EURASIP Journal on Audio, Speech, and Music Processing

RESEARCH Open Access

Improved capsule routing for weakly labeled sound event detection

Haitao Li[1], Shuguo Yang[1*] and Wenwu Wang[2]

Abstract

Polyphonic sound event detection aims to detect the types of sound events that occur in given audio clips, and their onset and offset times, in which multiple sound events may occur simultaneously. Deep learning–based methods such as convolutional neural networks (CNN) achieved state-of-the-art results in polyphonic sound event detection. However, two open challenges still remain: overlap between events and prone to overfitting problem. To solve the above two problems, we proposed a capsule network-based method for polyphonic sound event detection. With so-called *dynamic routing*, capsule networks have the advantage of handling overlapping objects and the generalization ability to reduce overfitting. However, dynamic routing also greatly slows down the training process. In order to speed up the training process, we propose a weakly labeled polyphonic sound event detection model based on the improved capsule routing. Our proposed method is evaluated on task 4 of the DCASE 2017 challenge and compared with several baselines, demonstrating competitive results in terms of *F*-score and computational efficiency.

Keywords: Polyphonic sound event detection, Capsule network, Weakly labeled, Dynamic routing

1 Introduction

Sound event detection (SED) is the task of accurately marking the onset and offset time information of each event and its type in the input sound signal. With SED, the computer can understand and respond to the surrounding environment through sound. SED can be applied to many fields, such as environment-aware speech recognition [1], remote medical monitoring [2], equipment monitoring [3] and bio-acoustical monitoring [4].

SED systems can generally be divided into two categories, monophonic and polyphonic. The former detects at most one sound event at each time instance, while the latter can detect two or more concurrent sound events [5]. The polyphonic SED is more useful in practical applications as multiple sound events often occur simultaneously in practice. However, it is a more challenging task to design the polyphonic SED system, as it is difficult to extract effective features to separate multiple overlapping sound events [6].

Traditionally, the classifiers used in sound event detection include the Gaussian mixture model (GMM) [7], hidden Markov model (HMM) [8], support vector machine (SVM) [9], and non-negative matrix factorization (NMF) [10]. Despite their promising performance, most of these classification methods are developed for the task of monophonic sound event detection.

In recent years, with the increase in the amount of training data, the models used for classification have gradually become more complex. Among them, deep learning has achieved remarkable performance in the field of pattern recognition including sound event detection. As an example, convolutional neural networks (CNN) can directly process multi-dimensional features and extract high-dimensional abstract information through the convolution method with local weight sharing and pooling operation. This can mitigate impacts of translation and scaling ambiguities, and enhance the generalization ability of the model. In [11], the author proposed a CNN-based network structure for environmental

* Correspondence: ysg_2005@163.com
[1]College of Mathematics and Physics, Qingdao University of Science and Technology, No.99 Songling Road, Qingdao 266061, China
Full list of author information is available at the end of the article

sound classification, which contains two convolutional layers and two fully connected layers. The convolutional recurrent neural networks (CRNN) based methods have achieved state-of-the-art results in the SED task, especially in the case of polyphonic SED [5]. CRNN has CNN properties with local displacement invariance and the ability to model short-term and long-term time dependence provided by the RNN layer.

The capsule network [12] has the ability to "route" from low-level features to high-level features, which greatly improves the ability of the model in detecting objects from overlapping features. In the capsule network, local information from input features is used to predict all classes by routing operations. Since the local information at different locations contributes differently to the prediction of different classes, the model can still detect the events even when they overlap with each other. However, the iterative operations in dynamic routing make it computationally more expensive than other deep networks [13], such as AlexNet [14] and ResNet [15].

Through the combination of capsule structure and dynamic routing algorithm, the network can identify the whole-part relationship between features [12], such as the position of the nose with respect to the face in an image. The capsule-based method has achieved promising results in the classification of highly overlapping handwritten digital images. Recent research shows that the capsule network is also suitable for sound event detection [16] which uses gated convolution for feature extraction, and at the end, an attention layer parallel to the high-level capsule layer is added to implement the attention mechanism. The final prediction result is obtained by merging the outputs of the two layers. The algorithm was evaluated on the weak label dataset of the DCASE 2017 Challenge and achieved improved performance. In [17], the capsule network was used for polyphonic SED, which significantly improves the performance of CNN in overlapping object detection. In [18], the author proposed a parallel capsule neural network–based system for sound event detection and used different shapes and sizes of convolution kernels to improve detection accuracy.

Traditionally, researchers have trained SED models using strong labels that specify the types of sound events and the onset and offset times. However, manually obtaining such a label is very cumbersome and labor-intensive. Therefore, in practice, the training data is often weakly labeled, i.e., only the types of sound events are provided, but without their onset and offset times. For example, AudioSet [19] released by Google is a large-scale weakly labeled corpus of this kind.

For weakly labeled SED, multiple instance learning (MIL) is a popular framework [20]. In the weakly labeled SED, the audio clip is divided into many frames, each frame is regarded as an instance, and the whole clip is a bag. We only know the label of bags but not the label of each instance. In this approach, the neural network predicts the probability of each sound event occurring in each frame. The frame-level probabilities are then aggregated to the clip-level probability by a pooling function.

Methods based on CNN, RNN, and their combinations are shown to provide state-of-the-art performance on the weakly labeled SED task. In [21], the authors proposed a CRNN for weakly labeled SED with learnable gated linear units (GLUs). By GLUs, the network can focus on sound events and ignore irrelevant sounds. In [22], bidirectional long short-term memory (BLSTM) was applied to weakly labeled learning. Since weakly labeled data was used, accurate error calculation cannot be performed to update the parameters of BLSTM. The authors introduce connectionist temporal classification (CTC) to calculate the loss. At the same time, the adaptive pooling operators are shown to offer better performance on the weakly labeled SED task compared with commonly used pooling operators, such as max-, or average-pooling [23–25]. Although these methods have obtained promising results, they have not fully addressed the problems such as overfitting. This is especially a problem for small data sets. Although the capsule-based method [16] is not prone to overfitting, the complex dynamic routing process will greatly slow down the model training process, resulting in a longer model training time. Here, we focus on addressing these limitations.

