Sound Event Detection of Weakly Labelled Data With CNN-Transformer and Automatic Threshold Optimization

Qiuqiang Kong ⬡, *Student Member, IEEE*, Yong Xu, *Member, IEEE*, Wenwu Wang ⬡, *Senior Member, IEEE*, and Mark D. Plumbley ⬡, *Fellow, IEEE*

Abstract—Sound event detection (SED) is a task to detect sound events in an audio recording. One challenge of the SED task is that many datasets such as the Detection and Classification of Acoustic Scenes and Events (DCASE) datasets are weakly labelled. That is, there are only audio tags for each audio clip without the onset and offset times of sound events. We compare segment-wise and clip-wise training for SED that is lacking in previous works. We propose a convolutional neural network transformer (CNN-Transformer) for audio tagging and SED, and show that CNN-Transformer performs similarly to a convolutional recurrent neural network (CRNN). Another challenge of SED is that thresholds are required for detecting sound events. Previous works set thresholds empirically, and are not an optimal approaches. To solve this problem, we propose an automatic threshold optimization method. The first stage is to optimize the system with respect to metrics that do not depend on thresholds, such as mean average precision (mAP). The second stage is to optimize the thresholds with respect to metrics that depends on those thresholds. Our proposed automatic threshold optimization system achieves a state-of-the-art audio tagging F1 of 0.646, outperforming that without threshold optimization of 0.629, and a sound event detection F1 of 0.584, outperforming that without threshold optimization of 0.564.

Index Terms—Sound event detection (SED), weakly labelled data, automatic threshold optimization.

I. INTRODUCTION

SOUND event detection (SED) is an important research topic which can be used in smart home, self-driving cars and smart cities. For example, a SED system can detect an ambulance siren even if the ambulance is far away. In this situation, it is

difficult to detect the siren with cameras because of the distance and obstructions. Different from audio tagging (AT) which only requires to detect the presence or absence of sound events in an audio recording, SED requires to predict the onsets and offsets of sound events. SED has attracted many researches since the introduction of the Detection and Classification of Acoustic Scenes and Events (DCASE) challenges [1]–[5].

One challenge of the SED task is that audio recordings are usually weakly labelled. That is, in the training data, we only know the presence or absence of sound events, without knowing their onset and offset times. We call this kind of data *weakly labelled data (WLD)*. In this paper, we focus on the large-scale weakly supervised sound event detection task for smart cars dataset from the DCASE 2017 challenge Task 4 [6]. The audio recordings from this task is a subset of the AudioSet dataset [7]. This task includes both AT and SED. All audio clips for training are weakly labelled without time information of sound events. In previous research of AT, several CNN-based methods have been applied [8]–[19]. Those approaches show that a robust feature extractor is important for AT and SED. Usually CNNs are applied to the log mel spectrogram of audio recordings followed by a sigmoid non-linearity to predict the presence probabilities of sound events.

General SED tasks can be divided into two categories according to the availability of frame-level or clip-level labels: strongly supervised SED, when frame-level labels are provided; or weakly supervised SED, when only clip-level tags are provided. Several deep learning based methods [20]–[27] have been proposed for the strongly supervised SED task. However, frame-level sound event labels are time consuming to obtain. Recently, the DCASE 2017 Task 4 provides a large-scale dataset designed for AT and SED with only weakly labelled data provided. To train with weakly labelled data, segment-wise based methods [10], [28] split audio clips into segments, and assign each segment with weak labels. On the other hand, clip-wise training methods [12] apply entire audio clips for training. There is a lack of research comparing the segment-wise and clip-wise based methods for SED.

Although previous CNNs based methods have been successful in audio tagging and SED tasks, CNNs do not capture the long time dependency in an audio clip well. For example, the receptive field of a CNN can be limited to a short duration with a fixed length that does not take long history information into account in

the system. To solve this problem, convolutional recurrent neural networks (CRNNs) including [18], [29], [30] and bidirectional long short term memory (BLSTM) systems [22] were used to consider the long temporal information for audio tagging and sound event detection. One disadvantage of CRNNs is that the hidden states of a CRNN have to be calculated one by one, and can not be calculated in parallel. Recently, transformers [31]–[33] have been proposed to consider the long time dependency of sequences. A transformer consists of several attention layers. Each state of a layer takes the information from all states of the previous layer. Therefore, each state retains the global information of the input sequence.

Another challenging problem of AT and SED is the selection of thresholds for post-processing [27], [34]. For example, in the AT subtask, if the predicted probability of a sound class is over a threshold in an audio clip, then the audio clip is regarded as containing this sound class. The thresholds selection is an important part of AT and SED. Usually, the thresholds are selected empirically. For example, in the winning system of the AT subtask in the DCASE 2017 [12], thresholds of 0.3 are used for all the sound classes. However, those thresholds are selected by experience and may not be optimal. In this work, we propose an automatic threshold optimization method to solve this problem.

This work contributes in the following aspects. First, we investigate segment-wise training and clip-wise training for AT and SED. We found that different systems perform differently for the AT and SED subtask. Second, we propose a CNN-Transformer system, and achieves competitive results to the CNN-GRU system. Third, we propose an automatic threshold optimization method for the AT and SED subtask. Our proposed systems outperform the best systems in the DCASE 2017 Task 4 challenge. This paper is organized as follows: Section II introduces CNN and CRNN for AT and SED. Section III introduces a CNN-Transformer system. Section IV introduces segment-wise and clip-wise training. Section V proposes an automatic threshold optimization method for AT and SED. Section VI shows experimental results. Section VII concludes this work.

II. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

A. Conventional CNNs

CNNs were originally designed for image classification [35], and have been recently used for audio related tasks such as speech recognition [36] and AT [37], [38]. A conventional CNN consists of convolution layers, pooling layers and fully connected layers. The input to each convolutional layer is a tensor with a shape (N, C, W, H) representing the number of input samples, channels, width and height. For AT, the input width and height represent the number of time frames and frequency bins. Each convolutional layer consists of a set of learnable kernels. The output of a convolutional layer is a tensor called feature maps. The kernels in a convolutional layer can learn local time-frequency patterns in the spectrogram of an audio clip. In audio processing, low level features [39] can be waveforms or time-frequency representations such as spectrogram. High level features are those extracted from low level features by convolutional layers. Recent CNN architectures

apply batch normalization [40] after convolutional layers to speed up and stabilise training. Nonlinear activation functions such as ReLU [41] are applied after each batch normalization. For AT and SED, pooling layers are applied along both time and frequency axes. A time distributed fully connected layer is applied on the output of the last convolutional layer to predict the presence probability of sound events along the time axis. Then the predicted probabilities are aggregated over the time axis to obtain the clip-wise sound event presence probability. The aggregation can be, for example, maximum or average operations over the time axis.

B. Convolutional Recurrent Neural Network (CRNNs)

The receptive field of CNNs have limited sizes. That is, CNNs can not capture long time dependency in an audio clip. However, some sound events have long time dependencies. For example, an ambulance siren may last for tens of seconds, and the temporal information is useful for AT and SED. Designing a system that is able to capture the temporal dependency is beneficial for AT and SED. Recurrent neural networks (RNNs) [42] are kinds of neural networks that can store history information in their hidden states, and thus capture long term dependency of sequential data. RNNs have been applied to language processing tasks such as [42]. The potential problem of a conventional RNN is that the gradient of weights may vanish or explode in training. Long short term memory (LSTM) [43] is a variation of RNN that introduces constant error carousel units, input gate, output gate and forget gate to avoid the gradient exploding and vanishing problem. An improved architecture of LSTM called gated recurrent units (GRU) [44] is proposed to reduce the parameters of LSTMs and simplify the gates to a reset gate and a forget gate. A GRU can be in both directions which we call bidirectional GRU (biGRU), which is applied in our AT and SED systems.

