DCASE 2016 SOUND EVENT DETECTION SYSTEM BASED ON CONVOLUTIONAL NEURAL NETWORK

Arseniy Gorin, Nurtas Makhazhanov, Nickolay Shmyrev

ACTechnologies LLC, 8/1 Nauchny pr., 117246, Moscow, Russia
gorinars@ya.ru, makhazhanovn@gmail.com, nshmyrev@gmail.com

ABSTRACT

The report describes a sound event detection system submitted to DCASE 2016 (detection and classification of acoustic scenes and events) challenge. In this work a convolutional neural network is used for detecting and classifying polyphonic events in a long temporal context of filter bank acoustic features. Given a small amount of training resources, data augmentation has been explored. On development data set the system achieves an average 7.7% relative error rate improvement and 55.1% relative F-measure improvement. However, it is still unable to detect short events with limited training data. Out of 17 challenge participants, this system was ranked top 12th in terms of segment error rate (0.98%) and 3rd in terms of F-measure (44.1%) on the evaluation data set.

Index Terms— acoustic event detection, convolutional neural networks, data augmentation

1. INTRODUCTION

The *DCASE 2016 sound event detection in real life audio* task consists in annotating the audio data containing 18 polyphonic acoustic events recorded in several acoustic conditions.

The challenge provided a limited amount of data per each acoustic event and the rules did not allow to use additional resources for training. To cope with this problem, our initial idea was to rely on various transformations of the training data including audio mixing and speed perturbation. Another problem with this task is that training of frame-based classifiers is difficult, because only segment-level annotations were available and many frames contained no relevant information for the tagged class.

The system described in this report is based on a convolutional neural network (CNN) model, which demonstrated promising results in various machine learning tasks, including acoustic event detection [1, 2, 3]. CNNs can benefit from a long context of features, but usually require sufficient amount of data for parameter estimation. The latter problem can be in part handled by artificial data augmentation. The code and the experimental setup described in this report are available on github[1].

2. SYSTEM DESCRIPTION

The system is built on top of python DCASE baseline distributed by the organizers and described in [4]. Feature extraction is done with python librosa[2] package and neural network modeling is based on keras[3] library. The remainder of this section describes the core parts of the system.

2.1. Feature extraction

The baseline MFCC acoustic features are replaced by log Mel filter banks. The original 44 kHz audios are down-sampled to 16 kHz. Then, 60 filter bank features are extracted within 25 ms frames with 15 ms overlap. Even for stacked context of features it was found to be useful to add first and second derivatives computed within a 9 frame window.

2.2. Data augmentation

Challenge rules did not allow to use additional data for training or mixing. At the same time for some classes less than 20 seconds of data were provided. Consequently, an attempt was made to enlarge the data set by applying two types of transformations: speed perturbation (or time stretching) and block mixing [5].

Speed perturbation is done using sox[4] "speed" command, which in turn adjusts both speed and pitch. The train data set of each fold was enlarged using 0.8, 0.9, 1.1 and 1.2 speed perturbation rates. The reference event time makers were adjusted accordingly.

We also tried to increase polyphony by applying block mixing. To do so, 5 second segments, each containing at least one event were randomly mixed with a slight random volume perturbation. No improvement was observed with block mixing in the preliminary experiment, so for the reported experiments and the final submission only speed perturbation was applied.

2.3. Neural network architecture

A single CNN is trained to handle both home and residential area recordings. However, in decoding time only the relevant subset of events is taken into account per each scene. In addition, a binary scene feature is added in both training and decoding.

The architecture of CNN is similar to the one described in [3]. It consists of the following:

- The network takes 60 frames of log Mel filter bank acoustic features with derivatives as an input (i.e., 0.6 seconds).
- Gaussian noise with 0.01 standard deviation is added during training
- First convolution layer consists of 80 filters (6x60 size, 1x1 stride) followed by max-pooling (4x3 pool shape, 1x3 stride)

[1]https://github.com/gorinars/dcase16-cnn
[2]https://github.com/librosa/librosa
[3]https://github.com/fchollet/keras
[4]http://sox.sourceforge.net/

- Second convolution layer has 80 filters (13 size, 11 stride) followed by max-pooling (13 pool size, 13 pool stride)
- Third and fourth layers are fully-connected with 1024 units per layer
- The output layer has sigmoid activation functions (1 unit per event with a separate unit for silence), which allows to classify overlapping events.

