Learning-based Cooperative Sound Event Detection with Edge Computing

Jingrong Wang[†], Kaiyang Liu[†*], George Tzanetakis[†], Jianping Pan[†]
[†]Department of Computer Science, University of Victoria, Victoria, Canada
[*]School of Information Science and Engineering, Central South University, Changsha, China
Email: {jingrongwang, liukaiyang, pan}@uvic.ca, gtzan@cs.uvic.ca

Abstract—**In this paper, we propose a novel real-time sound event detection framework, which combines multi-label learning and edge computing, to classify and localize abnormal sound events for city surveillance. Multiple devices equipped with acoustic sensors are deployed to collect the audio information. A learning-based approach is introduced to address the difficulties of accurately classifying the temporally overlapping acoustic events in a noisy environment. Then, edge computing is adopted to handle the high processing complexity of the learned analytics. Computation-intensive tasks of classification and localization can be offloaded to the nearby edge server for low-latency sound detection. An ensemble-based cooperative decision-making algorithm is also presented to aggregate the information from distributed devices in order to obtain better classification results. Extensive evaluations show the effectiveness of edge computing which helps reduce the time latency as well as the superiority of cooperative post-processing on the edge server to obtain a high accuracy.**

Index Terms—**Sound event detection, cooperative processing, edge computing, audio classification, deep learning**

I. Introduction

City security has been an important concern due to the violent crimes and especially gun violence. In 2017, there were more than six thousand shootings reported and confirmed in the US [1]. Traditional alarm process consists of an incoming call, security notification, finding cameras nearby and starting an investigation, which delays the response time to gun violence incidents. Compared to video surveillance which is not sensitive to abnormal sounds and is limited by blind areas, sound event detection has been utilized in recent years for city surveillance.

In a complex auditory environment, multiple sound events may be temporally overlapping. To detect these sound events, deep learning is widely used to provide intelligent detection analytics. In most of the existing work, end devices equipped with acoustic sensors keep streaming the audio to the applications for processing, which can be run locally or remotely [2]. Applications such as Google Translate choose to compress the deep learning algorithms and execute the task locally, i.e., run on the front-end devices. However, this approach may fail to meet the requirements of the processing latency and classification accuracy. Furthermore, it also incurs a high energy consumption when continuously running for a long time [3]. On the other hand, interactive smart assistants such as Apple Siri and Microsoft Cortana send the raw input data to the remote cloud for storage, processing and analysis. The limitation of this approach is that the performance will be severely affected by the network conditions and may lead to a high latency due to the transmission congestion or limited bandwidth [4].

The front-end devices often have limited computation capabilities, whereas utilizing the cloud may incur high communication latencies. To cope with these challenges, edge computing is considered as a promising solution for delay-sensitive computation-intensive services. It enhances and extends the cloud services at the edge of the network. More specifically, edge computing deploys computation capacity closer to where the data are captured and generated. This meets the real-time processing requirements of abnormal sound event detection.

In this paper, a learning-based sound event detection framework is proposed to classify and localize abnormal events in real time. It utilizes edge servers to help offload the computation-intensive tasks. The extracted audio features are fed into a frame-level deep neural network for sound event classification. The evaluation shows that the classification results from various acoustic devices are biased due to the different distances of the sound propagation. An ensemble-based algorithm is deployed on the edge server for cooperative classification, which combines the information aggregated from multiple acoustic devices to obtain an accurate result. Then, we formulate the sound source localization problem with the time difference of arrival (TDOA) and solve it by the least-squares minimization method. The main contributions of this paper are as follows:

- Leveraging edge computing, a learning-based sound event detection framework is proposed to classify and localize abnormal sounds in real time.
- Ensemble learning is employed to improve the classification accuracy.
- Extensive evaluations show that the performance of the latency and the detection accuracy are considerably improved by 64% and 48%, respectively.

The remainder of this paper is structured as follows. The related work is introduced in Section II. The system model is presented in Section III. The algorithms for sound classification and source localization are proposed in Section IV. The setup and the performance evaluation are analyzed in Section

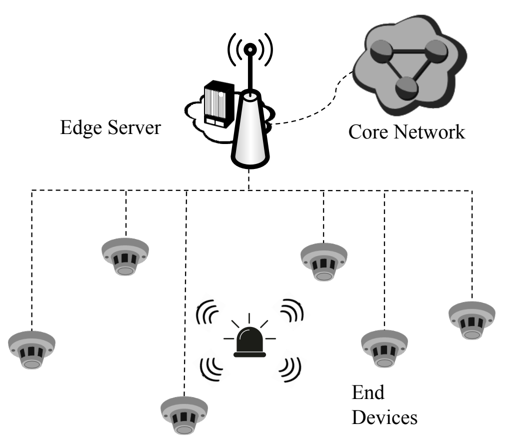


Figure 1. Abnormal sound detection in edge computing.