III. CNN-TRANSFORMER

CRNN can capture long time-dependency of sound events. On the other hand, the sequential nature of CRNNs also makes it more difficult to take advantage of modern fast computing devices such as GPUs. Recently, transformer [31] is proposed to learn correlations of time steps in a sequence such as natural language processing tasks [32]. Compared with RNNs which require to compute the hidden states in a sequence, a transformer can parallelize the computation which only requires matrix multiplication in the forward pass. Transformer applies a self-attention mechanism which directly models relationships between all time steps in a sequence. In an audio clip, a sound class may contain several sound events over time. For example, the speech of a human may appear in any time in an audio clip. A transformer can capture the correlation of speeches appearing in different part of an audio clip.

A. Transformer

Transformer was originally proposed in [31]. The motivation for the design of the transformer is to allow modeling of dependencies without regard to their distance in the input sequence. In addition, a transformer allows for more parallel computing than

RNNs by removing the recurrent connections. A transformer consists of several encoder and decoder layers. The encoder transforms an input to a high level embedding, and the decoder transforms an embedding to output. In a classification task such as AT or SED, we only need the encoder. Each encoder consists of several encoder layers. For each encoder layer, we denote the input to the encoder layer as a tensor x with a shape of $T \times C$, where T and C represent the number of time steps and channels. We follow the symbols used in [31]. An encoder layer consists of a query transform matrix W^Q, a key transform matrix W^K and a value transform matrix W^V. The matrices W^Q and W^K have a shape of $C \times d_k$, and W^V has a shape of $C \times d_v$ where d_k and d_v are integers. Then the query Q, key K and value V can be obtained by:

$$
\begin{aligned}
Q &= xW^Q \\
K &= xW^K \\
V &= xW^V.
\end{aligned}
\tag{1}
$$

The query Q and key K have a shape of $T \times d_k$, and the value V has a shape of $T \times d_v$. The output of an encoder layer can be written as:

$$
h = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,
\tag{2}
$$

where the output h has a shape of $T \times H$. Equation (2) computes the dot product of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values V [31]. The division of square root of d_k is a normalization term [31]. In (2), the inner product of Q and K^T has a shape of $T \times T$, representing the feature correlation of different time steps. The softmax operation converts the correlation value to probabilities along the time steps indicating how much the value V in a time step should be attended.

B. CNN-Transformer

For audio tagging and SED, the input is usually a time-frequency representation such as a log mel spectrogram. Log mel spectrogram is a low level feature and CNNs have been proposed to apply on the log mel spectrogram to extract high level features [37]. To build the CNN-Transformer system, we first apply a CNN described in Section II on the log mel spectrogram of an audio clip. Convolutional layers in the CNN are used to extract high level features of the input log mel spectrogram. We use the feature maps of the last convolutional layer to obtain embedding vectors along time axis. The embedding can be viewed as x with a shape of the number of time frames by the number of channels. The output of the transformer has a shape of $T \times d_v$. A fully connected layer followed by a sigmoid non-linearity is applied on this output to predict the presence probabilities of sound classes over time steps. An aggregation function such as average aggregation can be applied to average out those probabilities along time domain to obtain the audio tagging result.

IV. SEGMENT-WISE V.S. CLIP-WISE SED

Sections II and III introduce CNN, CNN-biGRU and CNN-Transformer architectures. In this section, we introduce how we apply the aforementioned architectures for AT and SED training with weakly labelled data. Conventional SED methods utilise strongly labelled data for supervised learning. However, collecting strongly labelled data is time consuming. The amount of strongly labelled data is therefore limited. To solve this problem, we propose to use weakly labelled dataset for SED. The SED systems with weakly labelled data can be categorized into segment-wise training [10], [16], and our previously proposed clip-wise training [12] methods. This section aims to investigate the comparison of the segment-wise and clip-wise training for AT and SED.

A. Segment-Wise Training

We denote the waveform of an audio clip as X. For an X lasting for several seconds, we split it into several segments $\{x_m\}_{m=1}^M$ where M is the number of segments. Each segment inherits the tags of the audio clip. We denote the tags of each segment as $y \in \{0,1\}^K$ where K is the number of sound classes. The SED problem is converted to an audio tagging problem on those segments. In training, a classifier f is trained on the segments. The loss function can be written as:

$$
E = -\sum_{m=1}^M \sum_{k=1}^K [y_k \log f(x_m)_k + (1-y_k)\log(1 - f(x_m)_k)].
\tag{3}
$$

In inference, an audio clip is split into segments $\{x_m\}_{m=1}^M$, and the SED result on each segment can be calculated by $f(x_m)$. The AT result can be obtained by aggregating $f(x_m)$ over all segments:

$$
F(X) = \text{agg}(\{f(x_m)\}_{m=1}^M).
\tag{4}
$$

The aggregation can be, for example, maximum or average operation over all the segments. The segment-wise classifier f can be CNN, CNN-biGRU or CNN-Transformer followed by a sigmoid non-linearity to predict the presence probability of sound events of each segment. We investigate the performance of choosing different duration of segments on SED and AT in Section VI.

B. Clip-Wise Training

In the segment-wise training, all segments x_m inherit the tags of an audio clip X. The problem of segment-wise training is that many sound events may only last for a short time in the audio clip. Therefore, the tags of x_m may be incorrect because the segment may not contain the sound event. To solve this problem, our previous work proposed attention neural network based clip-wise training [12]. The clip-wise training method does not explicitly assign tags for each segment x_m. Instead, the systems are designed to learn the tags of x_m implicitly, that is, from the hidden layer of a neural network. We denote the segment-wise prediction of a segment x_m to be $f(x_m)$. Then the prediction on the clip X can be obtained by aggregating the segment-wise predictions. For example, the aggregation can be a max, average or attention function over the prediction of all segments of each sound class. The max function can be defined

---

Algorithm 1: Audio Tagging.

1:   Inputs: predicted presence probability of sound events in an audio clip $F(X)$. AT thresholds $\{\mu_1, \ldots, \mu_K\}$.

2:   Outputs: Predicted audio tags.

3:   **for** $k = 1, \ldots, K$ **do**

4:      **if** $F(X)_k < \mu_k$ **then**

5:         return 0 for the k-th sound event.

6:      **else**

7:         return 1 for the k-th sound event.

---

as:

$$F(X)_k = \max_k f(x_m)_k. \tag{5}$$

For example, the average function can be defined as:

$$F(X)_k = \sum_{m=1}^{M} f(x_m)_k. \tag{6}$$

The decision-level function can be defined as [45]:

$$F(X)_k = \sum_{m=1}^{M} f(x_m)_k p(x_m)_k, \tag{7}$$

where $p(x_m) = \frac{\exp(w(x_m)_k)}{\sum_{j=1}^{M} \exp(w(x_j)_k)}$, and $w(\cdot)$ is a linear transformation. In training, we calculate the categorical binary crossentropy loss between the clip-level prediction $F(X)$ and the ground truth label of X:

$$E = -\sum_{k=1}^{K} [y_k \log F(X)_k + (1 - y_k) \log(1 - F(X)_k)]. \tag{8}$$

The difference between the clip-wise training (8) and the segment-wise training (3) is that the clip-wise training directly outputs $F(X)$, and can be trained in an end-to-end way with weakly labelled data. The $f(x_m)_k$ are latent representations learnt by the neural network.

