) we obtain an estimate of the noise dictionary which directly describes the surrounding background of the current input signal.

2.3. Supervised source separation

Incorporating the noise dictionary \mathbf{W}_n learned in Section 2.2 and the pre-trained event dictionary \mathbf{W}_s , the input test signal can be decomposed into the noise part and the event part by NMF in a supervised way as follows:

$$\mathbf{V} \approx \mathbf{W}_s \mathbf{H}_s + \mathbf{W}_n \mathbf{H}_n \quad (6)$$

in which $\mathbf{W}_s \mathbf{H}_s$ and $\mathbf{W}_n \mathbf{H}_n$ represent the estimated event and noise spectrograms, respectively. Here we conduct a second separation to the input noisy signal in a way that is different from the totally unsupervised decomposition in Section 2.2. This step can obtain more reliable separation results since both the prior knowledge of the noise and the target event class to be detected is utilized.

Source separation problem in (6) is solved by minimizing the KL divergence between the input matrix and its reconstruction. The corresponding optimization problem is expressed as

$$\min_{\mathbf{H}_s, \mathbf{H}_n} D(\mathbf{V} \mid \mathbf{W}_s \mathbf{H}_s + \mathbf{W}_n \mathbf{H}_n) \quad (7)$$

Update rules for the activation matrices \mathbf{H}_s and \mathbf{H}_n in (7) are given by

$$\mathbf{H}_s \leftarrow \mathbf{H}_s \odot \left(\mathbf{W}_s^T \frac{\mathbf{V}}{\mathbf{W}_s \mathbf{H}_s + \mathbf{W}_n \mathbf{H}_n} \right) / (\mathbf{W}_s^T \mathbf{1}) \quad (8)$$

$$\mathbf{H}_n \leftarrow \mathbf{H}_n \odot \left(\mathbf{W}_n^T \frac{\mathbf{V}}{\mathbf{W}_s \mathbf{H}_s + \mathbf{W}_n \mathbf{H}_n} \right) / (\mathbf{W}_n^T \mathbf{1}) \quad (9)$$

2.4. Event detection

Like what is done in speech processing [4-5], Wiener filtering is conducted on the input spectrogram to get the final estimate of the event spectrogram, that is,

$$\hat{\mathbf{V}}_s = \mathbf{V} \odot \frac{\mathbf{W}_s \mathbf{H}_s}{\mathbf{W}_s \mathbf{H}_s + \mathbf{W}_n \mathbf{H}_n} \quad (10)$$

For event detection, an energy detector is applied to $\hat{\mathbf{V}}_s$ by measuring the accumulation of energies of all frequency bins per frame. High energy values exceeding a threshold in a number of successive frames indicate the presence of a target event.

3. EXPERIMENTS

The proposed method is evaluated on DCASE 2017 task 2 dataset. This task focuses on the detection of three types of rare sound events (baby cry, glass break, and gunshot) in artificially created mixtures. The background audio material is from 15 different audio scenes including home, park, metro station, etc., making it a very complex detection scenario. In our detection algorithm, only isolated clean sound examples for target event classes are used for training and an event dictionary for each target event class is trained separately. It should be pointed out that no background audio is used for training. The development test set which contains 500 mixture audio examples for each target class is used for evaluation. Different SNR levels are tested including 6 dB, 0 dB, and -6 dB.

Metrics used in the challenge are event-based error rate (ER) and event-based F-score (see [17] for details). An event is considered correctly detected using onset-only condition with a collar of 500 ms.

3.1. Parameter settings

Major parameters in the proposed algorithm are R_s , R_n , and λ . We use the training mixtures in the development dataset for tuning the parameters, which is disjoint from the test set. The search range for each parameter is empirically determined, that is, $R_s, R_n \in \{16, 32, 48, 64\}$, $\lambda \in \{0.05, 0.1, 0.2, 0.5, 0.8, 1\}$.

