Parallel Capsule Neural Networks for Sound Event Detection

Kai-Wen Liang[1], Yu-Hao Tseng[1], and Pao-Chi Chang[1,2]
[1]Department of Communication Engineering, National Central University, Taoyuan, Taiwan
[2]Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan
E-mail: kwistron@gmail.com, yhtseng.vaplab@gmail.com, pcchang@ce.ncu.edu.tw

***Abstract*—In this work, we propose a sound event detection system based on a parallel capsule neural network. The system takes advantage of the capability of capsule neural networks in the detection of overlapping objects. It further develops a parallel architecture and uses the kernel design of different shapes and sizes to effectively utilize the feature information to increase the detection accuracy. The experimental results show that the performance of the proposed system is as low as 52.34% measured by the error rate, which is even lower than the rank 1 system in DCASE2017 challenge.**

Keywords—Computational Auditory Scene Analysis, Sound Event Detection, Deep Learning, Capsule Neural Network

I. Introduction

In the past, humans used the human auditory system to identify sound events in the environment. With the development of computing technology, humans began to classify and detect sound data and derived more applications. Many research experts and scholars engaged in related research, such as the calculation of auditory scene analysis (CASA) [1], [2]. In 1990, Breman proposed auditory scene analysis [2] to promote CASA, and enhanced by Slaney [3]. In 1997, Sawhney presented the first study on acoustic scene classification (ASC) [4], [5]. The academic community has been actively engaged in reaching solutions to SED (sound event detection) problems. Since 2016, detection and classification of acoustic scenes and events (DCASE) competitions have been held regularly every year.

In recent years, due to the vigorous development of machine learning technology in artificial intelligence applications, many related researches began to develop on the basis of machine learning and deep learning. Taking DCASE2017 for example, many participants used CNN network architecture as the basis of system design. However, a conventional CNN network has its limitation in dealing with overlapping sound events, which makes it hard to improve the accuracy in the detection of multiple events. Vesperini *et. al.* used a capsule neural network (Capsnet) that has advantages for overlapping object detection [6] to effectively improve the performance of CNN in overlapping sound events [7]. This study expands the neural network into a parallel network architecture with the design of different sizes and shapes of the kernel to effectively utilize the original feature information. As a result, the performance of the proposed system is improved substantially.

This paper is organized as follows: it presents the proposed system, including single capsule networks and parallel capsule networks, in Section II. The experimental environments setting and results are discussed in Section III. Finally, Section IV concludes this work.

II. Proposed System

The proposed system architecture, consisting of the feature extraction, CNN layers, and the capsule neural network, is shown as in Fig. 1.

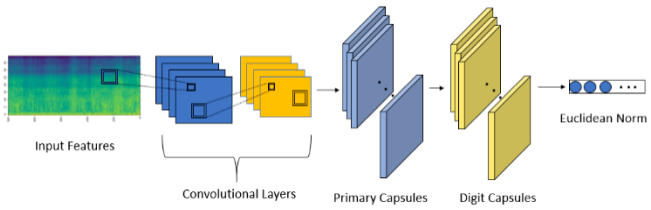

Fig. 1. The architecture of the proposed Capsnet system.

A. Feature preprocessing

All the audio data in the data set is down sampled to 16000Hz and then the log-Mel spectrogram is obtained through the procedure, shown as the flowchart in Figure 2. We apply STFT to compute 1024 points for each frame by using Hamming Window first. Then, the resulted STFT spectrogram is filtered by 40-band mel-scale fitter banks and then been taken logarithm to generate the log-mel spectrogram.

Fig. 2. Flowchart of generating log-mel spectrograms from audio waveforms.

B. Single Capsule Neural Network

The concept of using a capsule neural network architecture originates from its advantage in detecting overlapping events and its application in sound event detection [7]. Our proposed single capsule network architecture is represented in Fig. 1.

The parameters used in the proposed system are carefully chosen based on different experimental results. The proposed system includes two convolutional layers. The first convolutional layer performs a convolution over the input spectrogram with 4 filters and the second convolutional layer is with 16 filters in which the kernel shape is the same as the first convolutional layer. The width and the height of kernels in each layer are not symmetrical, that is not a squared kernel. The width and the height of kernel are 3 and 7, respectively, where 3 is in the frequency domain and 7 represents the axis in the time domain. The activation function used for kernels in both convolutional layers is rectifier linear units (ReLU).

There are two max-pooling layers which are placed separately between two convolutional layers and after the second convolutional layer. Specifically, max-pooling is only applied to the axis representing the frequency domain.