V. Automatic Threshold Optimization

To obtain the presence or absence of sound events in an audio clip, AT systems need to apply thresholds to the system outputs. A sound class is predicted as presence if the AT output is larger than its corresponding threshold. The thresholds for AT are denoted as $\Theta^{\text{AT}} = \{\mu_1, \ldots, \mu_K\}$. Algorithm 1 shows the algorithm to obtain AT result from the AT system outputs.

SED requires to predict not only the presence or absence but also the onset and offset times of sound events. Similar to AT, we first apply thresholds $\{\mu_1, \ldots, \mu_K\}$ on $F(X)$ to predict the presence or absence of K classes of sound events in an audio clip X. If the k-th sound class is predicted to be present, then we apply a threshold τ_k^{high} to the segment-wise predictions $f(x_m)$ to detect sound events. In addition, to reduce the number of missed detection, a second threshold τ_k^{low} is used. To begin with, we denote the neighbouring segments of an active segment as x'. Then, a lower threshold τ_k^{low} is applied on $f(x_m)$ to obtain the calibrated sound event detection result. All thresholds for SED are denoted

---

Algorithm 2: Sound Event Detection.

1:   Inputs: clip-wise prediction $F(X)$, segment-wise prediction $f(x_m)$, AT thresholds $\{\mu_1, \ldots, \mu_K\}$, SED high thresholds $\{\tau_1^{\text{high}}, \ldots, \tau_K^{\text{high}}\}$, SED low thresholds $\{\tau_1^{\text{low}}, \ldots, \tau_K^{\text{low}}\}$.

2:   Outputs: Detected sound events.

3:   **for** $k = 1, \ldots, K$ **do**

4:      **if** $F(X)_k < \mu_k$ **then**

5:         return 0 for the k-th sound event.

6:      **else**

7:         **for** $m = 1, \ldots, M$ **do**

8:           **if** $f(x_m)_k > \tau_k^{\text{high}}$ **then**

9:             Return 1 for the neighbouring segments x' of x_m if $f(x')_k < \tau_k^{\text{low}}$.

---

as $\Theta^{\text{SED}} = \{\mu_1, \ldots, \mu_K, \tau_1^{\text{high}}, \ldots, \tau_K^{\text{high}}, \tau_1^{\text{low}}, \ldots, \tau_K^{\text{low}}\}$. Algorithm 2 summarizes obtaining the SED results from the clip-wise and segment-wise predictions.

The winning system of the AT subtask in DCASE 2017 Task 4 [12] applies constant thresholds for both the AT and SED subtask. Setting those thresholds requires a lot of experience, and the manually selected thresholds are often not optimal. In addition, each sound class may have different thresholds. Therefore, sweeping over all combinations of thresholds is intractable. We propose an automatic threshold optimization method to solve this problem. In the first stage, we optimize the systems and evaluate the systems based on the metrics that do not depend on the thresholds such as mean average precision (mAP). In the second stage, for a trained system, we optimize the thresholds over a specific metric such as F1 score or error rate (ER) to optimize the thresholds.

For an audio clip X, the AT result r^{AT} can be obtained by $\text{alg}^{\text{AT}}(F(X), \Theta^{\text{AT}})$ where alg^{AT} is the AT algorithm shown in Algorithm 1. The SED result r^{SED} can be obtained by $\text{alg}^{\text{SED}}(F(X), \{f(x_m)\}_{m=1}^{M}, \Theta^{\text{SED}})$ where alg^{SED} is the SED algorithm shown in Algorithm 2. The goal of AT or SED is to minimize some loss $J(\Theta)$, for example, ER $J_{\text{ER}}(\Theta)$ or negative F1 score $J_{\text{F1}}(\Theta)$. The reason of using negative F1 is that minimizing $J_{\text{F1}}(\Theta)$ is equivalent to maximizing F1 score. The optimization of thresholds becomes solving the following problem:

$$\hat{\Theta} = \underset{\Theta}{\text{argmin}} J(\Theta). \tag{9}$$

The difficulty of solving (9) is that Θ consists of several parameters to be optimized. So applying grid search over all thresholds is inefficient. Another way is to use gradient based methods to iteratively optimize those thresholds. However, equation (9) is a non-differentiable function, so we can not calculate the gradient over the thresholds in an analytical way. This is because both the AT and SED algorithms in Algorithm 1 and Algorithm 2 contain non-differentiable operations such as thresholding. In addition, the evaluation metrics ER and F1 score are also non-differentiable. To solve this problem, we propose

Algorithm 3: Adam Optimization. Symbol g_t^2 Indicates the Elementwise Square $g_t \odot g_t$. Learning Rate is Denoted as α. Hyper-Parameters are Set to $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ Following [46].

1:   Inputs: parameters Θ.
2:   Init $\Theta_0, m_0 = 0, v_0 = 0, t = 0$
3:   **while** Θ not converged **do**
4:     $t \leftarrow t + 1$
5:     $g_t = \bigtriangledown_\Theta J$
6:     $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$
7:     $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$
8:     $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
9:     $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
10:   $\Theta_t \leftarrow \Theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

Algorithm 4: Automatic Thresholds Optimization.

1:   Inputs: Validation dataset $D = \{X^{(n)}, y^{(n)}\}_{n=1}^N$, trained AT system $F(\cdot)$, trained SED system $f(\cdot)$.
2:   Outputs: Optimized thresholds Θ.
3:   Initialize Θ.
4:   **for** $i = 1, \ldots, \text{ITER}$ **do**
5:     **for** $n = 1, \ldots, N$ **do**
6:       $\hat{y}^{(n)} = \text{alg}(F(X^{(n)}), f(x_m^{(n)}), \Theta)$.
7:       $J = \text{metric}(\{\hat{y}^{(n)}\}_{n=1}^{n=N}, \{y^{(n)}\}_{n=1}^{n=N})$
8:     **for** θ in Θ **do**
9:       $\bigtriangledown_\theta J = \frac{J(\Theta + \triangle\Theta) - J(\theta)}{\triangle\theta}$
10:     $\bigtriangledown_\Theta J = \{\bigtriangledown_\theta J\}_{\theta \in \Theta}$
11:     $\Theta \leftarrow \text{opt}(\Theta, \bigtriangledown_\Theta J)$

---

to calculate the gradients over the thresholds in a numerical way. That is, for each parameter θ, we calculate the gradient as:

$$\bigtriangledown_\theta J(\Theta) = \frac{J(\Theta + \triangle\Theta) - J(\Theta)}{\triangle\theta}, \qquad (10)$$

where $\triangle\theta$ is a small constant number, and $\triangle\Theta$ is a vector with all zero values the position of θ which has a value of $\triangle\theta$. After calculating the numerical gradient for all parameters $\bigtriangledown_\Theta J = \{\bigtriangledown_\theta J\}_{\theta \in \Theta}$, the optimized thresholds can be obtained by applying gradient based optimization methods iteratively: $\Theta \leftarrow \text{opt}(\Theta, \bigtriangledown_\Theta J)$, where opt denotes an optimization algorithm such as gradient descent (GD). GD optimization can be written by $\Theta \leftarrow \Theta - \alpha \bigtriangledown_\Theta J$ where α is a learning rate. We use Adam optimizer [46] to optimize $J(\Theta)$ due to its fast convergence. We describe Adam optimizer in Algorithm 3 to show how it is used in our method. Overall, the automatic threshold optimization algorithm is described in Algorithm 4. We have released our proposed automatic threshold optimization toolbox called *autoth*.[1]