V. Section VI concludes the paper and puts forward some future work.

II. Related Work

A. Audio Analysis

In the audio analysis, classifiers are trained to differentiate sounds by utilizing extracted features such as Mel-Frequency Cepstral Coefficients (MFCC) and spectral centroid [5]. Guo et al. [6] trained support vector machines (SVMs) to classify the acoustic events and achieved a lower error rate when compared with nearest neighbor (NN), k-NN and nearest center. Clavel et al. [7] built Gaussian mixture models (GMMs) for each sound class based on MFCC extracted from the audio and selected the sound class with the highest Maximum A Posteriori (MAP) score as the classification result. Heittola et al. [8] modeled feature distributions as continuous-density hidden Markov models (HMMs) by recognizing the context and utilizing the specific context information. However, in a noisy environment, the traditional methods are not accurate enough to detect temporally overlapping sound events.

To address this problem, deep learning approaches are widely used in sound event detection. Cakir et al. [9] trained multi-label deep feed-forward neural networks (DNNs) with frame-wise spectral-domain features in polyphonic sound event classification and showed that their approach outperforms the more traditional HMM. Hershey et al. [10] classified the sounds through convolutional neural networks (CNNs) in which common structures can be efficiently captured by the convolutional units. To reduce the classification error rate, Xu et al. [11] proposed a convolutional gated recurrent neural network (CGRNN) leveraging the capability of recurrent units to learn long-term sound patterns. Similar to the approaches mentioned above, deep learning is used in our work for sound detection. Although learning-based approaches obtain high accuracies, they are computation-intensive and require high CPU and memory cost. The main challenge we tackle in this work is how to provide low latency for audio stream analysis while maintaining high classification accuracy.

B. Task Offloading and Edge Computing

Edge computing has been an effective solution to reduce the latency of computation-intensive applications. Noble et al. [12] showed its potential by offloading the speech recognition task to a resource-limited mobile device. The client can either recognize the utterance locally or send it to the remote server equipped with a speech recognition system based on HMM. Hu et al. [13] evaluated the performance of edge computing over both Wi-Fi and LTE with three statically pre-partitioned applications, i.e., face recognition, augmented reality and fluid simulation. The results showed that compared to local execution, offloading the task to a nearby computation node significantly reduced the response time and energy consumption. Ran et al. [3] designed an offloading framework to run deep learning algorithms for video object recognition. The task can be executed on front-end devices or the remote cloud based on the client-side decisions. However, the extra transmission delay may lead to data staleness in real-time processing, decreasing the accuracy of object recognition. Wang et al. [14] addressed the computation-intensive transcoding with edge computing and proposed an adaptive wireless video transcoding framework. The source video stream only needs to be sent to the edge server once and the edge server can transcode videos according to the requests.

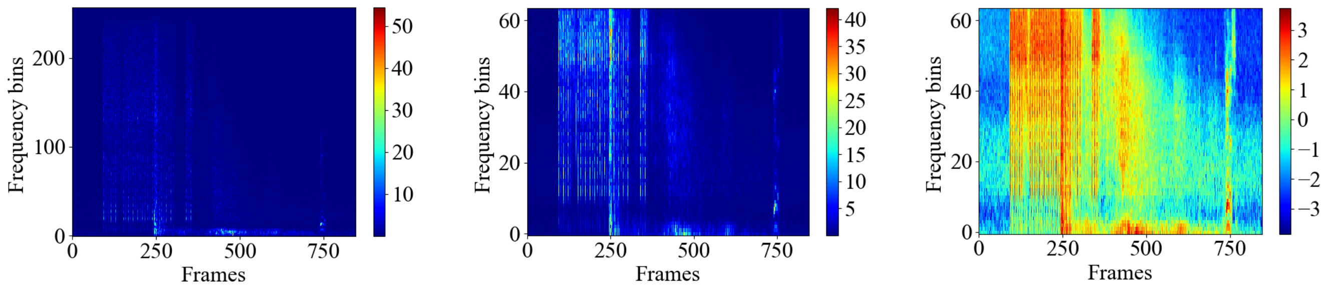
Similar to the literature, we leverage the emerging edge computing for intensive computation and communication in the learning-based audio analysis. In contrast to the multimedia applications mentioned above, real-time sound event detection highlights the similarity of the audio recordings captured by nearby acoustic devices. Considering the high computation capability, edge computing has a natural advantage in cooperative processing and thus provides a quick and precise response to the abnormal sound. In this paper, we focus on the learning-based sound event detection that takes advantage of edge computing. A cooperative post-processing step on the edge server based on an ensemble approach is proposed.

