AUDIO EVENT DETECTION USING MULTIPLE-INPUT CONVOLUTIONAL NEURAL NETWORK

Il-Young Jeong[1,2], *Subin Lee*[1,2], *Yoonchang Han*[2], *Kyogu Lee*[1]

[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 $\hat{t} = [\hat{t}_1, \cdots, \hat{t}_C]^T$ 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 $\pm$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:

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

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 $\pm$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

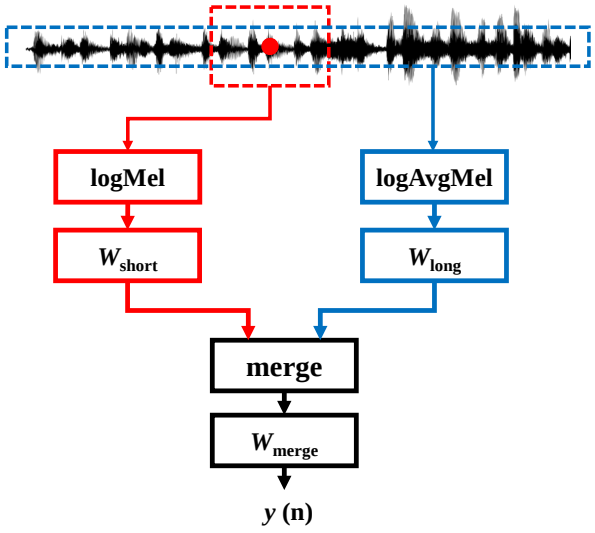

Figure 1: Framework for detecting events from the red dot point using short- and long-term data. Ws denote the respective concatenations of computational layers, including convolution, pooling, and fully-connected layer.

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 (mel×channel)×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×frame)	layer long-term	output size (filter×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×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:

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

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

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

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:

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

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 are more severely imbalance, 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 means 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 preprocessing 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.