TABLE I
SOUND EVENTS IN THE DCASE 2017 TASK 4 "LARGE-SCALE WEAKLY SUPERVISED SOUND EVENT DETECTION FOR SMART CARS"

	Event name	Training number
Warning sounds	Train horn	441
	Air horn, truck horn	407
	Car alarm	273
	Reversing beeps	337
	Ambulance (siren)	624
	Police car (siren)	2399
	Fire engine, fire truck (siren)	2399
	Civil defense siren	1506
	Screaming	744
Vehicle sounds	Bicycle	2020
	Skateboard	1617
	Car	25744
	Car passing by	3724
	Bus	3745
	Truck	7090
	Motorcycle	3291
	Train	2301

VI. EXPERIMENTS

A. Experimental Setup

There are several SED datasets including the DCASE 2017 Task 4 [6], the DCASE 2018 Task 4 [47] and the DCASE 2019 Task 4 [48]. We evaluate our SED system on the DCASE 2017 Task 4 "large-scale weakly supervised sound event detection for smart cars". The reason we choose this dataset is because it is a large-scale dataset containing over 140 hours of weakly labelled audio clips for training. The audio recordings of the DCASE 2017 Task 4 are from a subset of AudioSet [7] where each audio clip is extracted from YouTube video. DCASE 2017 Task 4 consists of 17 sound events divided into two categories: "warning" and "vehicle". Most of those audio clips have duration of 10 seconds. The audio clips shorter than 10 seconds are padded with silence to 10 seconds. Table I lists the sound events and their statistics. The DCASE 2017 Task 4 dataset consists of a training subset with 51172 audio clips, a validation subset with 488 audio clips, and an evaluation set with 1103 audio clips. The training subset is weakly labelled. The validation and evaluation subsets are both weakly and strongly labelled for evaluation. The source code of this work is released.[2]

B. Feature

We use log mel spectrogram as input feature following previous work on audio tagging [10], [37], [49]. To begin with, all audio clips are converted to monophonic and resampled to 32 kHz. The short time Fourier transform with a Hanning window of 1024 samples and a hop size of 320 samples is used to extract spectrogram which leads to 100 frames in a second. We apply 64 mel filter banks on the spectrogram followed by logarithmic operation to calculate log mel spectrogram. The number 64 is chosen so that it can be evenly divided by a power of 2 in the down-sampling layers of CNNs. The mel filter banks

---

[1][Online]. Available: https://github.com/qiuqiangkong/autoth

[2][Online]. Available: https://github.com/qiuqiangkong/sound_event_detection_dcase2017_task4

TABLE II
CNN ARCHITECTURE

Layers	Output size	Param. num.
Input: log mel spectrogram	bs $\times 1 \times 640 \times 64$	-
$\begin{pmatrix} 3 \times 3 \text{ @ } 64 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	bs $\times 64 \times 640 \times 64$	37,696
2×2 avg. pooling	bs $\times 64 \times 320 \times 32$	-
$\begin{pmatrix} 3 \times 3 \text{ @ } 128 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	bs $\times 128 \times 320 \times 32$	221,696
2×2 avg. pooling	bs $\times 128 \times 160 \times 16$	-
$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	bs $\times 256 \times 160 \times 16$	885,760
2×2 avg. pooling	bs $\times 256 \times 80 \times 8$	-
$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	bs $\times 512 \times 80 \times 8$	3,540,992
Embedding: Avg. out freq. bins	bs $\times 512 \times 80 \times 1$	-

have a lower cut-off frequency of 50 Hz and a higher cut-off frequency of 14 kHz to avoid aliasing caused by resampling.

C. Model

The segment-wise training systems are described in equation (4), and are modeled by a 9-layer CNN which has shown to perform well on a variety of audio tagging tasks [49]. Table II shows that the 9-layer CNN consists of 4 convolutional blocks, where each convolutional block consists of 2 convolutional layers with kernel sizes of 3×3. Batch normalization [40] and ReLU non-linearity [50] is applied after each convolutional layer. The convolutional block consists of 64, 128, 256 and 512 feature maps, respectively. A 2×2 average pooling is applied after each convolutional block to extract high level features. We did not apply residual connections in our CNNs as gradient vanishing is not a problem with 8 convolutional layers. In Table II, the number following @ represents the number of feature maps. The second column shows the number of batch size (bs), feature maps, frames and frequency bins. We average out the frequency axis of the output from the last convolutional layer. Then time distributed fully connected layer with sigmoid non-linearity is applied to predict the presence probability of sound events of each time frame. To obtain the AT result for supervised learning, aggregation functions including max, average and attention along time frames are applied. Adam [46] optimizer with a learning rate of 0.001 is applied, and is reduced to 0.0001 after the performance is plateaued on validation data. Mixup [51] with alpha of 1.0 is used in all experiments to prevent training from overfitting. The training is stopped at 60,000 iterations by observing the performance on the validation set.

D. Evaluation Metrics

To evaluate the systems performance, we use the precision, recall and F1 score which are described in [52]:

$$P = \frac{TP}{TP + FP} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = \frac{2P \cdot R}{P + R}, \tag{13}$$

where TP, FP, FN are the number of true positive, false positive and false negative samples, respectively. The higher precision, recall and F1 score indicate better performance. Usually thresholds need to be manually selected and applied on the system outputs to calculate TP, FP and FN. We use average precision (AP) metric [7] to compare the performance of different systems because the AP does not depend on thresholds. AP is defined as the area under the precision-recall curve calculated at multiple thresholds. Mean average precision (mAP) is the averaged AP over all sound classes. The higher mAP indicates better performance. Random guess has an mAP of 0.06 for the DCASE 2017 Task 4 containing 17 sound classes. The mAP is a *macro-averaging* statistic because it is calculated independently within a sound class. Then, the statistics are averaged across all sound classes. On the other hand, *micro-averaging* statistic is calculated from outputs and ground truths flattened from all classes.

For the AT subtask, systems are ranked based on macro-averaging F1 score. For the SED subtask, systems are ranked based on micro-averaging F1 score and Error rate (ER) evaluated on 1-second segments [4]. Error rate measures the amount of errors in terms of insertions (I), deletions (D) and substitutions (S), and is defined as follows [52]:

$$ER = \frac{\sum_m S(m) + \sum_m D(m) + \sum_m I(m)}{\sum_m N(m)}, \tag{14}$$

where $I(m), D(m), S(m), N(m)$ are the number of inserted, deleted, substituted, and ground truth sound events in the m-th segment. Lower ER indicates better performance. The segment based evaluation is calculated in a fixed time grid, using segments of one second length to compare the ground truth and the system output [6]. Similarly, segment based F1-score are calculated in the same way.