III. System Model

In this section, the system architecture followed by the communication and computation model is introduced.

A. System Architecture

The system model for abnormal sound detection in edge computing is illustrated in Fig. 1. Front-end acoustic devices $\mathcal{N} = \{1, 2, ..., N\}$ equipped with a microphone and limited computation capabilities are deployed to capture and detect the abnormal sound in real time with machine learning algorithms. The devices can access the edge server through a wireless base station (BS) and then a wired connection [15]. In distributed audio acquisition scenarios, we assume that the devices can communicate with each other. The devices can either process the data locally or offload the task to the edge server. The edge server has higher computation capabilities and thus provides low-latency services.



(a) Spectrogram

(b) Mel spectrogram

(c) Log-scale mel spectrogram

Figure 2. Audio feature extraction.

B. Communication Model

The surveillance devices keep detecting the environment and sending the data to the edge server. The received signal strength (in dBm) for BS from device $n \in \mathcal{N}$ can be calculated based on the path loss model, which is expressed as [16]

$$P_n = P^{\mathsf{TX}} - PL_n - X_{\sigma_1}, \tag{1}$$

where P^{TX} (in dBm) is the transmitted power of device n. For simplicity, we assume that all devices have the same transmission power, X_{σ_1} denotes the shadowing fading (in dB) subject to the Gaussian distribution with zero mean and standard deviation σ_1, and PL_n denotes the path loss (in dB) between device n and the BS

$$PL_n = PL(d_0) + 10\theta \log\left(\frac{d_n}{d_0}\right), \tag{2}$$

where d_n (in m) $\geq d_0$ is the distance between them, θ is the path loss exponent, and d_0 is the reference distance for the antenna far-field propagation effect.

Based on Shannon's Theory, the maximum uplink transmission rate for device n can be calculated as

$$\gamma_n^{\mathsf{TX}} = W \log_2(1 + \frac{10^{P_n/10}}{I_n + N_0}), \tag{3}$$

where W is the channel bandwidth, N_0 is the noise power and I_n is the interference signal from other devices within the radius r when transmitting simultaneously

$$I_n = \sum_{n'=1}^{N} 10^{P_{n'}/10}, n' \neq n, n' \in \mathcal{N}. \tag{4}$$

In the transport layer, UDP is used for communication. Compared to TCP which requires a 3-way handshake and is suitable for delay-tolerant applications, UDP is more suitable for applications that are delay-sensitive.

C. Computation Model

Three different modes can be used to process the data: 1) each device classifies the sound event individually and localizes the sound source after communicating with other devices, 2) the data gathered by multiple end devices are sent to the edge server for processing, and 3) the BS is treated as a relay to send the data to the cloud for classification and localization.

1) ***Local task execution****:* The local task execution time is denoted as

$$D^{\mathsf{LP}} = \frac{C}{f}, \tag{5}$$

where C is the computation cycles of the task and f is a default set of the allocated computation speeds in terms of cycles per second for all devices. Therefore, the total response time of local execution on device n is

$$D^{\mathsf{L}} = D^{\mathsf{S}} + D^{\mathsf{LP}}, \tag{6}$$

where D^{S} is the sound travel time determined by the sound speed and the distance between the sound source and the end device. The cost for synchronization among sensors is ignored due to the relatively small data size.

2) ***Task execution on the edge server****:* In this mode, the end device needs to transmit the data to the edge server. The transmission delay of the task is given by

$$D^{\mathsf{TX}} = \frac{D}{\gamma^{\mathsf{TX}}}, \tag{7}$$

where D is the data size transmitted to the edge server and γ^{TX} is the data transmission rate.

The computation delay of the task executed at the edge server consists of the time for both classification and localization, which is computed as

$$D^{\mathsf{EP}} = \eta \frac{C}{\lambda f}, \tag{8}$$

where η is the computational gain when compared with the first-come-first-serve sequential approach [17], λ is the computation capability ratio between the edge server and the end device, and λf is the allocated computation speeds for the task on the edge server.

Thus, the overall response time of executing the task at the edge service D^{E} consists of the sound travel, transmission and computation delay

$$D^{\mathsf{E}} = D^{\mathsf{S}} + D^{\mathsf{TX}} + D^{\mathsf{EP}}. \tag{9}$$

*3) **Task execution on the cloud server:*** The latency in the core network is given by

$$D^{\mathsf{I}} = D^{\mathsf{Q}} + D^{\mathsf{R}}, \tag{10}$$

where D^{Q} is the queueing delay in the BS and $D^{\mathsf{R}} = \frac{D}{\gamma^{\mathsf{I}}}$ is the transmission delay in the core network. γ^{I} denotes the transmission rate in the core network.