The numbers of event bases and noise bases are very important parameters. For qualitative analysis, using a sufficient number of bases is preferable to model audio sources precisely. However, using too many bases may degrade the performance since it would easily lead to the mixing problem, that is, noise

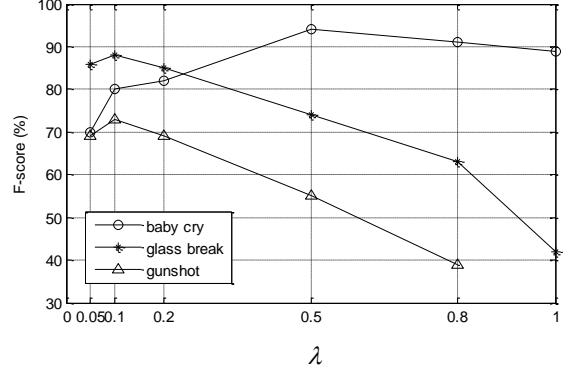


Figure 2: F-score results on the training mixtures with different values of the sparsity parameter λ . Best results are achieved at 0.1 for glass breaks and gunshots, 0.5 for the baby cry class.

may be wrongly described by the event bases or in the reverse way. After a grid search over the pre-defined range, we found that $R_s=32$ and $R_n=32$ are good choices which guarantee excellent performance and also a satisfactory computational load.

The sparsity parameter λ used in the noise dictionary learning step has a significant effect on performance. It controls the strength of the sparsity constraint on the foreground event part and thus determines a trade-off between noise reduction and signal distortion. A larger λ means a sparser foreground estimate and a more sufficient noise estimate but at the expense of including some foreground event components. Since the goal of this decomposition is to learn a noise dictionary, it is better not to retain too many foreground event components within the noise part. So λ should be a relatively small value. According to the F-score results in Fig. 2, best results for glass breaks and gunshots are achieved around $\lambda=0.1$. The performance degrades a bit under 0.1 and a larger λ also yields poor results. However, the case for the baby cry class differs and it turns out that a relatively large λ produces better results. This may be attributed to the considerable difference between the baby cry spectrum and the noise spectrum which enables a tolerance of the event residue within the noise part. Hence we set $\lambda=0.5$ for the baby cry class and $\lambda=0.1$ for glass breaks and gunshots in experiments.

For NMF implementation in our method, the matrices if needed are randomly initialized, and we run the algorithm several times to find the one which produces the lowest cost. The number of iterations is set to be 200 which is sufficient for convergence in our experiments. For post-processing, the energy sequence computed from the estimated event spectrogram is further smoothed by a moving average filter. The duration of the filter is set to be a little shorter than the minimum length of the target event class. Very short detected events are also removed. Additionally, in order to obtain more accurate onset/offset results, a double-thresholding strategy is adopted. In other words, events are first detected using a large threshold, and then a small threshold is used to search within a small range before and after the detected event for the final onset and offset. This strategy is necessary and found to be useful especially for the baby cry class. Because there might be regions of short pauses within a baby cry event, the detector using one threshold may only locate the part with higher energies thus resulting in inaccurate onset or offset.

Table 1: Class-wise results on the development test set

| Event class | Proposed | | Semi-supervised | | Baseline | |
|-------------|----------|------|-----------------|------|----------|------|
| | ER | F(%) | ER | F(%) | ER | F(%) |
| Baby cry | 0.20 | 89.9 | 0.27 | 83.5 | 0.67 | 72.0 |
| Glass break | 0.22 | 89.2 | 0.31 | 80.1 | 0.22 | 88.5 |
| Gunshot | 0.42 | 78.4 | 0.57 | 65.8 | 0.69 | 57.4 |
| Average | 0.28 | 85.8 | 0.39 | 76.5 | 0.53 | 72.7 |

Table 2: Results on the final evaluation set (taken from DCASE2017 Challenge website)

| Method | Proposed | | Baseline | |
|---------|----------|------|----------|------|
| | ER | F(%) | ER | F(%) |
| Average | 0.31 | 84.2 | 0.63 | 64.1 |