After CNN layers it fed output to the Primary Capsule Layer. Then the capsule layer is closely connected the L capsules, where L is the number of categories, i.e., sound events. Because the previous layer is also a capsule layer, a dynamic routing algorithm is used to calculate the output. Finally, it calculates the Euclidean length of each output capsule and generates the classification results.

C. Covered Area by Convolution and Pooling

As mentioned previously, our proposed method emphasizes on the importance of feature in the time domain on both kernel design and pooling process. Because the temporal resolution is more important than that in the frequency domain for sound event detection, no pooling is used in CNN layers to preserve enough temporal information. Therefore we need to slightly enlarge the kernel size to get more time information for effective feature extraction. In the frequency domain, similar to many conventional CNNs, a two-to-one maximum pooling is used. Our system design aims to reduce the complexity of the network, e.g., the number of CNN layers and the amount of necessary data acquisition, while the performance is maintained. We then discuss the influence of the covered area on both the kernel size and the pooling function.

Different kernel sizes will extract different features. Intuitively, the larger the kernel size is, the more information the extracted points represent, which refers that each output of a kernel carries information of more input points when the kernel size increases. The covered area defined by C for an output point with the specific kernel size in CNN layers is formulated as (1).

$$C = m + (l\text{-}1)(m\text{-}1) \quad (1)$$

where m is the size of the kernel and l is the number of convolution layers. In addition, when we take the pooling layer into consideration, the coverage C of a point will be enlarged and can be formulated by (2).

$$C = 2^l + \sum_1^l 2^{l\text{-}1}(m\text{-}1) \quad (2)$$

As illustrated in Figure 3, this example shows that one output depicted by the blue point carries information affected to 5 points in the original resolution when it is located on layer 2 with kernel size 3 in each layer.

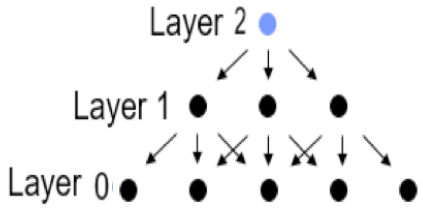

Fig. 3. Covered area by two convolution layers with kernel size 3

A pooling function can rapidly increase the covered area. In Figure 4, the covered area of the affected points by pooling is demonstrated. In this example, the blue point is the current output with two layers of pooling and totally 10 points are covered in the original data region when it is located on layer 2 with kernel size 3 in each layer.

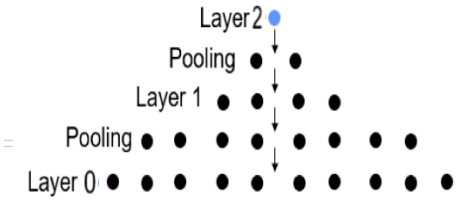

Fig. 4. Covered area by two convolution layers with pooling and kernel size 3

In our system, we have good results when we choose kernel size 7 in the time domain. That means this kernel size can preserve enough information to effectively present the original data. Further increasing the size may obtain better results but the complexity cost will increase, too. That may not be a practical realization. In the frequency domain, since we have used pooling, it is not necessary to take a large kernel size. If a large kernel size is chosen, the coverage will be too large so that the extracted features are not sensitive to details. The detailed experimental results are shown in Section IV.

D. Parallel Capsule Neural Network

In this section, we introduce the proposed parallel capsule neural networks with two types, early fusion and late fusion, in detail. First, each of the both systems has three parallelized convolution layers with three different kernel sizes which are determined as (3, 3), (5, 3), (7, 3), respectively. The reason for choosing different kernel sizes is to get detailed information on different resolutions. Other parameters of the Primary layer and the Digit layer remain the same as that in the single architecture.

The difference between the proposed two types is that the early fusion system concatenates the outputs of the parallel neural networks before sending them into the primary layer, as shown in Fig. 5. On the contrary, the late fusion system sends outputs of each network directly into the primary layers and concatenates the output to the digit layer, as shown in Figure 6. Both architectures allow the digit layer to access all features generated from parallel

CNN layers. However, the early fusion version might mix up the features generated from different CNNs too early in the system and degrade the function of parallelism. On the other hand, the late fusion architecture is able to keep the features from different CNNs more "pure" until the last full connection. Therefore, it is expected the late fusion version will perform better in the parallel architecture.