As the computation capability in the cloud could be more powerful, we assume the computation capability ratio between the cloud and the edge is μ. The total response time in cloud computing is expressed as

$$D^{\mathsf{C}} = D^{\mathsf{S}} + D^{\mathsf{TX}} + D^{\mathsf{I}} + D^{\mathsf{CP}}, \tag{11}$$

where $D^{\mathsf{CP}} = D^{\mathsf{EP}}/\mu$.

For all three modes, considering the relatively small data size of the detection results, similar to the previous studies [18], the response time of reporting the result is not considered in this work.

IV. Sound Event Detection

A. Sound Event Classification

1) Dataset: The Google AudioSet [19] used in this work consists of 2.1 million 10-second audio excerpts that are collected from YouTube videos, providing large-scale evaluation tasks for the sound event detection. There exist 632 human-labeled sound event classes covering numerous human sounds, daily environmental sounds and abnormal sounds such as fusillade, gunfire, explosion and fire alarm. The balanced dataset provided by Google contains the segments of each class with at least 59 examples. Therefore, the unbalanced dataset contains the remainder of the dataset. The Google Audioset provides audio features extracted from the video and describes YouTube video with video ID, start time, end time and its corresponding labels. Audio features are extracted by the VGG acoustic model [10] and the labels of sound classes are mapped to integers.

2) Feature extraction: The features used for sound event detection are vectors in a 128-dimensional space. The audio is first divided into non-overlapping frames with a default length of 960 ms. As shown in Fig. 2(a), the spectrogram is then calculated by a Short-time Fourier Transform with a periodic Hann window. The window size is 25 ms and the window hop is 10 ms. To obtain a mel spectrogram shown in Fig. 2(b), the spectrogram is mapped to 64 mel-bins, which varies from 125Hz to 7500 Hz. After taking a logarithm [1], the resulting log-mel spectrogram contains 96 $\times$ 64 bins. A batch of 128 patches is randomly sampled from all patches, which constitute the input of the classifier.

3) Algorithm: As shown in Fig. 3, Deep Bag-of-Frames (DBoF) is proposed in YouTube-8M model to classify and determine different sounds [20].

The 128-dimensional frame-level features as the input extracted above are first projected onto a higher dimensional space through a fully connected layer with shared parameters.

[1]To avoid the logarithm of zero, 0.01 is added in the VGG model.

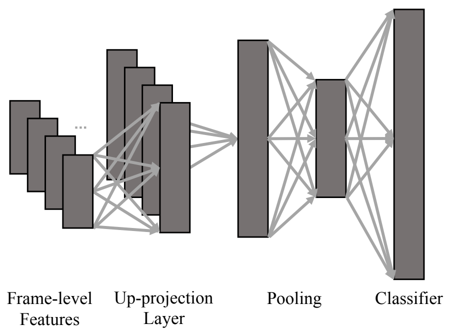


Figure 3. The network architecture of the DBoF [20].

Then, a pooling layer converts the frame-level sparse codes into a video-level representation. It aggregates the codes of the all frames into a single fixed-length video representation. Next, max pooling is used to perform the aggregation where the score for each class is the maximum value across all classifiers. To improve the stability and speed up convergence, a batch normalization layer is used before pooling. Last, the obtained fixed length descriptor of the audio is fed into a few hidden layers, and a classification layer provides the final video-level predictions.

B. Sound Source Localization

Considering the sound propagation model, the spreading loss model is applied in this work. The equation to compare the sound pressure levels at two different locations is

$$Lp_1 - Lp_2 = 20\log(\frac{r_1}{r_2}), \tag{12}$$

where Lp is the sound pressure level (dB re $2{\times}10^{-5}$ N/m^2). This means doubling the distance from the source, e.g., $r_2 = 2r_1$, corresponds to a loss of 6 dB.

The travel time of emitting the sound signals helps to estimate relative distances between the acoustic devices and the sound source. The TDOA at multiple reference devices can be used as the positioning parameter to measure the distances between the sound source and multiple reference devices [21]. It does not require the absolute time and the special type of antennas, showing the superiority to the time of arrival method and the direction of arrival method. A hyperbola is generated between the two acoustic devices by a particular TDOA information on which the sound source may exist. The location of the sound source is in the intersection of all hyperbolas. Thus, the problem is defined as a localization problem which finds the location of sound events in the environment detected by these end devices. When all data aggregated from the end devices are sent to the server, the server can find the time difference among the audio data through the filter and pick the peak with Fourier transform. A general assumption is that the correspondence between different sound events can be captured as long as there exist sufficiently long time gaps between any two sound events.