This paper proposes a weakly labeled polyphonic SED model based on the improved capsule routing, in which the convolution layer uses the early fusion parallel convolution structure proposed in [18] and adds a recurrent layer after the capsule layer to perform the time-dependent modeling. The computational cost of the conventional capsule network is much higher than that of the convolutional neural network due to the iterative operations used, which lead to slow speed in network training. In this paper, we replace the conventional dynamic routing with the weight-sharing dynamic routing proposed in [26] to reduce the amount of parameters and speed up the training process. The results show that, the model greatly accelerates the training speed. We evaluated the model on the weakly labeled SED task data set of the 2017 DCASE Challenge [27]. The experimental results show that the proposed method offers significant performance improvement as compared with the capsule baseline in [16]. Overall, the main contributions of our work are summarized below.

- First, we introduce an early fusion–based parallel convolution structure with a recurrent layer, so that the model can make an effective use of the original feature information.

- Second, we apply the capsule network to weakly labeled data sets by leveraging its strong generalization ability, which helps address the data scarcity issue in model training.
- Third, we use weight-sharing dynamic routing [26] to reduce computational overhead and the number of model parameters and to improve the training speed.

2 Capsule network

The concept of capsule network was proposed by Hinton in 2017 [12]. The main idea is to change the input and output of neurons from a scalar form to a vector form, in order to reduce the loss of feature information and improve the feature extraction ability of the model. Capsule vector neurons and normal scalar neurons are shown in Fig. 1, where x_i, $i = 1, 2, \ldots, n$ represents the input of the scalar neuron, $\boldsymbol{u}_i$, $i = 1, 2, \ldots, n$ represents the low-level capsule, $\hat{\boldsymbol{u}}_j, j = 1, 2, \ldots, n$ represents the prediction of $\boldsymbol{u}_i$ to high-level capsule $\boldsymbol{v}$, b represents bias and w_i, $i = 1, 2, \ldots, n$ represent corresponding weights, Σ represents weighted summation, and g and *squash* represent activation functions.

The length of the capsule represents the probability of occurrence of the corresponding category, and each dimension of the capsule represents some attributes of the modeled category. Each layer of the capsule network has a number of nodes, with each node representing a capsule. The weight connecting the low-level capsule with the high-level capsule will change in the learning process, which causes the change of the node connection degree, via dynamic routing.

The operation of the dynamic routing algorithm is shown in Fig. 2.

For the high-level capsule j, its output $\boldsymbol{v}_j$ can be calculated as follows:

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i \tag{1}$$

$$\boldsymbol{s} = j\sum_i c_{ij}\hat{\boldsymbol{u}}_{j|i} \tag{2}$$

$$\boldsymbol{v}_j = \frac{\left\|\boldsymbol{s}_j\right\|^2}{1+\left\|\boldsymbol{s}_j\right\|^2}\frac{\boldsymbol{s}_j}{\left\|\boldsymbol{s}_j\right\|} \tag{3}$$

where $\boldsymbol{u}_i$ is the output of low-level capsule i, $\hat{\boldsymbol{u}}_{j|i}$ is the prediction of $\boldsymbol{u}_i$ to $\boldsymbol{v}_j$, and $\boldsymbol{W}_{ij}$ is the corresponding weight matrix. All prediction vectors of $\boldsymbol{v}_j$ are weighted by a set of coupling coefficients c_{ij} and are compressed by a nonlinear squashing function (3) to represent the probability of the existence of the target. The coupling coefficient c_{ij} in (2) is determined by a dynamic routing algorithm. The updated formula for c_{ij} is as follows:

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{4}$$

$$b_{ij} \leftarrow b_{ij} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j \tag{5}$$

The higher the similarity between $\hat{\boldsymbol{u}}_{j|i}$ and $\boldsymbol{v}_j$ (measured by the inner product), the more increment c_{ij} gets.

In (5), b_{ij} is the parameter used to update the coupling coefficient c_{ij}. In each forward propagation, b_{ij} is initialized to 0, and the initial value of the coupling coefficient c_{ij} is calculated from Eq. (4), and then $\boldsymbol{v}_j$ is calculated from the forward propagation of the network. The value of b_{ij} is updated in terms of (5), which is used to update the value of c_{ij} and to further modify the value of $\boldsymbol{s}_j$ by forward propagation in order to change the value of the output vector $\boldsymbol{v}_j$. Finally, an optimal set of coupling coefficients is obtained.

The differences between vector neurons and scalar neurons are summarized and presented in Table 1.