E. Segment-Wise AT and SED

There is a lack of research comparing segment-wise [10] and clip-wise [12] training for AT and SED. We first investigate the segment-wise training method. To begin with, an audio clip is split into segments. Then SED predictions are calculated by running audio tagging system on segments. Because the audio clips are weakly labelled, there is no information when a sound event occurs and how long they last. This can affect the label accuracy of segment-wise training because all segments inherit the tags from the audio clip. In inference, the SED result is obtained by predicting audio tags on segments. Table III shows the average percentage of time frames in an audio clip containing different sound events from the validation set of DCASE 2017 Task 4. Sound events such as civil defense siren has a presence percentage of 0.930 which indicates segment-wise labels are more likely to be correct. Sound events such as train horn has a presence percentage of 0.400 which indicates segment-wise label is less likely to be correct.

The 10-second audio clips are split into segments with different lengths from 1 to 10 seconds. Each segment inherit the tags from the audio clip. The minimum 1-second setting follows [10].

TABLE III
THE PERCENTAGE OF TIME CONTAINING SOUND EVENTS IN AN AUDIO CLIP LABELLED AS CONTAINING THE SOUND EVENT

Train horn	Air horn, truck horn	Car alarm	Reversing beeps	Bicycle	Skateboard	Ambulance	Fire engine, fire truck	Civil defense siren
0.400	0.553	0.463	0.538	0.478	0.616	0.829	0.905	0.930

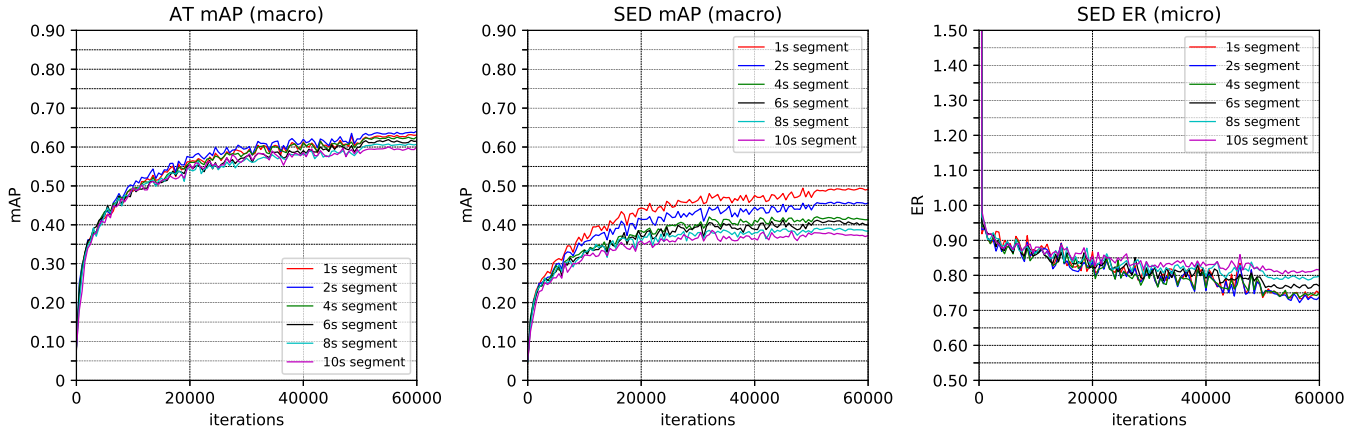| Police car	Screaming	Car	Car passing by	Bus	Truck	Motorcycle	Train	Avg
0.746	0.619	0.606	0.768	0.885	0.764	0.696	0.776	0.681

Fig. 1. Segment-wise training result trained with different durations of audio segments. From left to right: Audio tagging macro mAP; SED macro mAP; SED micro error rate.
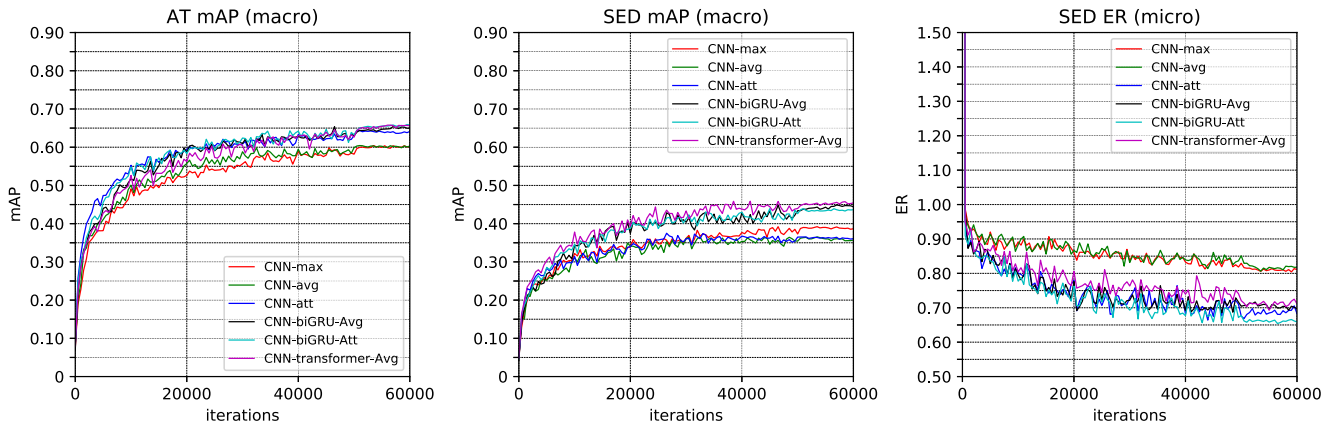

Fig. 2. Clip-wise training result with different systems. Audio tagging macro mAP; SED macro mAP; SED micro error rate.

A 9-layer CNN is applied to build the segment-wise training systems. Fig. 1 shows the mAP and ER of AT and SED with different segment durations. Training with 2-second segments achieves an mAP of 0.64 in audio tagging, slightly outperforming other segment duration in AT. This indicates that the prediction of long segments does not perform well when no attention or temporal dependency is used. The second column of Fig. 1 shows that training with 1-second segments achieves an SED mAP of 0.44, outperforming other segment duration. This indicates that shorter segments achieve better SED result than longer segments when using segment-wise training systems. One explanation is that SED is obtained by AT on segments, so AT on shorter segments can provide higher SED resolution. To calculate the ER, we use constant AT thresholds of $\mu_k = 0.5, k = 1, \ldots, K$,

and SED thresholds of $\tau_k^{\text{high}} = 0.3, \tau_k^{\text{low}} = 0.1, k = 1, \ldots, K$ for all sound class following [12]. The third column of Fig. 1 shows that the 1-second and 2-second segment duration achieve an ER of 0.74, outperforming other segment durations.