Let X denote the location of end devices. A is the location of sound events and T is the emission time for the sound signals, which can be defined as follows

$$X = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_N & y_N \end{bmatrix}, \ A = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_M & b_M \end{bmatrix}, \ T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_M \end{bmatrix}. \quad (13)$$

We assume that the locations of N end devices are already known, and the locations of M sound events and the corresponding emission time for the sound signals are unknown. Then, the detection time of the sound events by the individual end devices are expressed as

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,M} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N,1} & d_{N,2} & \cdots & d_{N,M} \end{bmatrix}. \quad (14)$$

In (14), $d_{i,j}$ can be calculated as

$$d_{i,j} = t_j + c^{-1} \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} a_j \\ b_j \end{pmatrix} \right\|_2 + X_{\sigma_2}, \quad (15)$$

where $||\cdot||_2$ is the Euclidean distance, c is the sound speed, and X_{σ_2} is an additive white Gaussian noise with zero mean and standard deviation σ_2. More complex scenarios can also be applied here. Let (a_i^*, b_i^*) denote the relative location of source i to node n. Thus, the matrix of relative device location is calculated as

$$X^* = \begin{bmatrix} x_1^* & y_1^* \\ x_2^* & y_2^* \\ \vdots & \vdots \\ x_N^* & y_N^* \end{bmatrix} = \begin{bmatrix} x_1 - x_n & y_1 - y_n \\ x_2 - x_n & y_2 - y_n \\ \vdots & \vdots \\ x_N - x_n & y_N - y_n \end{bmatrix}. \quad (16)$$

This relative arrival time is defined as

$$\begin{aligned} d_{i,j}^* &= t_j + c^{-1} \left\| \begin{pmatrix} x_i^* \\ y_i^* \end{pmatrix} - \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 - t_j - c^{-1} \left\| \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 \\ &= c^{-1} \left\{ \left\| \begin{pmatrix} x_i^* \\ y_i^* \end{pmatrix} - \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 - \left\| \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 \right\}. \end{aligned} \quad (17)$$

According to (14), the matrix of the arrival time is calculated as

$$D^* = \begin{bmatrix} d_{1,1} - d_{n_1,1} & d_{1,2} - d_{n_2,2} & \cdots & d_{1,M} - d_{n_M,M} \\ d_{2,1} - d_{n_1,1} & d_{2,2} - d_{n_2,2} & \cdots & d_{2,M} - d_{n_M,M} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N,1} - d_{n_1,1} & d_{N,2} - d_{n_2,2} & \cdots & d_{N,M} - d_{n_M,M} \end{bmatrix}. \quad (18)$$

After getting the TDOA information, the sound source localization problem is transformed into a least-squares formulation, i.e., minimizing the quadratic difference between the predicted relative measurements and the actual measurements. The optimization problem is shown as follows

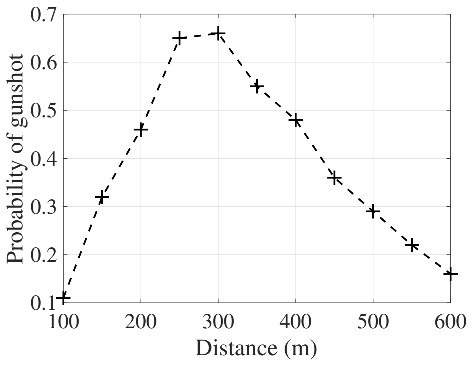


Figure 4. Effect of the distance in audio classification.

$$\begin{aligned} A^* = \arg\min_A \sum_{i=1}^{N} \sum_{j=1}^{M} \Bigg\{ & \left\| \begin{pmatrix} x_i^* \\ y_i^* \end{pmatrix} - \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 \\ & - \left\| \begin{pmatrix} a_j^* \\ b_j^* \end{pmatrix} \right\|_2 - d_{i,j}^* \Bigg\}^2 . \end{aligned} \quad (19)$$

This formulated optimization problem can be solved efficiently by a sequential quadratic programming solver [22]. One of the performance metrics to evaluate the results of localization is defined as the root mean square (RMS) error between the predicted location and the real location. As TDOA can find the location of an event in the intersection of all hyperbolas, the noise deteriorates the situation where there is no cross point among all curves. Thus, the area ratio of deadzone, which is defined as the ratio of the number of points that cannot be localized to the total number of points tested, is also evaluated in Section V.

C. Cooperative Processing

As shown in Fig. 4, the classification accuracy is affected by the distance between the sound source and the acoustic device. The performance of the classifier depends on the sound pressure level of the audio that is used for training. In Fig. 4, the reference distance of the trained instances is 250 m. Thus, the audio aggregated from different location results in a variety of classification performance. Therefore, the scenario of sound event detection forms a multi-classifier system. The individual devices can be considered as weak learners with bias. The edge server can be considered as an aggregate learner that makes a decision based on these weak learners. The observations aggregated from the devices do not equally contribute to the final classification result. We now consider how to combine the results of these multiple devices into a strong classifier that is superior to any of the individual devices.