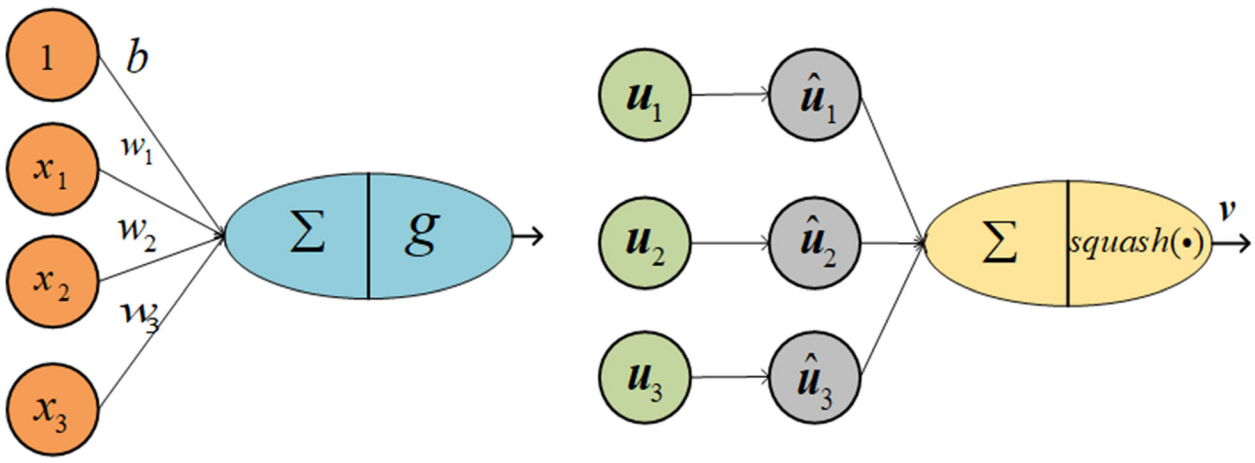

Fig. 1 Scalar and vector neurons

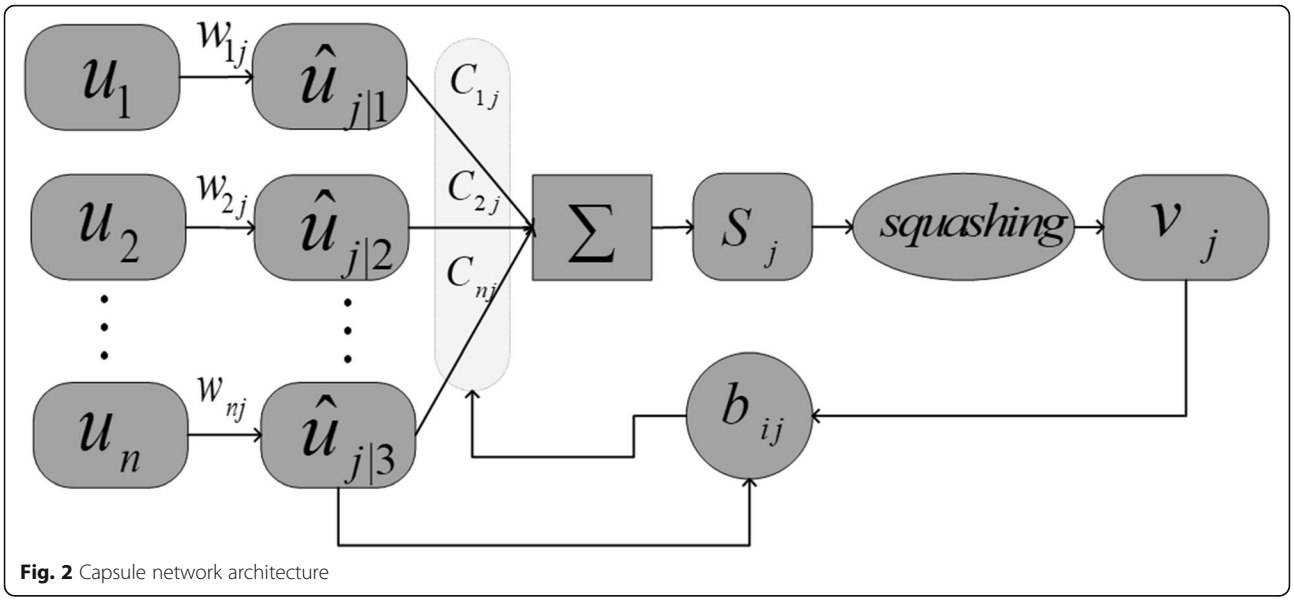

Fig. 2 Capsule network architecture

3 Polyphonic sound event detection model based on capsule network with weak labeling

3.1 Parallel convolutional layer

In this section, the parallel convolution layer proposed in [18] will be introduced, as shown in Fig. 3. In the capsule network, the parallel convolution layer is used to extract local features from the input features.

The parallel convolution layer is composed of three parallelized convolution layers with the same parameters except kernel sizes. The kernel size is set asymmetrically in the frequency axis and time axis, and the larger size is used in the time axis to obtain more temporal information [28]. In the convolutional layer, different convolution kernel sizes are used to obtain information of different resolutions. After each convolution layer, max pooling is used to reduce the dimensionality. Through the features extracted by the parallel convolutional layers, the model can obtain information of different resolutions, so that the original feature information can be used more efficiently.

Table 1 Differences between vector neurons and scalar neurons

	Vector neurons	Scalar neurons
Input	$\boldsymbol{u}_i$	x_i
Affine transformation	$\hat{\boldsymbol{u}}_{j\|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i$	-
Weighted sum	$\boldsymbol{s} = j\sum_i c_{ij}\hat{\boldsymbol{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
Nonlinear activation	$\boldsymbol{v}_j = \frac{\|\boldsymbol{s}_j\|^2}{1+\|\boldsymbol{s}_j\|^2}\frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|}$	$h_j = g(a_j)$
Output	$\boldsymbol{v}_j$	h_j

3.2 Improved dynamic routing

In the dynamic routing process proposed in [12], each low-level capsule $\boldsymbol{u}_i$, $i = 1, 2, \ldots, n$ must be multiplied by a weight matrix to obtain the prediction vector of the high-level capsule $\boldsymbol{v}_i$, $i = 1, 2, \ldots, n$, as shown in Fig. 4.

In order to speed up the training process and reduce the number of parameters, the weight-shared capsule [26] was used in this work. The idea is based on weight sharing in convolution networks, where weight matrices are shared between capsules. For the high-level capsule j, the transformation matrix connected to all the low-level capsules is shared, so the number of weight matrices is the same as the number of high-level capsules (as shown in Fig. 5). Equation (1) can be expressed as:

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_j\boldsymbol{u}_i \tag{6}$$

By sharing the weight matrix among the capsules, the number of parameters of the model can be greatly reduced, and the training speed of the model can be significantly improved.

3.3 Proposed method

The architecture of the proposed method is shown in Fig. 6, which includes three parts: convolutional layer, capsule layer, and recurrent layer. Hyperparameters used are presented in Tables 2 and 3.

3.3.1 Convolutional layer

The convolutional layer consists of three parallel convolutional paths, each path contains 4 convolutional layers, and the convolution kernel of the same path has the same size.