F. Clip-Wise AT and SED

We investigate the clip-wise training systems in this section. The difference of the clip-wise training and the segment-wise training is that with clip-wise training, the SED result can be obtained from the intermediate layer of a neural network. Then, the AT predictions can be calculated by the aggregation functions such as (5, 6, 7). Fig. 2 shows the AT and SED performance of the clip-wise CNN systems. For the AT subtask, the decision-level
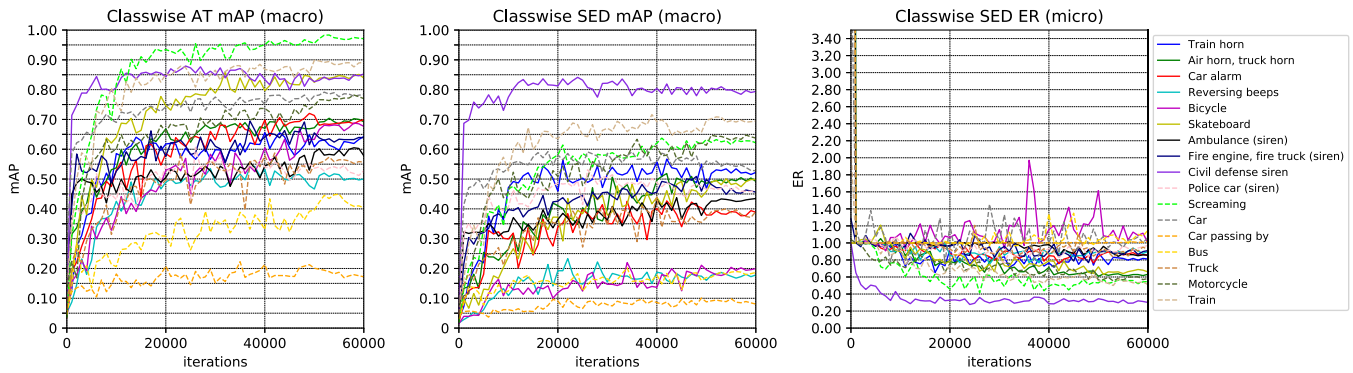
Fig. 3.   From left to right: Class-wise AT average precision; Class-wise SED average precision; Class-wise SED error rate with the CNN-biGRU-Att system.
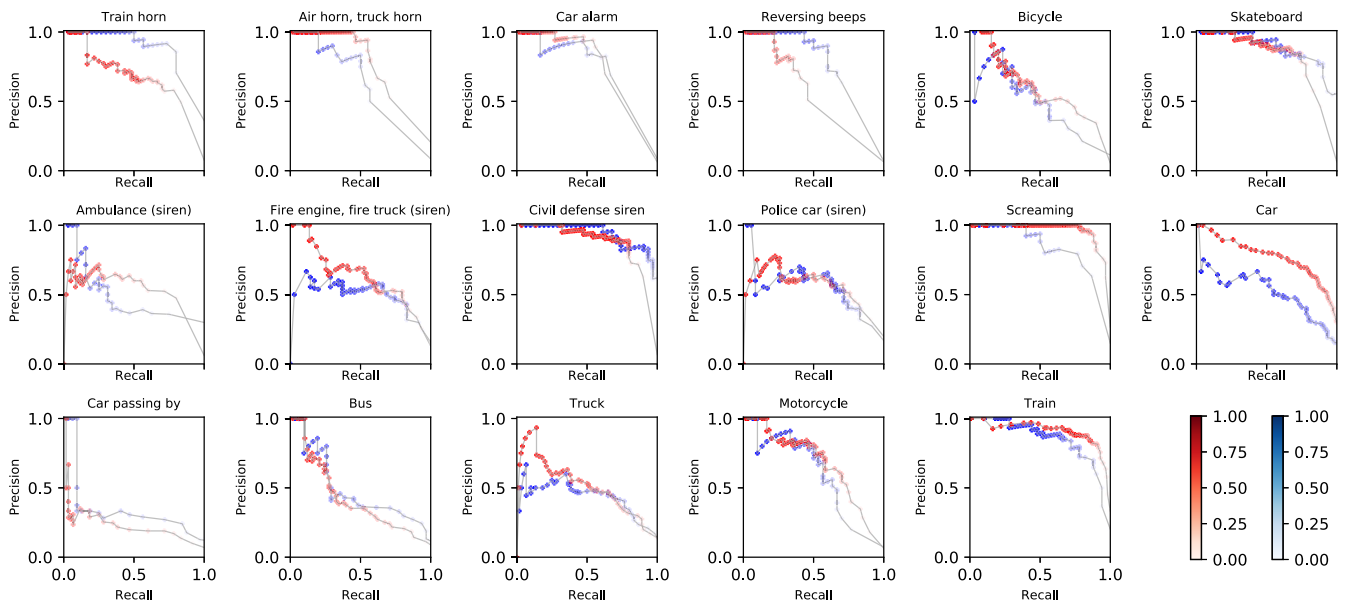

Fig. 4.   Precision-recall curves of sound classes at different thresholds with the CNN-biGRU-Att system.

maximum (CNN-Max) or decision-level average (CNN-Avg) systems achieve an mAP of 0.60. The decision-level attention (CNN-Att) improves the mAP to 0.64, indicating the attention plays an important role in AT. The CNN-biGRU systems and the CNN-Transformer system further improve the mAP performance to 0.65, indicating that the temporal dependency information is helpful for AT. For the SED subtask, the CNN-Transformer system achieves an SED mAP of 0.45, slightly outperforming the CNN-biGRU systems of 0.44 and other CNN systems of 0.36 to 0.39, respectively. On the other hand, CNN-biGRU achieves an ER of 0.66, outperforming other systems ranging from 0.69 to 0.86. To calculate ER we applies thresholds that are the same as the segment-wise training systems. Fig. 1 and Fig. 2 show that the segment-wise training achieves better mAP on the SED task, while the clip-wise training achieves better ER on the SED task. One explanation is that mAP is evaluated in frame-wise, while ER is evaluated in 1-second segments.

Fig. 3 shows the class-wise performance of the CNN-biGRU-Att system over training iterations. The performance on different sound classes varies. The prediction of screaming achieves the highest AT mAP of 0.94. On the other hand, the prediction of car passing by achieves the lowest mAP of 0.20. One explanation of the underperformance of sound classes such as car passing by is that they are difficult to perceive even by human in audio recordings. For SED, some sound classes achieve better mAP than others, for example, civil defense siren achieves the highest mAP of 0.80, indicating the system is performing well on these sound classes. The ER curve of different sound classes is different. Civil defense siren has an ER of 0.26 while other sound classes such as bicycle has ER over 1. The class-wise results show that both the AT and SED performance vary from sound class to sound class.

G. Automatic Thresholds Optimization

Previous subsection shows that the performance of different sound classes can be different. It can be useful to observe their precision-recall curves under various thresholds. Fig. 4 shows the AT precision-recall curve of sound classes with the CNN-biGRU-Att system under different thresholds ranging