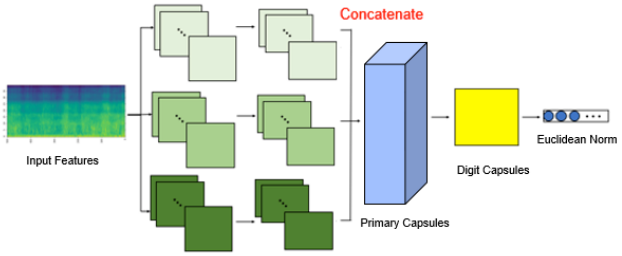

Fig. 5. Parallel capsule neural network (Early fusion)

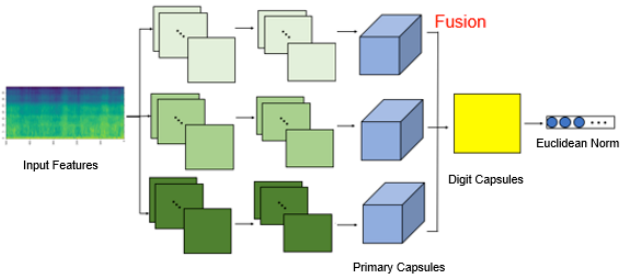

Fig. 6. Parallel capsule neural network (Late fusion)

III. Experimental Settings

This section describes the specifications of software and hardware, tools, and databases used in the experiments. In feature extraction, we use librosa library [8]. The experimental environment is shown as in Table I.

TABLE I. Simulation Environment

CPU	Core i7-8700K
GPU	ASUS TURBO-GTX1080TI * 2
RAM	Micron Crucial DDR4 2400/16G RAM
OS	Ubuntu-x64
Software language	Python3
Neural network tool	Tensorflow , Keras
Data feature extraction tool	librosa
Training setting	Batch size: 10 100 epochs Learning rate 1

A. Dataset

The data we use is TUT Sound Event 2017 [9], which is a database created in 2017 for DCASE. The database contains development datasets [9] and evaluation dataset [9]. Dataset is recordings of street acoustic scenes for a total of 121 minutes with 6 different labels. The 6 labels are brakes squeaking, car, children, large vehicle, people speaking, and people walking.

B. Evaluation Metrics [10]

The error rate (ER) is the main scoring standard for DCASE 2017. The ER score is calculated from an intermediate statistic of one second. It is the sum of substitution number ($S(k)$), insertion number ($I(k)$), and deletion number ($D(k)$). ER is calculated as

$$ER = \frac{\sum_{k=1}^{K} S(k)+\sum_{k=1}^{K} D(k)+\sum_{k=1}^{K} I(k)}{\sum_{k}^{K} N(k)} \quad (3)$$

$$S(k) = \min\left(FN(k), FP(k)\right) \quad (4)$$

$$D(k) = \max\left(0, FN(k)\text{-}FP(k)\right) \quad (5)$$

$$I(k) = \max\left(0, FP(k)\text{-}FN(k)\right) \quad (6)$$

where $N(k)$ is the total number of active groundtruth events. Since the error rate is based on substitution number, insertion number, and deletion number, it is possible to be greater than 1. Substitution number $S(k)$ is the case when the system detects an event in a given segment, but gives it a wrong label. $I(k)$ is after counting the number of substitutions per segment, the remaining false positives in the system output are counted. $D(k)$ is after counting the number of substitutions per segment, the remaining false negatives.

IV. Experimental Results

First, we used the data set provided by DCASE 2017 sound event detection competition to train the Capsule Neural Network. Then we compared it with DCASE2017 Baseline [9] and the first place architecture of DCAE2017 competition [11]. As shown in Table II, Our Capsule Neural Network resulted in the best performance. The preprocessing to get features for all works is to get the same log-mel spectrograms. After that, we also tried different kernel sizes by referring to the method proposed in [12]. As shown in Table III, the best combination of the kernel size is 3 in the frequency domain and 7 in the time domain, which obtained 57.12% in error rate. The increasing of the kernel size in the time domain results in better performance. However, a large kernel size in the frequency domain does not improve the performance. This can be explained by the discussion on the covered area in Section II-C.

TABLE II. Comparison of Capsule Neural Network with DCASE2017 Baseline and First CRNN Architecture

Network	Feature	Error Rate
Baseline [9]	mbe	0.9358
CRNN [11]	mbe	0.7914
Capsnets [7]	mbe	0.58
Capsnets (Single)	**mbe**	**0.5712**
Capsnets (Early)	**mbe**	**0.5347**
Capsnets (Late)	**mbe**	**0.5234**

TABLE III. COMPARISON OF DIFFERENT KERNEL SIZE

Networks	Kernel Size	Error Rate
Capsnets	3x3	**0.7105**
Capsnets	3x5	0.7414
Capsnets	3x7	0.8011
Capsnets	5x3	**0.6343**
Capsnets	5x5	0.6765
Capsnets	5x7	0.7052
Capsnets	**7x3**	**0.5712**
Capsnets	7x5	0.5831
Capsnets	7x7	0.5968

Based on the simulation results shown in Table IV, we also use different kernel sizes in the proposed parallel capsule network introduced in Section II-D and the results are represented in Table IV. The error rates of each combination of kernel size for early and late fusion structure are shown. The results show that the late fusion performs better than the early fusion. The possible reason is that the late fusion system can separate the features generated from each CNN till the last full connection stage and avoid early mixture of features to loss feature details.