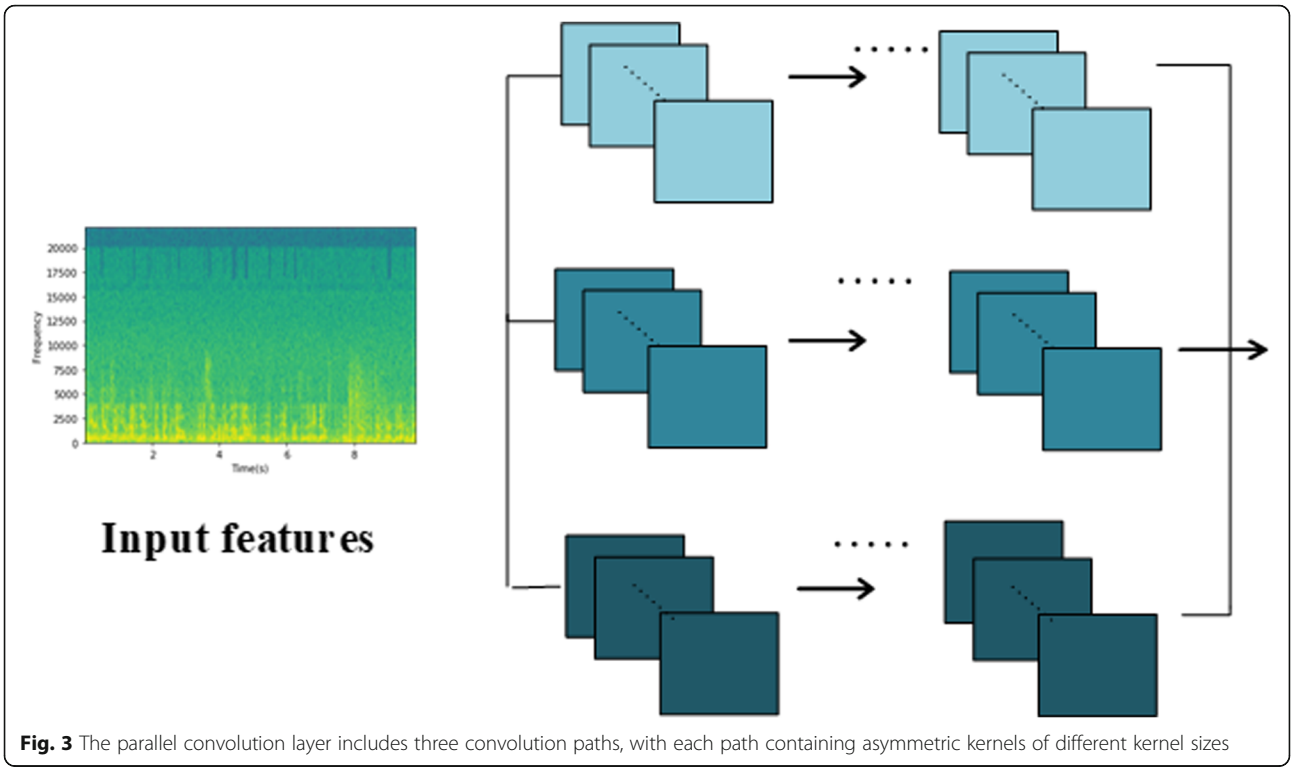

Fig. 3 The parallel convolution layer includes three convolution paths, with each path containing asymmetric kernels of different kernel sizes