TABLE IV
PERFORMANCE OF THE PROPOSED SYSTEMS ON THE VALIDATION (VAL.) AND EVALUATION (EVAL.) SUBSET

Systems	VAL. AT				VAL. SED			EVAL. AT				EVAL. SED		
	mAP	F1	P	R	mAP	F1	ER	mAP	F1	P	R	mAP	F1	ER
CNN-Max	0.579	0.479	0.497	0.462	0.359	0.352	0.83	0.602	0.565	0.593	0.539	0.387	0.466	0.75
CNN-Max (auto thres)	#	0.538	0.633	0.467	#	0.489	0.74	#	0.582	0.674	0.513	#	0.520	0.72
CNN-Avg	0.587	0.479	0.495	0.464	0.342	0.346	0.83	0.603	0.553	0.578	0.530	0.357	0.452	0.76
CNN-Avg (auto thres)	#	0.564	**0.649**	0.498	#	0.357	0.76	#	0.597	0.665	0.542	#	0.518	0.74
CNN-Att	0.637	0.559	0.557	0.561	0.339	0.480	0.82	0.641	0.605	0.607	0.604	0.361	0.552	0.80
CNN-Att (auto thres)	#	**0.604**	0.596	**0.612**	#	0.523	0.68	#	0.617	0.610	0.624	#	0.545	0.69
CNN-biGRU-Avg	0.650	0.555	0.541	0.569	**0.456**	0.491	0.90	0.650	0.632	0.638	0.627	0.444	0.564	0.84
CNN-biGRU-Avg (auto thres)	#	0.602	0.609	0.594	#	0.534	0.685	#	0.632	0.648	0.617	#	0.567	0.72
CNN-biGRU-Att	0.647	0.557	0.554	0.559	0.419	0.492	0.80	0.658	0.625	0.632	0.617	0.436	0.564	0.78
CNN-biGRU-Att (auto thres)	#	0.581	0.575	0.587	#	**0.537**	**0.65**	#	0.640	0.637	**0.642**	#	**0.584**	**0.68**
CNN-Transformer-Avg	**0.653**	0.557	0.554	0.561	0.437	0.483	0.91	**0.656**	0.629	0.644	0.616	**0.454**	0.556	0.87
CNN-Transformer-Avg (auto thres)	#	0.599	0.636	0.566	#	0.524	0.75	#	**0.646**	**0.691**	0.607	#	0.573	0.75

TABLE V
AT OF DIFFERENT SYSTEMS

Dev-set	F1	Precision	Recall
DCASE2017 Baseline [4]	0.109	0.078	0.175
MIL-NN ensemble [53]	0.353	0.286	0.460
CNN-ensemble [16]	0.570	**0.703**	0.479
Previously submitted fusion [12]	0.577	0.565	0.589
CNN-biGRU-Att (auto thres)	0.581	0.575	0.587
CNN-Transformer-Avg (auto thres)	**0.599**	0.636	0.566
Eval-set	**F1**	**Precision**	**Recall**
DCASE2017 Baseline [4]	0.182	0.150	0.231
MIL-NN [53]	0.352	0.316	0.397
CNN-ensemble [16]	0.526	**0.697**	0.423
Previously submitted fusion system [12]	0.556	0.614	0.508
CNN-biGRU-Att (auto thres)	0.640	0.637	**0.642**
CNN-Transformer-Avg (auto thres)	**0.646**	0.691	0.607

TABLE VI
SED RESULTS OF DIFFERENT SYSTEMS

Dev-set	F1	Error rate
DCASE2017 baseline [4]	0.138	1.02
CNN-ensemble [16]	0.471	**0.71**
MultiScale-CNN [28]	0.342	0.84
Previously submitted fusion [12]	**0.497**	0.72
CNN-biGRU-Att (auto thres)	**0.537**	**0.65**
CNN-Transformer-Avg (auto thres)	0.524	0.75
Eval-set	**F1**	**Error rate**
DCASE2017 baseline [4]	0.284	0.93
CNN-ensemble [16]	0.555	**0.66**
MultiScale-CNN [28]	0.471	0.75
Previously submitted fusion [12]	0.518	0.73
CNN-biGRU-Att (auto thres)	**0.584**	0.68
CNN-Transformer-Avg (auto thres)	0.573	0.75

from 0 and 1. The blue and red curve show the validation and evaluation precision-recall curve, respectively. Fig. 4 shows that the validation and the evaluation curve have similar trend but are not overlapped, indicating that the data distribution of validation and evaluation data can be slightly different. Some sound classes such as screaming have high precision at a variety of thresholds. On the other hand, the precision drops rapidly with the increase of recall for some sound classes such as car passing by. Fig. 4 shows that different sound classes have different thresholds to achieve optimal metrics such as F1 score.

Table IV shows the AT and SED performance with the clip-wise training systems. We first apply constant thresholds for both AT and SED systems. The constant thresholds are the same as the thresholds applied in previous sections. In addition to applying the constant thresholds, we apply thresholds Θ^{AT} and Θ^{SED} for AT and SED obtained by using the automatic thresholds optimization algorithm described in Algorithm 4. Table IV shows that the proposed automatic thresholds optimization improved both the AT and SED performance. For example, the CNN-Transformer AT F1 score improves from 0.557 to 0.599, and 0.629 to 0.646 in the validation and evaluation set, respectively. The CNN-biGRU SED ER is reduced from 0.80 to 0.65, and 0.78 to 0.68 in the validation and evaluation set, respectively. Those results show the effectiveness of the proposed automatic threshold optimization method.

Table V shows the precision, recall and F1-score of different methods for the AT on the validation and evaluation sets, respectively. The official DCASE2017 baseline is give in [4]

by using a multilayer perceptron (MLP) classifier, denoted as "DCASE2017 Baseline". The MIL-NN is a multiple instance learning based neural network system proposed in [53]. The CNN-ensemble system is proposed by [16] and ranked the 1st in the SED subtask in Task 4 of the DCASE 2017 challenge. Our proposed systems achieve an F1 score of 0.646 on the evaluation set, outperforming the other methods in AT. The CNN-biGRU and the CNN-Transformer systems achieve similar performance. Table VI shows the SED result with different methods. On the evaluation set, our proposed CNN-biGRU-Att with automatic thresholds optimization achieves an F1 score of 0.584, outperforming other systems. The system achieves an ER of 0.68 which is comparable with the ensemble based CNN system [16].

VII. CONCLUSION

This paper investigates sound event detection (SED) with weakly labelled data. The variants of convolutional neural networks (CNNs) and CNN-Transformer systems were proposed for the audio tagging and sound event detection. The segment-wise training and clip-wise training systems were investigated. The clip-wise training achieves an mAP of 0.650 in audio tagging and an ER of 0.68 in SED. A novel automatic thresholds optimization method is proposed to approach the thresholds selection problem. The automatic thresholds optimization method improves the AT F1 score from 0.629 to 0.646, and reduces the ER from 0.78 to 0.68. We show that the CNN-Transformer performs similarly to the CRNN system, while

the CNN-Transformer has the advantage of being computed in parallel. In addition, the improvements of audio tagging and SED are mainly from the automatic threshold optimization. In future, we will focus on extending the sound event detection systems to large-scale training data such as AudioSet.

REFERENCES

[1] 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. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2013, pp. 1–4.

[2] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct. 2015.

[3] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. IEEE Eur. Signal Process. Conf.*, 2016, pp. 1128–1132.

[4] A. Mesaros *et al.*, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. Workshop Detect. Classific. Acoust. Scenes Events*, 2017, pp. 85–92.

[5] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," *Workshop Detect. Classific. Acoust. Scenes Events*, 2018.

[6] "DCASE 2017 Task 4," 2017. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-large-scale-sound-event-detection

[7] J. F. Gemmeke *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events," in *IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 776–780.

[8] E. Cakır, T. Heittola, and T. Virtanen, "Domestic audio tagging with convolutional neural networks," DCASE2016 Challenge, Tech. Rep., 2016. [Online]. Available: http://dcase.community/challenge2016

[9] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3461–3466.

[10] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 131–135.

[11] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proc. ACM Multimedia Conf.*, 2016, pp. 1038–1047.

[12] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 121–125.

[13] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 791–795.

[14] S.-Y. Chou, J.-S. R. Jang, and Y.-H. Yang, "Learning to recognize transient sound events using attentional supervision." in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3336–3342.