For further analyzing, we change one pooling strategy in one of the three parallel capsule neural network. As shown in Table IV, average pooling in kernel size (3,3) is with better performance than max pooling. The reason is that the parallel system can learn both background information by average pooling and significant features by max pooling at the same time and results in better performance.

TABLE IV. COMPARISON OF SYMMETRIC KERNEL AND ASYMMETRIC KERNEL

Network	Feature	Kernel Size	Error Rate
Capsnets(Early)	mbe	(3,3)(5,5)(7,7)	0.5639
Capsnets(Early)	mbe	(3,3)(5,3)(7,3)	0.5498
Capsnets(Late)	mbe	(3,3)(5,5)(7,7)	0.5611
Capsnets(Late)	mbe	(3,3)(5,3)(7,3)	0.5321

TABLE V. COMPARISON OF DIFFERENT POOLINGS IN PARALLEL CAPSULE NEURAL NETWORKS

Network	Feature	Kernel Size	Pooling	Error Rate
Capsnets(Early)	mbe	(3,3)	Max	0.5498
		(5,3)	Max	
		(7,3)	Max	
Capsnets(Early)	mbe	(3,3)	Average	0.5347
		(5,3)	Max	
		(7,3)	Max	
Capsnets(Late)	mbe	(3,3)	Max	0.5321
		(5,3)	Max	
		(7,3)	Max	
Capsnets(Late)	mbe	(3,3)	Average	0.5234
		(5,3)	Max	
		(7,3)	Max	

V. CONCLUSION

In this paper, we proposed the architecture of Parallel Capsule Neural Networks which includes asymmetric kernels and different pooling with log-mel spectrogram input. The parallel architecture uses different kernel sizes and pooling functions to capture different features for classification. The experimental results show that the error rate can be reduced to 52.34%, which is 26% better than the first place of DCASE challenge 2017, and 5% better than the state-of-the-art system. The proposed system is able to detect multiple and overlapped events. It exhibits great potential for many applications such as autonomous vehicle driving.

ACKNOWLEDGMENT

This work was supported in part by Ministry of Science and Technology under grant no. MOST 108-2634-F-008-004 and MOST 107-2218-E-009-062 through Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan.

REFERENCES

[1] D. Wang and G. J. Brown, “Computational Auditory Scene Analysis: Principles, Algorithms, and Applications”. Wiley-IEEE Press, 2006.

[2] A. S. Bregman, “Auditory Scene Analysis,” MIT Press, Cambridge, MA, 1990.

[3] M. Slaney, “The History and Future of CASA,” Speech separation by humans and machines, pp.199-211, Springer US, 2005.

[4] N. Sawhney, “Situational Awareness from Environmental Sounds,” Technical Report, Massachusetts Institute of Technology, 1997.

[5] D. Barchiesi, D. Giannoulis, D. Stowell, M. D. Plumbley, “Acoustic Scene Classification,” in IEEE Signal Processing Magazine, vol. 32, no. 3, pp.16-34, May 2015.

[6] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Advances in neural information processing systems, p. 3856-3866, 2017.

[7] F. Vesperini, et al. "Polyphonic Sound Event Detection by using Capsule Neural Networks." IEEE Journal of Selected Topics in Signal Processing, 2019.

[8] Librosa: an open source Python package for music and audio analysis, https://github.com/librosa, retrieved Dec. 1, 2016.

[9] A. Mesaros, et al, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events, 2017.

[10] A. Mesaros, et al, “TUT Database for Acoustic Scene Classification and Sound Event Detection,” IEEE 2016 24th European Signal Processing Conference, pp. 1128-1132, Aug. 2016.

[11] S. Adavanne , and T. Virtanen, “A report on sound event detection with different binaural features,” arXiv preprint arXiv:1710.02997 , 2017.

[12] Y. C. Wu, P. C. Chang, C. Y. Wang and J. C. Wang, “Asymmetrie Kernel Convolutional Neural Network for acoustic scenes classification, ” in 2017 IEEE International Symposium on Consumer Electronics (ISCE), Kuala Lumpur, Malaysia, Nov. 2017.