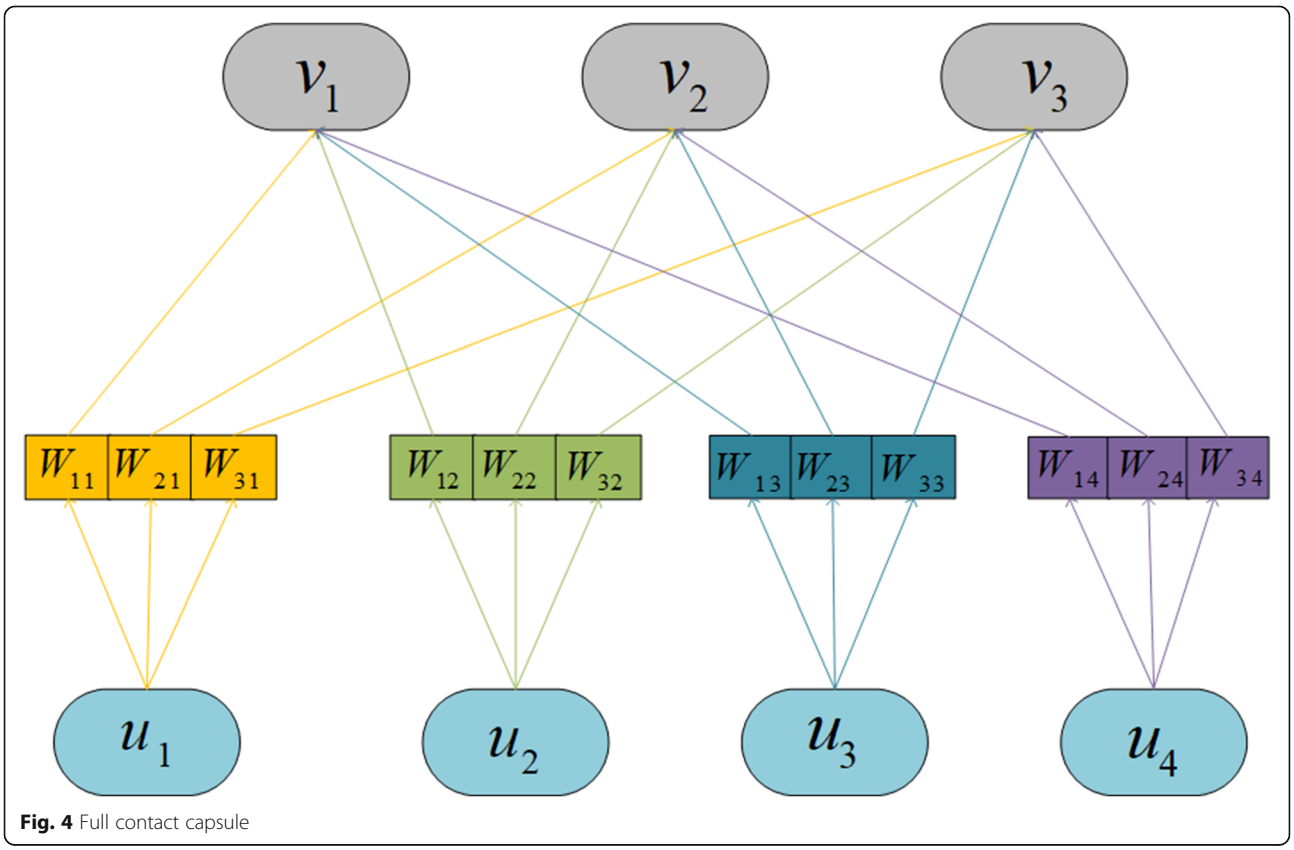

Fig. 4 Full contact capsule

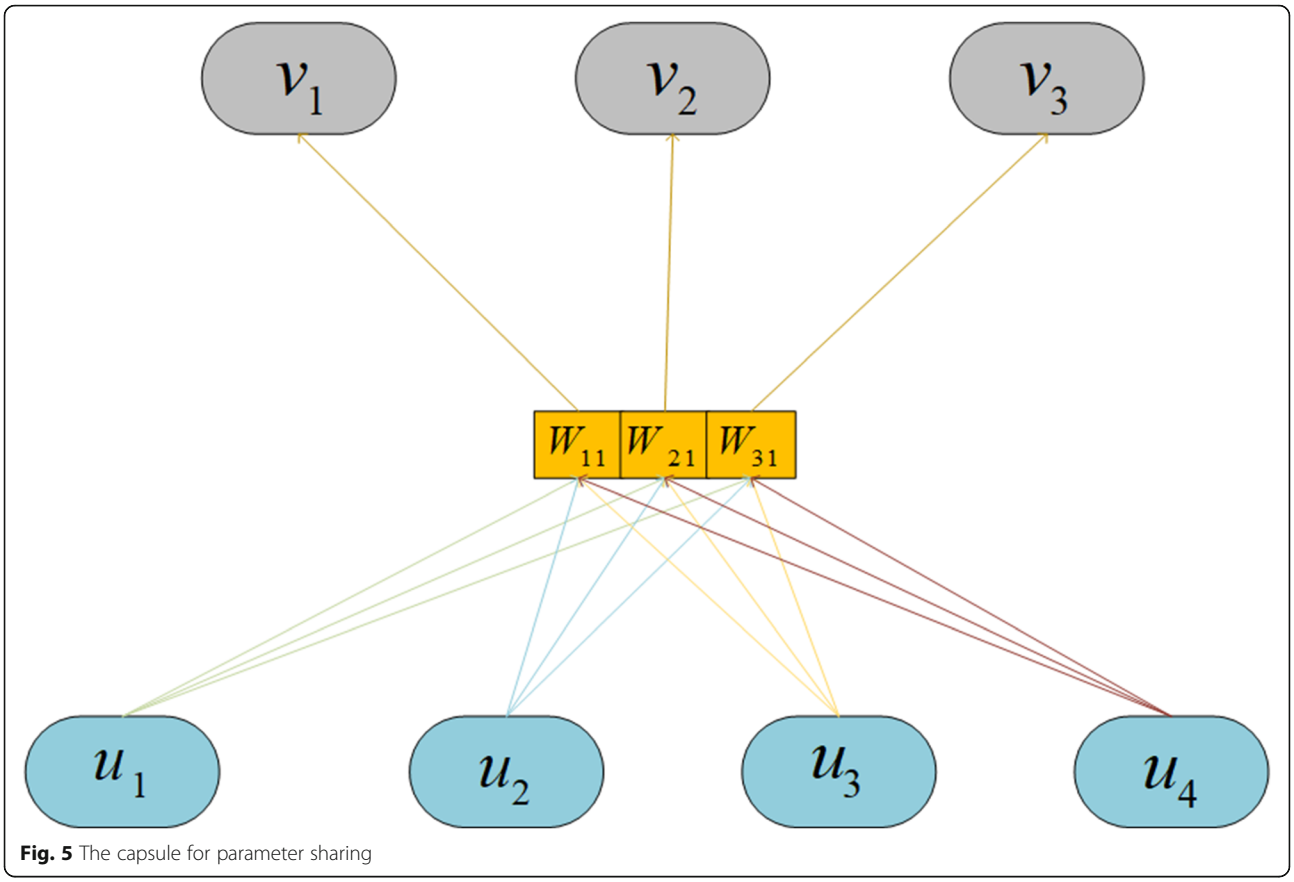

Fig. 5 The capsule for parameter sharing

Before being transported to the capsule layer, the output of the convolution layer is concatenated so that the capsule layer has all the features of the convolution layer. The input feature vector has a shape $T \times F$, where T is the number of time frames, and F is the number of frequency bins in the input feature. The output of the convolutional layer is a tensor of dimension $T' \times F' \times Q$, where Q is the number of the feature maps obtained after the outputs of the three paths are concatenated, and T' and F' are the number of frames and frequency bands after feature extraction of the convolutional layer.

3.3.2 Capsule layer

The first capsule layer is a convolutional layer with 32 channels, each consisting of an 8-dimensional capsule. In the second capsule layer, the prediction vector of the high-level capsule is first calculated by multiplying the output of the low-level capsule by a weight matrix. Then

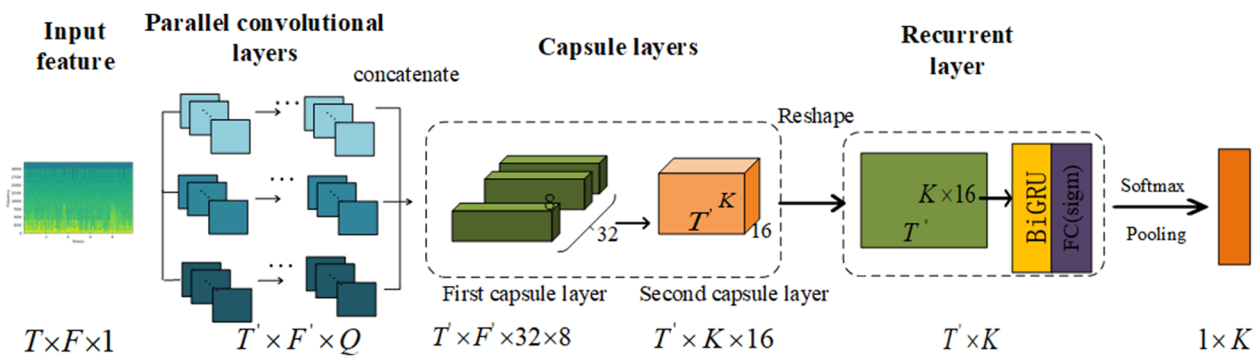

Fig. 6 The proposed neural network structure, which consists of three parts. (1) Feature extraction: parallel convolution layer with different kernel sizes. (2) Capsule layer: the outputs of convolutional layers are fed into two capsule layers. (3) Recurrent layer: a bidirectional GRU and one FC layer are used to learn temporal context information and estimate event activity probabilities

Table 2 Model parameters (feature extraction)

	Feature extraction			
	Conv1	Conv2	Conv3	Conv4
Kernel size[a]	64 @ 3 × 3	64 @ 3 × 3	64 @ 3 × 3	64 @ 3 × 3
Stride	1 × 1	1 × 1	1 × 1	1 × 1
Pooling size	1 × 2	1 × 2	2 × 2	2 × 2
Activation function	ReLU	ReLU	ReLU	ReLU
Num. of hidden units	–	–	–	–
Capsule dimension	–	–	–	–

[a]Take a path as an example

the prediction vector is weighted and summed to obtain the output vector, the weight of which is determined by the so-called dynamic routing, and its purpose is to allow the low-level capsule to independently choose the best path to transmit information to the high-level capsule. Finally, the output vector is non-linearly mapped to obtain the final output vector.

The main process of the capsule layer can be summarized as follows:

- In the first capsule layer, $T^{'} \times F^{'}$ 8-dimensional capsules are used in each channel.
- The second capsule layer is used to calculate the K 16-dimensional high-level capsules representing event categories based on the input capsules from each frame, which leads to a tensor of shape $T^{'} \times 16 \times K$.