[15] J. Salamon, B. McFee, and P. Li, "DCASE 2017 submission: Multiple instance learning for sound event detection," DCASE2017 Challenge, Tech. Rep., 2017. [Online]. Available: http://dcase.community/challenge2017

[16] K. Lee, D. Lee, S. Lee, and Y. Han, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," DCASE2017 Challenge, Tech. Rep., Sep. 2017. [Online]. Available: http://dcase.community/challenge2017

[17] S. Chou, J. Jang, and Y.-H. Yang, "FrameCNN: A weakly-supervised learning framework for frame-wise acoustic event detection and classification," DCASE2017 Challenge, Tech. Rep., 2017. [Online]. Available: http://dcase.community/challenge2017

[18] S. Adavanne, P. Pertilä, and T. Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 771–775.

[19] L. Ford, H. Tang, F. Grondin, and J. Glass, "A deep residual network for large-scale acoustic scene analysis," in *Proc. INTERSPEECH*, 2019, pp. 2568–2572.

[20] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2015.

[21] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017.

[22] T. Hayashi, S. Watanabe, T. Toda, T. Hori, J. Le Roux, and K. Takeda, "BLSTM-HMM hybrid system combined with sound activity detection network for polyphonic sound event detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 766–770.

[23] J. L. Dai Wei, P. Pham, S. Das, S. Qu, and F. Metze, "Sound event detection for real life audio DCASE challenge," DCASE2016 Challenge, Tech. Rep., 2016. [Online]. Available: http://dcase.community/challenge2016

[24] X. Xia, R. Togneri, F. Sohel, and D. Huang, "Frame-wise dynamic threshold based polyphonic acoustic event detection," in *Proc. INTERSPEECH*, 2017, pp. 474–478.

[25] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," in *Proc. INTERSPEECH*, 2016, pp. 3653–3657.

[26] Y. Wang and F. Metze, "A first attempt at polyphonic sound event detection using connectionist temporal classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2986–2990.

[27] S. Gururani, C. Summers, and A. Lerch, "Instrument activity detection in polyphonic music using deep neural networks." in *Proc. Int. Soc. Music Inf. Retrieval*, 2018, pp. 569–576.

[28] J. Lee, J. Park, S. Kum, Y. Jeong, and J. Nam, "Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection," *Proc. Workshop Detect. Classific. Acoust. Scenes Events (DCASE)*, 2017, pp. 69–73.

[29] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2392–2396.

[30] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017.

[31] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Advances Neural Inform. Process. Syst.*, 2017, pp. 5998–6008.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2019, pp. 4171–4186.

[33] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," 2019, *arXiv:1906.04972*.

[34] L. Cances, P. Guyot, and T. Pellegrini, "Evaluation of post-processing algorithms for polyphonic sound event detection," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2019, pp. 318–322.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[36] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[37] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *Int. Soc. Music Inf. Retrieval*, 2016.

[38] Q. Kong, T. Iqbal, Y. Xu, W. Wang, and M. D. Plumbley, "DCASE 2018 challenge baseline with convolutional neural networks," in *Proc. Workshop Detect. Classific. Acoust. Scenes Events*, 2018, pp. 217–221.

[39] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," in *Proc. Int. Conf. Learn. Representations*, 2017.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015.

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[42] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[45] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley, "Weakly labelled audioset tagging with attention neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 11, pp. 1791–1802, Nov. 2019.

[46] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int. Conf. Learn. Representations*, 2014.

[47] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah, "Large-scale weakly labeled semi-supervised sound event detection in domestic environments," in *Proc. Detect. Classific. Acoust. Scenes Events Workshop*, 2018, pp. 19–23.

[48] N. Turpault, R. Serizel, A. P. Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Proc. Detect. Classific. Acoust. Scenes Events Workshop*, 2019, pp. 253–257.

[49] Q. Kong, Y. Cao, T. Iqbal, Y. Xu, W. Wang, and M. D. Plumbley, "Cross-task learning for audio tagging, sound event detection and spatial localization: DCASE 2019 baseline systems," 2019, *arXiv:1904.03476*.

[50] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 8609–8613.

[51] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *Proc. Int. Conf. Learn. Representations*, 2018.

[52] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Appl. Sci.*, vol. 6, no. 6, pp. 162–178, 2016.

[53] S.-Y. Tseng, J. Li, Y. Wang, J. Szurley, F. Metze, and S. Das, "Multiple instance deep learning for weakly supervised audio event detection," in *Proc. INTERSPEECH*, 2018, pp. 3279–3283.

Wenwu Wang (Senior Member, IEEE) was born in Anhui, China. He received the B.Sc. degree in 1997, the M.E. degree in 2000, and the Ph.D. degree in 2002, all from Harbin Engineering University, China. He then worked in King's College London, Cardiff University, Tao Group Ltd. (now Antix Labs Ltd.), and Creative Labs, before joining University of Surrey, UK, in May 2007, where he is currently a Professor in signal processing and machine learning, and a Co-Director of the Machine Audition Lab within the Centre for Vision Speech and Signal Processing. He has been a Guest Professor at Qingdao University of Science and Technology, China, since 2018. His current research interests include blind signal processing, sparse signal processing, audio-visual signal processing, machine learning and perception, machine audition (listening), and statistical anomaly detection. He has (co)-authored over 200 publications in these areas. He served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2014 to 2018. He currently serves as Senior Area Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING and Associate Editor for IEEE/ACM Transactions on Audio Speech and Language Processing.

Qiuqiang Kong (Student Member, IEEE) received the B.Sc. and M.E. degrees from the South China University of Technology, Guangzhou, China, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree from the University of Surrey, Guildford, U.K on sound event detection. His research topic includes sound understanding, audio signal processing and machine learning. He was nominated as the postgraduate research student of the year in University of Surrey, 2019.

Yong Xu (Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2015, on the topic of DNN-based speech enhancement and recognition. Currently, he is a Senior Research Scientist in Tencent AI lab, Bellevue, USA. He once worked at the University of Surrey, U.K. as a Research Fellow from 2016 to 2018 working on sound event detection. He visited Prof. Chin-Hui Lee's lab in Georgia Institute of Technology, USA from Sept. 2014 to May 2015. He once also worked in IFLYTEK company from 2015 to 2016 to develop far-field ASR technologies. His research interests include deep learning, speech enhancement and recognition, sound event detection, etc. He received 2018 IEEE SPS best paper award.

Mark D. Plumbley (Fellow, IEEE) received the B.A.(Hons.) degree in electrical sciences and the Ph.D. degree in neural networks from the University of Cambridge, Cambridge, U.K., in 1984 and 1991, respectively. Following his PhD, he became a Lecturer at King's College London, before moving to Queen Mary University of London in 2002. He subsequently became Professor and Director of the Centre for Digital Music, before joining the University of Surrey in 2015 as Professor of Signal Processing. He is known for his work on analysis and processing of audio and music, using a wide range of signal processing techniques, including matrix factorization, sparse representations, and deep learning. He is a co-editor of the recent book on Computational Analysis of Sound Scenes and Events, and Co-Chair of the recent DCASE 2018 Workshop on Detection and Classifications of Acoustic Scenes and Events. He is a Member of the IEEE Signal Processing Society Technical Committee on Signal Processing Theory and Methods, and a Fellow of the IET and IEEE.