For all layers rectified linear units (ReLU) are used. To reduce over-fitting, dropout with rate 0.2 is applied to the first convolution layer and with rate 0.5 on the two fully-connected layers. L2 weight regularization is also used for all layers with a small 0.0001 penalty.

The network is trained using Adam algorithm [6] with learning rate 0.001 optimizing cross-entropy. The training ends when validation loss does not improve for 10 epochs. For data augmentation experiments the network is fine-tuned with stochastic gradient descent using only the original training data and not updating parameters of the first convolution layer. The parameters were tuned monitoring development data loss, although due to time constraints only a few architectures were compared.

2.4. Event detection

The decoder was not sufficiently modified compared to the baseline system. Event probabilities are extracted from CNN sigmoid activations. Then, a sliding smoothing window of 1 second length is applied. If the class is spotted within this window in more than 20 frames, the event is considered as detected in the central frame. In fact, in preliminary tests using no smoothing at all demonstrated similar results.

3. EXPERIMENTAL RESULTS

All experiments are conducted on 4-fold cross-validation set provided by the organizers. For the final submission, the system was trained on all annotated data.

Table 1 summarizes the average performance of the baseline system described in [4] compared to CNN trained without and with data augmentation.

Table 1: Segment-based overall metrics for baseline GMM system and the proposed model trained without (CNN) and with (CNN+DA) data augmentation

	ER, %			F1, %		
	base	CNN	CNN+DA	base	CNN	CNN+DA
home	0.97	0.94	0.92	15.4	20.4	24.6
residential	0.86	0.75	0.75	31.5	52.1	51.5
average	0.91	0.84	0.84	23.4	36.3	38.1

The model achieves 7.7% relative improvement of the segment error rate (ER) and 55.1% relative improvement of F-measure (F1). Data augmentation only allowed to slightly improve the average F-score.

Looking at the results per each event (Tables 2 and 3) we can conclude that the model significantly better classifies long events with larger amount of training data available (washing dishes, water tap running, bird singing, and car passing), while similar to the baseline it is unable to recognize short events with limited training data available.

Table 2: Segment-based F-score per class for home recordings. GMM baseline and the proposed model trained without (CNN) and with (CNN+DA) data augmentation

event	F1, %		
	base	CNN	CNN+DA
(object) rustling	3.2	0.0	0.0
(object) snapping	0.0	0.0	0.0
cupboard	0.0	0.0	0.0
cutlery	0.0	0.0	0.0
dishes	0.0	2.6	2.6
drawer	0.0	0.0	0.0
glass jingling	0.0	0.0	0.0
object impact	1.8	2.5	0.0
people walking	0.0	0.0	0.0
washing dishes	3.7	28.6	28.9
water tap running	15.9	58.3	65.8

Table 3: Segment-based F-score per class, residential area. GMM baseline and the proposed model trained without (CNN) and with (CNN+DA) data augmentation

event	F1, %		
	base	CNN	CNN+DA
(object) banging	0.0	0.0	0.0
bird singing	30.1	62.8	61.2
car passing by	54.5	64.4	68.3
children shouting	0.0	0.0	0.0
people speaking	25.0	14.0	5.0
people walking	1.7	0.0	0.0
wind blowing	11.8	2.1	1.8

4. CONCLUSION

A CNN based system submitted for DCASE 2016 challenge was described. Although many successful applications of CNN are reported in the literature for sound event detection, we could achieve only a small improvement compared to the baseline GMM-based system in terms of the challenge evaluation measure (segment error rate). Interestingly, other DCASE challenge participants also could not demonstrate significant error rate improvements. Overall, out of 17 participants our system was 12th in terms of ER and 3rd in terms of F-measure. Similar to our work, other systems had difficulties in detecting short events.

Neural network based models generally require more data for training. Disappointingly, not much improvement came from artificial data augmentation either. We plan to investigate whether using additional data can improve the results submitted for the challenge.

5. REFERENCES

[1] 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.

[2] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, “Exploiting spectro-temporal locality in deep learning based acoustic event detection,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, p. 1, 2015.

[3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proc. of MLSP*. IEEE, 2015, pp. 1–6.

[4] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. of EUSIPCO 2016*, Budapest, Hungary, 2016.

[5] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *Proc. of ICASSP*. IEEE, 2016, pp. 6440–6444.

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