Ensemble learning merges multiple learners to obtain a more accurate prediction than any individual learner alone. Majority voting in bootstrap aggregating, a typical algorithm in ensemble learning, makes final decisions following the consensus of more than half of the classifiers, which means

Algorithm 1 Ensemble-based cooperation algorithm

```
1: Predict the labels of a sound event instance $m$ aggregated from each end device and record the confidence of the predicted class $p$, i.e., (20) and $v_{n,p}$.
2: Calculate the total vote for each predicted class $V(p) = \sum_{n=1}^{N} v_{n,p}$.
3: if $\max C'(m,p) > \epsilon$ OR $V(p) >= N/2$ then
4:     Class $p$ is added to the final decision.
5: else
6:     Class $p$ is not considered in the final decision.
7: end if
```

a high confidence in the final result. Only the classification results that receive a majority are added to the final decision. Otherwise, the results are not considered in the decision-making.

Let $C(m,p)$ denote the set of the probabilities of a certain sound event instance m predicted as the sound event p by N end devices, which is

$$C(m,p) = \{h_1(m,p), h_2(m,p), \ldots, h_N(m,p)\}. \quad (20)$$

Moreover, a binary variable $v_{n,p}$ is introduced to indicate whether the device n predicts class p ($v_{n,p} = 1$) or not ($v_{n,p} = 0$).

The ensemble-based cooperation (EC) algorithm that makes the final decision based on the aggregated information is illustrated in Algorithm 1. The edge server is naturally a collector, processing the data with high computation capability and aggregating each result without increasing the network traffic among the end devices. After processing the data, the edge server combines the individual decisions through a majority vote. Intuitively, the prediction result is credible if most of the devices classify the sound. Moreover, if some classifiers have the confidence of an event higher than a certain threshold ϵ, the prediction will also be immediately added to the final decision. In comparison with EC, in local execution, if the confidence of a certain sound event exceeds the threshold ϵ, the event will be confirmed and added to the final decision.

V. Performance Evaluation

The implementation of sound event detection with the support of edge computing is presented in this section.

A. Setup

Acoustic end devices are deployed in a grid, which is illustrated in Fig. 5. The neural networks are implemented on the basis of Tensorflow. The hardware used for the experiments features a 2.9 GHz Intel Core i5 processor and 8 GB of memory. The path loss model PL_n of the end devices is $127 + 30 * \log_{10}(d_n/1000)$ [23]. The parameters are summarized in Table I. In the default setting, the number of devices $N = 9$, the computation capability ratio $\mu = \lambda = 8$ and the threshold $\epsilon = 0.3$.

A series of gunshot audio consisting of speech and fusillade fetched from the Google Audioset is tested. In the experiments, local execution (Local), edge computing (Edge) and

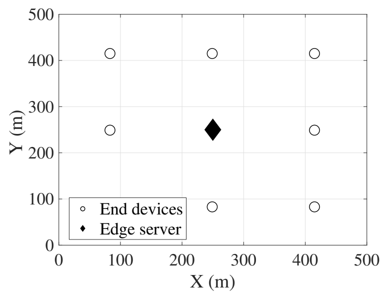


Figure 5. Grid deployment.

Table I
System parameters in the simulation

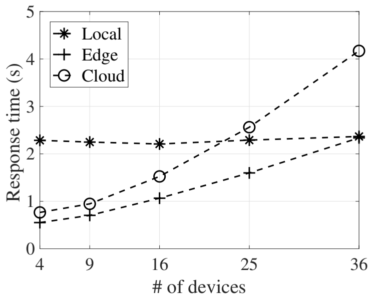
Parameter	Value	Parameter	Value
Area	500 m×500 m	r	100 m
W	20 MHz	D	3840 kbit
P^{TX}	23 dBm	N_0	-174 dBm
$1/\eta$	4.28 [24]	σ_1	3.6
σ_2	1	γ^{I}	[2, 10] Mbps

cloud computing (Cloud) are conducted for a fair performance comparison. In Local, each device will first classify the sound event individually and then communicate with other devices to finalize the sound event and localize the sound source. In Edge, the data gathered by the device will be sent to the edge server for classification and localization. In Cloud, the sink BS will send the data to the cloud. Considering the dynamic network condition, the transmission rate in the core network is randomly selected between 2 and 10 Mbps [25].