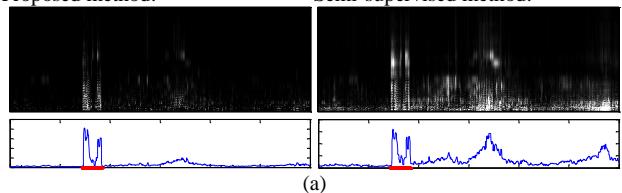
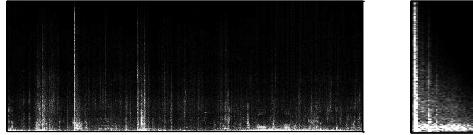
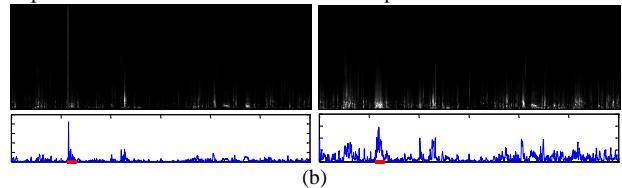
3.2. Results and discussions

The detection results for each target event class on the development test set are presented in Table 1. Compared with the baseline results provided by the challenge [18-19], our method achieves better performance. The average ER and F-score of the proposed method are 0.28 and 85.8% compared to 0.53 and 72.7% of the baseline. The baseline system employs a multi-layer perceptron classifier for each target class and needs to be trained on mixture audio examples of different scenes and different SNR levels. In contrast, our method only uses isolated event examples for training and adopts an adaptive noise reduction strategy, which is an advantage over the baseline system. The results on the final evaluation set are also presented in Table 2 which are taken from the DCASE2017 Challenge website [19].

We also compared the proposed method with the semi-supervised NMF method which was applied in several literatures [8, 20, 21]. In this approach, the noise dictionary in (7) is unknown and needs to be updated along with other matrices in the source separation step. Parameters like the numbers of event and noise bases for this method are set to those that produce the best performance. This method to some extent has the adaptability to describe the time-varying characteristics of noise spectra. However, since it lacks control over the noise bases, it may wrongly decompose noise into the event part or conversely. So it can easily lead to noise residue or signal distortion within the estimated event spectrogram. The situation gets worse especially when encountering analogous background interference. The proposed method estimates a noise dictionary from the current input signal which directly models the surrounding background and thus can obtain accurate separation results in the supervised NMF step.

It can be observed from Table 1 that our method outperforms the semi-supervised version and the detection results are improved for all three event classes. More detailed comparison of the two methods is illustrated by two test examples as shown in Fig. 3. In the baby cry detection example, there existed strong noise interference in the test signal. Our method had a fairly good effect of noise reduction and obtained excellent detection results. But for the semi-supervised method, the baby cry event was well captured by the event bases but part of the noise were also describe by the event bases, which led to harmful confusion in detection. In the second example of gunshot detection, the background noise in the test signal was highly non-stationary and

Spectrogram of test signal: baby cry 7.2s-9.2s 0dB Baby cry event

Estimated event part and its energy curve
Proposed method: Semi-supervised method:(a)
Spectrogram of test signal: gunshot 5.7s-6.4s 0dB Gunshot eventEstimated event part and its energy curve
Proposed method: Semi-supervised method:(b)
Figure 3: Detection examples of (a) a baby cry event and (b) a gunshot event

contained sounds whose spectral profile was very similar to that of a gunshot event. Our method demonstrated its ability of tackling similar background interference and correctly located the gunshot event. However, it can be seen from the estimated event part of the semi-supervised method that many important event components were lost.

4. CONCLUSIONS

This paper presents a sound event detection method based on supervised source separation by NMF. In order to deal with non-stationary noises and background variations, this paper proposes to estimate a noise dictionary online from the input test signal using the technique of sparse and low-rank decomposition. Because the estimated noise dictionary can exactly describe the surrounding background, the succeeding supervised source separation via NMF can provide accurate separation of target events and noise. Experimental results demonstrate the noise reduction ability of the proposed method when dealing with non-stationary noises and similar background interference. The proposed method achieves better results than the baseline system based on MLP and also outperforms another NMF-based method which employs a semi-supervised strategy for noise reduction. Note that the proposed work needs a relatively long signal to learn a noise dictionary. It is not suitable for real-time applications. Future work will be dedicated to develop real-time noise dictionary learning techniques.