3.3.3 Recurrent layer

A recurrent layer is added after the capsule layer to learn temporal context information, as in [5, 6, 29]. The output is reshaped into a tensor of dimension $T^{'} \times (K \times 16)$ after the second capsule layer, by combining K 16-dimensional capsules in each frame. Then the bidirectional gated recurrent unit (GRU) is adopted to learn temporal context information from the combined vectors. Use a feedforward layer with a sigmoid activation function after the recurrent layer to get probabilities of event activity per frame. The output of the recurrent layer is a tensor of $T^{'} \times K$. By adding a recurrent layer, the model can learn more temporal information, thereby improving the accuracy of temporal localization of events.

The output of the recurrent layer is frame-level prediction, that is, the probability of each sound event on each frame. However, in the case of weak labeling, there are no frame-level tags, and only clip-level tags are available. Therefore, it is necessary to aggregate the frame-level predictions into clip-level predictions to calculate the loss function. We use the softmax pooling function, which is defined as follows:

$$y_l = \frac{\sum_i y_i \exp(y_i)}{\sum_i \exp(y_i)} \tag{7}$$

where $y_i \in [0, 1]$ is the frame-level predicted probability of one sound event type, and $y_l \in [0, 1]$ is the clip-level aggregation probability of the event.

The softmax pooling function calculates y_l as a weighted average of y_i, where a larger y_i will yield a larger weight. In this way, the clip-level probability is mainly determined by the larger frame-level probability, but a frame with a lower probability also has a chance to receive an error signal.

Choosing a threshold τ_1 for event l, when $y_l > \tau_1$, the event occurs. In order to calculate the onset time and offset time, another value τ_2 is selected as the frame-level prediction threshold, and then the onset and offset time can be determined from the obtained binary matrix.

4 Experiment

4.1 Datasets and performance indicators

Because the method we propose is for weakly labeled polyphonic SED, we use the weakly labeled data set provided by DCASE 2017 task 4 for evaluation. This data set is a subset of AudioSet, consisting of 17 sound events, divided into two categories: "Warning" and "Vehicle". Each audio segment has a maximum duration of 10 s and may correspond to more than one potentially overlapping sound event. We evaluated two tasks on this dataset: audio tagging and sound event detection.

Table 3 Model parameters (capsule layers, recurrent layer)

	Capsule layers		recurrent layer	
	First capsule layer	Second capsule layer	GRU	FC
Kernel size[a]	32 @ 3 × 3	–	–	–
Stride	1 × 1	–	–	–
Pooling size	–	–	–	–
Activation function	Squashing	Squashing	–	Sigmoid
Num. of hidden units	–	–	256	17
Capsule dimension	8	16	–	–

[a]Take a path as an example

Among them, audio tagging aims to predict the type of sound event included in an audio clip, and the sound event detection also predicts the onset and offset time of the event. For these two tasks, the precision, recall, and the micro-average of the *F*-score are used to evaluate the performance of the model. For SED, we also calculated a segment-based error rate with 1-s resolution. The *sed_eval* [30] toolbox was used to evaluate SED tasks.

4.2 Baseline system

We compare the proposed method with the following baseline systems:

GCCaps is the system proposed in [16], which consists of three gated convolution blocks, two capsule layers, and a layer of attention parallel to the high-level capsule layer. Each convolution block is composed of two gated convolution layers.

GCRNN is the system proposed in [21], where the gated linear units are used to replace the ReLU activation function, and shown to provide the state-of-the-art performance in the audio tagging subtask of DCASE 2017 task 4. In the system, the gated convolutional layer is used to extract features, which are stacked over the frequency axis and then passed to a bi-directional GRU layer. The outputs of the two parallel feedforward layers with different activation functions are fused to get the final output.

GCNN is the baseline from work [16], similar to GCRNN but without a recurrent layer.

Capsnets-RNN is the method proposed in the first section of this article. The output of parallel convolutional layer is concatenated before being sent to the capsule layer. In the training process, the pooling function aggregates the frame-level prediction probabilities into the clip-level prediction and performs iterative training by calculating the binary cross-entropy loss.

Capsnets-RNN (conventional routing) is similar to the proposed Capsnets-RNN, except that the conventional routing algorithm is used in the dynamic routing part.

Capsnets-RNN (single) uses a single path convolutional layer. The convolutional kernel uses the 7×3 convolutional kernel, where the frequency axis is 3 and the time axis is 7.

Table 4 Performance results of audio tagging subtask

Method	*F*-score	Precision	Recall
Capsnets-RNN	61.1%	53.9%	70.4%
Capsnets-RNN (conventional routing)	**61.5%**	54.2%	71.1%
Capsnets-RNN (single)	55.2%	52.3%	62%
GCCaps	58.6%	59.2%	57.9%
GCRNN	57.3%	53.6%	59.6%
GCNN	57.2%	59.0%	57.2%

Table 5 Performance results of sound event detection subtask

Method	*F*-score	Precision	Recall	Error rate
Capsnets-RNN	50.5%	60.4%	43.4%	0.74
Capsnets-RNN (conventional routing)	49.9%	60.1%	42.7%	**0.73**
Capsnets-RNN (single)	49.5%	59.7%	42.3%	0.79
GCCaps	46.3%	58.3%	38.4%	0.76
GCRNN	43.3%	57.9%	34.8%	0.79
GCNN	37.5%	46.6%	31.1%	0.88

4.3 Experimental setup

To make a fair comparison with [16], we also use the log-Mel spectrogram to be the input feature, which is essentially a short-time Fourier transform, followed by a Mel filter bank and logarithmic nonlinear operation. Before extracting features, each audio segment was resampled to 16 kHz. The logarithmic Mel feature is calculated using 64ms frame length, 20ms overlap, and 64 Mel frequency units per frame. For each 10-s sample, a feature vector of 240×64 will be generated.

In order to reduce the occurrence of over-fitting and accelerate the speed of convergence, we use batch normalization after each convolutional layer and primary capsule layer [31]. For the number of routing iterations, we set it to $r = 4$. We use an Adam optimizer for training and set the learning rate to 0.001 which was decayed by a factor of 0.9 every two epochs. For loss function, we used binary cross-entropy and calculated the gradient by a mini-batch size of 44. A total of 30 epochs were trained.

The number of events in the test set and the evaluation set is balanced, while the training set is not, which will lead to classification bias. The data balancing technique proposed in [21] was used to reduce the impact of this problem.

In the inference process, we average the predictions of the obtained models to get the final result. Here we choose the five models (epochs) with the highest accuracy on the validation set. In our system, the detection threshold is set to $\tau_1 = 0.3$ and $\tau_2 = 0.6$. For SED, the expansion and corrosion sizes are set to 10 and 5, respectively.