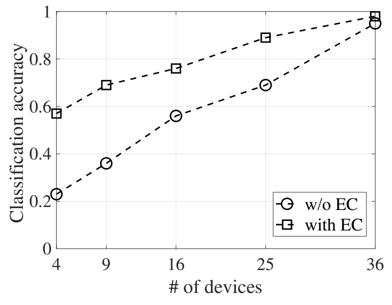
B. Impact of the Number of Devices

The effect of the number of devices on the response time is shown in Fig. 6(a). The device can execute the task independently where the processing time dominates. However, the response time of local execution, which depends on the computation capability of the distributed devices, substantially remains constant and maintains some fluctuation due to the dynamic CPU allocation. The more devices deployed in the grid area, the more data generated by all devices, which indicates that the edge server may receive and process more requests. Thus, the response time for task offloading increases with the number of devices due to the increased workload. The impact is more obvious in Cloud where the bottleneck is the data traffic in the core network. In the default setting, the performance of Edge is improved by 64% when compared with Local. To some extent, the response time of task offloading may experience a sharp deterioration with a large number of devices, which is affected by the computation capability ratio between the server and the end device.

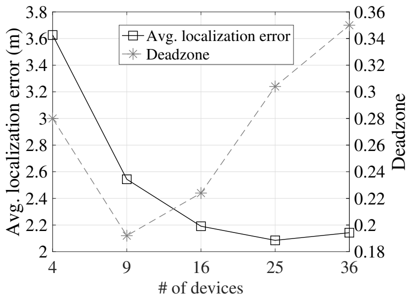
Fig. 6(b) presents the effect of the number of devices on the classification accuracy. A small number of devices cannot cover the detection area well and thus lead to a low classification accuracy. With more devices participating in detecting and classifying the sound event, the classification accuracy increases with a better coverage of the spot of the



(a) Response time



(b) Classification accuracy.



(c) Localization performance.

Figure 6. Impact of the number of devices.

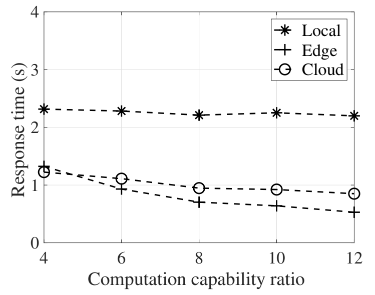


Figure 7. Impact of the computation capability ratio on the response time.

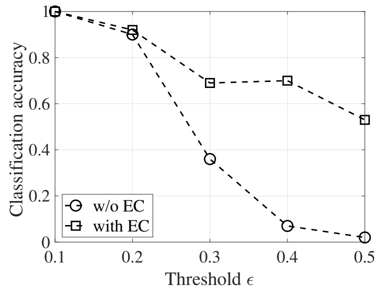


Figure 8. Impact of the threshold ϵ on the classification accuracy.

accident. With the support of the edge server which can easily aggregate and integrate all information, the classification accuracy is higher in EC because the vast majority of the devices can reach a consensus even with a low confidence. In the default setting, the classification accuracy is increased by 48 % with EC. It is worth noting that there exists a trade-off between the response time and the classification accuracy where the increase of the devices will not only improve the classification performance but also lead to a severe latency.

Moreover, as shown in Fig. 6(c), the effect of the number of devices on the localization performance is also studied. With the increasing number of devices, the location error is reduced as more location information is provided by the devices. It drops slower when a large number of devices are involved. Deadzone ratio first declines as more devices can cover the area better and thus narrow down the deadzone. However, it then faces a consistent growth. Although more devices provide more information for localization, much more measurement noise will make it unsolvable for the optimization problem to find the sound source location.

C. Impact of the Computation Ratio

Fig. 7 shows the effect of the computation ratio λ and μ. Here, we set $\mu = \lambda$. With the increase of the computation ratio, the total response time of task offloading continually drops. Edge outperforms Cloud even though the data processing speed of Cloud is much higher than Edge. The bottleneck of Cloud is the data transmission in the core network, which dominates the total response time. In the default setting, when compared with Local and Cloud, the response time in Edge is reduced by 68% and 26%, respectively.

D. Impact of the Threshold ϵ

The impact of the threshold ϵ is illustrated in Fig. 8. Intuitively, the confidence of the predicted event is affected by the trained model. By maintaining the same classification model, a lower threshold allows the devices output the results with low confidence. Without EC, a large ϵ sets a high threshold to determine whether the predicted event should be added to the final decision. This leads to a lower accuracy because the predicted results are not considered. However, in EC, the final decision not only depends on the confidence of the results predicted by individual information gathered from the devices, but also makes an ensemble based on the majority vote. If most of the evidence shows the existence of an event, an agreement will be made and reported the abnormal event. Using the same trained model, the classification accuracy increases almost tenfold with ϵ equal to 0.4.