5. REFERENCES

- [1] M. Crocco, M. Cristani, A. Trucco, V. Murino, "Audio surveillance: a systematic review," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 52:1–52:46, 2016.
- [2] A. Rabaoui, Z. Lachiri, and N. Ellouze, "Using HMM-based classifier adapted to background noises with improved sounds features for audio surveillance application," *Int. J. Signal Process.*, vol. 5, no. 1, pp. 46–55, 2009.
- [3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.
- [4] M. Sun, Y. Li, J. F. Gemmeke, and X. W. Zhang, "Speech enhancement under low SNR conditions via noise estimation using sparse and low-rank NMF with K-L divergence," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 23, no. 7, pp. 1233–1242, Jul. 2015.
- [5] M. N. Schmidt, J. Larsen, and K. Lyngby, "Wind noise reduction using non-negative sparse coding," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2007, pp. 431–436.
- [6] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Trans. Audio Speech Lang. Process.*, vol. 21, no. 10, pp. 2140–2151, 2013.
- [7] J. Gemmeke, L. Vuggen, P. Karsmakers, B. Vanrumste, and H. Van hamme, "An exemplar-based NMF approach to audiodetection," in *Proc. WASPAA*, 2013, pp. 1–4.
- [8] T. Komatsu, T. Toizumi, R. Kondo, and S. Yuzo, "Acoustic event detection method using semi-supervised non-negative matrix factorization with a mixture of local dictionary," in *Proc. DCASE2016 Workshop*, 2016, pp. 45–49.
- [9] C. V. Cotton and D. P. W. Ellis, "Spectral vs. spectro-temporal features for acoustic event detection," in *Proc. WASPAA*, 2011, pp. 69–72.
- [10] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, "Sound event detection in multisource environments using source separation," in *Proc. CHiME*, 2011, pp. 36–40.
- [11] T. Heittola, A. Mesaros, T. Virtanen, and G. Moncef, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Proc. ICASSP*, 2013, pp. 8677–8681.
- [12] P. S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, "Singing-voice separation from monaural recordings using robust principal component analysis," in *Proc. ICASSP*, 2012, pp. 57–60.
- [13] L. Zhang, Z. Chen, M. Zheng, and X. He, "Robust non-negative matrix factorization," *Front. Electr. Electron. Eng. China*, vol. 6, no. 2, pp. 192–200, 2011.
- [14] Z. Chen and D. P. W. Ellis, "Speech enhancement by sparse, low-rank, and dictionary spectrogram decomposition," in *Proc. WASPAA*, 2013, pp. 1–4.
- [15] J. Eggert and E. Korner, "Sparse coding and NMF", in *Proc. IEEE Int. Conf. Neural Networks*, 2004, vol. 4, pp. 2529–2533.
- [16] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *NIPS*, 2001, pp. 556–562.
- [17] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Appl. Sci.*, vol. 6, no. 6, pp. 162, 2016.
- [18] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: tasks, datasets and baseline system," in *Proc. DCASE2017 Workshop*, 2017, submitted.
- [19] DCASE2017 Challenge, "IEEE ASSP Challenge of Detection and Classification of Acoustic Scenes and Events," <http://www.cs.tut.fi/sgn/arg/dcse2017/challenge/>.
- [20] S. Mohammed and I. Tashev, "A statistical approach to semi-supervised speech enhancement with low-order non-negative matrix factorization," in *Proc. ICASSP*, 2017, pp. 546–550.
- [21] F. Weninger, J. Feliu, and B. Schuller, "Supervised and semi-supervised suppression of background music in monaural speech recordings," in *Proc. ICASSP*, 2012, pp. 61–64.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4042-4