Table 6 Comparison of the FLOPs, number of trainable parameters (the second capsule layer), and training time per step

Method	FLOPs	Params	Seconds/step
Capsnets-RNN (conventional routing)	3423786	278528	3.1
Capsnets-RNN (weight sharing routing)	2871082	2176	2.6

Table 7 Comparison of different kernel size

Kernel size	Error rate
3 × 3	**1.10**
3 × 5	1.21
3 × 7	1.15
5 × 3	**1.09**
5 × 5	1.22
5 × 7	1.24
7 × 3	**0.79**
7 × 5	1.24
7 × 7	1.25

4.4 Results and discussion

The results of audio tagging and sound event detection are shown in Table 4 and Table 5. For audio tagging, the early fusion method using conventional routing performs best, with an *F*-score of 61.5%, which is slightly higher than the weight-sharing strategy routing method. The single-path model score is slightly higher than GCCaps. In this subtask, GCRNN and GCNN provide roughly the same performance.

For SED, Capsnets-RNN (conventional routing) achieved the highest *F*-score and the lowest error rate. The error rate of the single path model is higher than that of Capsnets-RNN. GCCaps performed slightly better than GCRNN, with an F-score of 46.3% and an error rate of 0.76. The introduction of the recurrent layer enhances the time positioning ability of the GCRNN model, because its score is significantly higher than that of GCNN, and the error rate is relatively low.

In order to show the benefit for training speed brought by weight sharing, we show in Table 6 the floating-point operations (FLOPs) [32] of conventional routing and weight sharing routing, the number of trainable parameters of the second capsule layer, and the training time per step. It can be seen from Table 6 that the weight-sharing strategy can greatly reduce the complexity of the model. Among them, weight sharing reduces the number of FLOPs of the model by 16%, and the trainable parameters of the second capsule layer by 99%. As a result, the training speed is increased by 0.5 s/step.

We also refer to the method proposed in [28] and tested different kernel sizes in Capsnets-RNN (single). As shown in Table 7, the model performance is optimal when the kernel size is 7 × 3. The increase of the kernel size in the time axis results in better performance in the SED subtask. However, further increasing the convolution size in the time axis will increase the complexity of the model. Increasing the convolution kernel size in the frequency axis did not seem to improve the performance. The main reason is that the pooling operation is performed along the frequency axis. If the size of the convolution kernel is large, the features extracted may be from the extended area rather than the local details of the signal [18].

Table 8 and Table 9 show the results of all sound events by the proposed model on the two subtasks. For audio tagging, some sound events such as "civil defense siren" and "screaming" have higher classification accuracy while other sound events such as "car passing by" and "bus" are difficult to recognize. For SED, some classes such as "civil defense siren" and "train" have a lower error rate, while other classes such as "bicycle" and "car passing by" have a higher error rate.

We use the improved dynamic routing method proposed in Section 3.2 in the Capsnets-RNN model. In the experiments, we found that this method can reduce the complexity of the model. Specifically, it can reduce the amount of floating-point calculations and trainable parameters of the model, thus speeding up the training process. At the same time, the routing process of the capsule network enables the model to recognize the whole-part relationship, which can enhance the generalization ability of the model. Coupled with the parallel convolutional layer, the model can make full use of the original feature information. Finally, the use of the asymmetric convolution kernel and the addition of RNN enable the model to capture more temporal information from data. Therefore, the model achieves promising performance on both subtasks of audio tagging and sound event detection.

These results show that the use of an improved dynamic routing method leads to a small reduction in the accuracy of audio tagging subtask, but compared to the improvement in training speed, the drop in performance is minor. At the same time, the introduction of RNN can enable the model to learn contextual information and enhance the model's ability in temporal localization.

Table 8 *F*-score of audio tagging subtask for each event

Train horn	Air horn, truck horn	Car alarm	Reversing beeps	Bicycle	Skateboard	Ambulance (siren)	Fire engine, fire truck (siren)	Civil defense siren
59.1%	66.7%	64.3%	46.2%	44.8%	63.5%	65.2%	64.3%	85.0%
Police car (siren)	**Screaming**	**Car**	**Car passing by**	**Bus**	**Truck**	**Motorcycle**	**Train**	**Micro average**
61.3%	86.4%	66.7%	34.9%	40.2%	46.2%	61.3%	75.6%	61.1%

Table 9 Error rate of sound event detection subtask for each event

Train horn	Air horn, truck horn	Car alarm	Reversing beeps	Bicycle	Skateboard	Ambulance (siren)	Fire engine, fire truck (siren)	Civil defense siren
1.12	0.77	0.87	0.88	1.41	0.92	0.95	0.91	0.35
Police car (siren)	**Screaming**	**Car**	**Car passing by**	**Bus**	**Truck**	**Motorcycle**	**Train**	**Error rate**
0.85	0.86	0.79	1.21	1.01	1.10	0.78	0.67	0.74

The introduction of the multi-path convolutional layer can enhance the generalization ability of the model, so that the model can learn more information, which is consistent with the conclusion obtained in [18].

5 Conclusion

In this paper, a polyphonic SED method based on capsule routing is proposed to address the problem of sound event detection with overlapping events. The multi-path convolution layer is used to extract different resolution features, which is helpful to enhance the generalization ability of the model. Among them, using a convolution kernel with a larger convolution size in the time axis can lead to better performance. By sharing the weights in the capsule layer, the parameters of the model are reduced, and the training speed is improved. The introduction of the recurrent layer enhances the temporal localization ability of the model. Experimental results show that through the dynamic routing algorithm, the model can extract the most representative features of sound events, thus improving the accuracy of polyphonic SED. Our next research directions include finding more effective feature extraction methods and studying the recently proposed expectation–maximization algorithm [33] to further improve the ability of vector capsules in extracting complex features. We are also interested in self-supervised sound event detection and the use of capsule networks for other low-class discrimination data.

Abbreviations

CNN: Convolutional neural networks; SED: Sound event detection; GMM: Gaussian mixture model; HMM: Hidden Markov model; SVM: Support vector machine; NMF: Non-negative matrix factorization; CRNN: Convolutional recurrent neural networks; MIL: Multiple instance learning; GLUs: Gated linear units; BLSTM: Bidirectional long short-term memory; CTC: Connectionist temporal classification; GRU: Bidirectional gated recurrent unit; FLOPs: Floating-point operations

Acknowledgements

Not applicable

Authors' contributions

H. Li conceptualizes the research and conducts experiments. S. Yang and W. Wang supervised the work and edited the manuscript. All authors read and approved the final manuscript.