E. Impact of the Device Deployment

Besides the grid deployment, other device deployment scenarios are also studied in the performance evaluation. An example is shown in Fig. 9 where devices are randomly deployed in the area and the edge server also stays in the center. As shown in Table II, four performance metrics are evaluated in both the grid and random deployment, i.e., response time with the support of edge server (RT), the classification accuracy with EC (CA), localization error (LE)

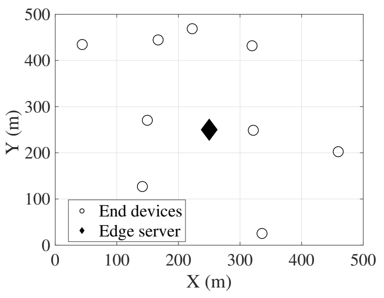


Figure 9. Random deployment.

Table II
IMPACT OF THE DEVICE DEPLOYMENT SCENARIO

Values	RT	CA	LE	DZ
Grid	0.752 s	69 %	2.34 m	18 %
Random	1.182 s	71 %	3.12 m	24 %

and the area ratio of deadzone (DZ). For a fair comparison, the topology of devices in random deployment will be updated in each iteration and the performance values are obtained by taking the average of 1,000 runs.

The response time in random deployment is higher than that in grid deployment because of the asymmetric locations of the devices. Some devices that are far from the edge server need a high communication cost to connect to the edge server. The classification performance is almost the same in these two deployments as they both make an ensemble on all the information aggregated from the end devices. Considering the performance of localization, random deployment performs worse than the grid deployment, as it cannot guarantee to cover the area well if the sound events are uniformly generated. Thus, both localization error and the area ratio of deadzone increase in the random deployment.

VI. Conclusion and Future Work

In this paper, an edge-assisted sound event detection framework is proposed to classify and localize the abnormal sound in city surveillance. Edge computing leverages the superiority in computation and communication, providing fast response to delay-sensitive, computation-intensive audio processing tasks. An ensemble-based cooperative processing algorithm is introduced to improve the classification accuracy at the edge server which aggregates the information from multiple end devices. Experiments show that the response time of executing a task is reduced with the involvement of edge computing. The accuracy of sound event detection is improved by combining with multiple observations of the sound signal.

In the future work, we plan to adopt a more realistic sound propagation model in a complex acoustic scenario. The distance-weighted differentiation can also be used in the post-processing at the edge.

Acknowledgment

This work is supported in part by NSERC, CFI and BCKDF.

References

[1] "Gun violence archive," http://www.gunviolencearchive.org/past-tolls.

[2] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[3] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. of IEEE INFOCOM*, 2018.

[4] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proc. of ACM SIGCOMM workshop on Mobile cloud computing*, 2013, pp. 15–20.

[5] B. Logan, "Mel frequency cepstral coefficients for music modeling." in *Proc. of ISMIR*, vol. 270, 2000, pp. 1–11.

[6] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, 2003.

[7] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *Proc. of IEEE ICME*, 2005, pp. 1306–1309.

[8] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1–13, 2013.

[9] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *Proc. of IEEE IJCNN*, 2015, pp. 1–7.

[10] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. of IEEE ICASSP*, 2017, pp. 131–135.

[11] 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. of IJCNN*, 2017, pp. 3461–3466.

[12] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," in *Proc. of ACM SIGOPS*, vol. 31, no. 5, 1997, pp. 276–287.

[13] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, and M. Satyanarayanan, "Quantifying the impact of edge computing on mobile applications," in *Proc. of ACM SIGOPS*, 2016, pp. 1–8.

[14] D. Wang, Y. Peng, X. Ma, W. Ding, H. Jiang, F. Chen, and J. Liu, "Adaptive wireless video streaming based on edge computing: Opportunities and approaches," *IEEE Transactions on Services Computing*, 2018.

[15] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Network*, vol. 32, no. 1, pp. 48–53, 2018.

[16] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[17] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[18] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing," *Future Generation Computer Systems*, vol. 64, pp. 1–14, 2016.

[19] "AudioSet download," https://research.google.com/audioset/download.html.

[20] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8M: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[21] S. Thrun, "Affine structure from sound," in *Proc. of NIPS*, 2006, pp. 1353–1360.

[22] K. Yu and Y. J. Guo, "Improved positioning algorithms for nonline-of-sight environments," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2342–2353, 2008.

[23] C. Niu, Y. Li, R. Q. Hu, and F. Ye, "Fast and efficient radio resource allocation in dynamic ultra-dense heterogeneous networks," *IEEE Access*, vol. 5, pp. 1911–1924, 2017.

[24] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[25] E. Dimogerontakis, R. Meseguer, and L. Navarro, "Internet access for all: Assessing a crowdsourced web proxy service in a community network," in *Proc. of ACM PAM*, 2017, pp. 72–84.