Funding

Not applicable

Availability of data and materials

The datasets supporting the conclusions of this article are available on the internet. The DataSet of DCASE 2017 task 4 was obtained from DCASE Website, dcase.community/challenge2017/download.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1]College of Mathematics and Physics, Qingdao University of Science and Technology, No.99 Songling Road, Qingdao 266061, China. [2]Center for Vision Speech and Signal Processing, Department of Electrical and Electronic Engineering, Faculty of Engineering and Physical Sciences, University of Surrey, Guildford GU2 7XH, UK.

Received: 30 May 2021 Accepted: 14 February 2022
Published online: 07 March 2022

References

1. N. Cho, E.K. Kim, Enhanced voice activity detection using acoustic event detection and classification. IEEE Trans. Consum. Electron. **57**, 196 (2011).
2. N. C. Phuong and T. Do Dat, Sound classification for event detection: Application into medical telemonitoring, 2013 Int. Conf. Comput. Manag. Telecommun. ComManTel 2013 330 (2013).
3. T.K. Chan, C.S. Chin, Health stages diagnostics of underwater thruster using sound features with imbalanced dataset. Neural Comput. Appl. **31**, 5767 (2019).
4. Z. Zhao, S. Zhang, Z. Xu, K. Bellisario, N. Dai, H. Omrani, B.C. Pijanowski, Automated bird acoustic event detection and robust species classification. Ecol. Inform. **39**, 99 (2017).
5. E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, Convolutional recurrent neural networks for polyphonic sound event detection. IEEE/ACM Trans. Audio, Speech and Lang. Proc. **25**, 1291 (2017).
6. G. Parascandolo, H. Huttunen, T. Virtanen, Recurrent neural networks for polyphonic sound event detection in real life recordings. Paper presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6440–6444, 2016.
7. T. Heittola, A. Mesaros, A. Eronen, T. Virtanen, Audio context recognition using audio event histograms. Paper presented at the 18th European Signal Processing Conference, pp. 1272–1276, (2010).
8. N. Degara, M.E.P. Davies, A. Pena, M.D. Plumbley, Onset event decoding exploiting the rhythmic structure of polyphonic music. IEEE J. Sel. Top. Signal Process. **5**, 1228 (2011).
9. L. Lu, H.-J. Zhang, S.Z. Li, Content-based audio classification and segmentation by using support vector machines. Multimed. Syst. **8**, 482 (2003).
10. J.J. Carabias-Orti, T. Virtanen, P. Vera-Candeas, N. Ruiz-Reyes, F.J. Canadas-Quesada, Musical instrument sound multi-excitation model for non-negative spectrogram factorization. IEEE J. Sel. Top. Signal Process. **5**, 1144 (2011).
11. K.J. Piczak, Environmental sound classification with convolutional neural networks. Paper presented at the IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6, (2015).
12. S. Sabour, N. Frosst, G.E. Hinton, *Dynamic routing between capsules*. Paper presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 3859–3869, (2017).
13. A. Mobiny, H. Van Nguyen, Fast CapsNet for lung cancer screening. Lect. Notes Comput. Sci. **11071 LNCS**, 741 (2018).

14. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks. Commun. ACM **60**, 84 (2012).
15. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, (2016).
16. T. Iqbal, Y. Xu, Q. Kong, W. Wang, Capsule routing for sound event detection. Paper presented at the 26th European Signal Processing Conference (EUSIPCO), pp. 2255–2259, (2018).
17. Y. Liu, J. Tang, Y. Song, L. Dai, A capsule based approach for polyphonic sound event detection. Paper presented at the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 1853–1857, (2018).
18. K.-W. Liang, Y.-H. Tseng, P.-C. Chang, Parallel capsule neural networks for sound event detection. Paper presented at the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 1933–1936, (2019).
19. J.F. Gemmeke, D.P.W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R.C. Moore, M. Plakal, M. Ritter, Audio set: An ontology and human-labeled dataset for audio events, Paper presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 776–780, (2017).
20. J. Amores, Multiple instance classification: Review, taxonomy and comparative study. Artif. Intell. **201**, 81 (2013).
21. Y. Xu, Q. Kong, W. Wang, M.D. Plumbley, Large-scale weakly supervised audio classification using gated convolutional neural network. Paper presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 121–125, (2018).
22. S. Lee, M. Kim, Y. Jeong, End-to-end deep convolutional neural network with multi-scale structure for weakly labeled sound event detection (2019).
23. B. McFee, J. Salamon, J.P. Bello, Adaptive pooling operators for weakly labeled sound event detection. IEEE/ACM Trans. Audio, Speech, Lang. Process. **26**, 2180 (2018).
24. I. Martín-Morató, M. Cobos, F.J. Ferri, Adaptive distance-based pooling in convolutional neural networks for audio event classification. IEEE/ACM Trans. Audio, Speech, Lang. Process. **28**, 1925 (2020).
25. S. Liu, F. Yang, Y. Cao, and J. Yang, Frequency-dependent auto-pooling function for weakly supervised sound event detection, EURASIP J. Audio, Speech, Music Process. **2021**, 19 (2021).
26. R. Huang, J. Li, S. Wang, G. Li, W. Li, A Robust weight-shared capsule network for intelligent machinery fault diagnosis. IEEE Trans. Ind. Informatics **16**, 6466 (2020).
27. A. Mesaros, T. Heittola, A. Diment, B.M. Elizalde, A. Shah, E. Vincent, B. Raj, 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), pp. 85–92, (2017).
28. Y.-C. Wu, P.-C. Chang, C.-Y. Wang, J.-C. Wang, Asymmetrie kernel convolutional neural network for acoustic scenes classification (2017). Paper presented at the IEEE International Symposium on Consumer Electronics (ISCE), pp. 11–12, (2017).
29. Y. Xu, Q. Kong, Q. Huang, W. Wang, M.D. Plumbley, Convolutional gated recurrent neural network incorporating spatial features for audio tagging. Paper presented at the International Joint Conference on Neural Networks (IJCNN), pp. 3461–3466, (2017).
30. A. Mesaros, T. Heittola, T. Virtanen, Metrics for polyphonic sound event detection. Appl. Sci. **6** (2016).
31. S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML), Lille, France, pp. 448–456, (2015).
32. P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference. Paper presented at the 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, pp. 1–17, (2017).
33. G. E. Hinton, S. Sabour, and N. Frosst, Matrix capsules with EM routing, in ICLR (Poster) (OpenReview.net, 2018).

Publisher's Note