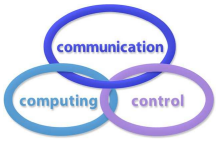
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 18, Issue: 5, Month: October, Year: 2023
Article Number: 5814, https://doi.org/10.15837/ijccc.2023.5.5814



CCC Publications

Urban Noise Analysis and Emergency Detection System using Lightweight End-to-End Convolutional Neural Network

J. Park, T. Yoo, S. Lee, T. Kim

Jinho Park
Department of Mechanical and Information Engineering/Smart Cities, University of Seoul
163 Seoulsiripdae-ro, Dongdaemun-gu, Seoul 02504, South Korea
jhparknr0116@uos.ac.kr

Taeyoung Yoo
Nota Inc.
332 Teheran-ro, Gangnam-gu, Seoul 06212, South Korea
taeyoung.yoo@nota.ai

Seongjae Lee
Department of Mechanical and Information Engineering/Smart Cities, University of Seoul
163 Seoulsiripdae-ro, Dongdaemun-gu, Seoul 02504, South Korea
junior209@uos.ac.kr

Taehyoun Kim*
Department of Mechanical and Information Engineering/Smart Cities, University of Seoul
163 Seoulsiripdae-ro, Dongdaemun-gu, Seoul 02504, South Korea
*Corresponding author: thkim@uos.ac.kr

Abstract

In recent years, the application of deep learning to environmental sound classification (ESC) has received considerable attention owing to its powerful ability to recognize the context of urban sounds. In general, deep learning models with high accuracy require substantial computing and memory resources. Consequently, to apply complex deep learning models to ESC in the real world, model inference has been performed on cloud servers with powerful computing resources. However, heavy network traffic and security issues occur when inferences are performed on a cloud server. In addition, deploying a deep learning model trained on a single public ESC dataset may not be sufficient for classifying various classes of urban noise and emergency-related sounds. To address these problems, we propose an on-device, real-time urban sound monitoring system that can classify various urban sounds at low system construction costs. The proposed system consisted of an edge artificial intelligence (AI) node and a FIWARE-based server. To enable the real-time inference on a resource-constrained edge AI node, we developed a lightweight convolutional neural network (CNN) by adjusting the input and model configurations to achieve high accuracy with a low number of parameters. The model achieved 94.9% classification accuracy using only 331 K parameters on an integrated dataset that included 17 classes of urban noises and emergencies. Furthermore, a prototype of the proposed system was developed and evaluated to verify its feasibility. The

prototype system was built at a cost of less than USD 50 and could perform the entire monitoring process every 3 s. We also visualized the sound monitoring data using Grafana on a FIWARE-based server.

Keywords: End-to-end neural network, environmental sound classification, FIWARE, on-device deep learning.

1 Introduction

Owing to rapid urbanization, noise pollution has become a significant environmental issue. Noise pollution causes various diseases, such as sleep disorders, cardiovascular diseases, and mental illnesses [27]. A standard approach to regulate noise pollution is to set maximum sound thresholds and apply noise mitigation activities when the noise exceeds the threshold [5]. However, this method relies only on the magnitude of the noise without recognizing the context of the noise source; hence addressing noise pollution based on the type of noise is difficult.

To overcome this limitation, environmental sound classification (ESC) has recently received considerable attention [6]. By using the ESC to classify the context of noise sources, manual work can be reduced, and effective decision-making for noise mitigation policies can be facilitated [19, 28, 30]. Moreover, since sound carries rich contextual information, ESC is widely used for intelligent urban sound monitoring in smart cities. For example, studies have been conducted to detect emergency situations, such as vehicle accidents, crimes, and sirens, from urban noise [3, 21, 29].

Deep learning has achieved remarkable performance improvements in many classification tasks such as image, pattern, or speech recognition. Deep learning-based urban sound classification research in ESC has also consistently achieved high accuracy [1, 11, 13, 17, 22, 25, 35]. Since urban sounds are collected from multiple sensor nodes placed outdoors, deep learning-based ESC applications typically perform model inference on cloud servers. However, heavy network traffic may be generated during data transmission between the cloud and sensor nodes for computation, and security issues exist because the collected sound data may contain personal information [20].

In recent years, on-device deep learning has gained attention for overcoming the limitations of cloud-based deep learning applications. On-device deep learning performs model inferences on edge devices rather than on a cloud server. Because of minimal data transmission between the cloud and edge devices, on-device deep learning is less affected by network conditions and can effectively handle data security problems. However, since edge devices have limited computing power and memory space, directly applying deep and complex models to edge devices is difficult. Although model compression techniques, such as quantization [32], pruning the redundant weights of the models [14], and knowledge distillation [15], can facilitate the seamless migration of complex models to resource-constrained edge devices, on-device inference without sacrificing the accuracy of the original model is a challenging problem. Additionally, the costs of the measurement equipment for data collection and monitoring systems for noise analysis remains high [19, 26].

To address these problems, we propose a low-cost, on-device urban sound monitoring system using a lightweight end-to-end convolutional neural network (CNN). To this end, we first considered two prospective candidate models, a one-dimensional (1D) CNN model and a two-dimensional (2D) CNN model, as the baseline and compared their performances. Based on the evaluation results, we selected the 1D CNN baseline model and adjusted the sampling rate and model configuration to determine the optimal model architecture. Subsequently, we examined the changes in accuracy, power consumption, and inference speed after model compression using three quantization techniques to select the best technique. Finally, we determined the input audio length, which is a major factor in determining the sound monitoring period. The final lightweight CNN model achieved an accuracy of 94.9% with only 331 K parameters without preliminary feature extraction on our dataset that included emergency sounds and traffic, industrial, and residential noises.

We also demonstrated the actual applicability of the system by constructing a prototype consisting of an edge artificial intelligence (AI) device and a FIWARE-based server that performed real-time data collection, sound context classification, and user-friendly data visualization processes. The prototype was constructed at a cost of less than USD 50 and could perform real-time noise monitoring at 3 s intervals.

The remainder of this paper is organized as follows. In Section 2, we introduce research related to the ESC, deep learning, and on-device deep learning. In Section 3, we suggest the overall configuration of the proposed system and our strategy for developing a lightweight deep learning model. In Section 4, we present the tuning results of the proposed deep learning model and the performance of the system. Finally, in Section 5, we summarize and conclude the study.

2 Related work

Several studies have been conducted to classify environmental noise sources by using ESC. Tanweer *et al.* [28] classified bus, market, and industrial noise using machine learning algorithms. Maijala *et al.* [19] proposed a smart sensor system using Gaussian mixture models and artificial neural networks (ANNs) to classify the noise from a rock-crushing plant. In [30], Tsalera *et al.* classified the noise generated in urban areas using the K-nearest neighbors algorithm.

Studies have also been conducted to detect emergency situations using urban sounds. Padhy *et al.* [21] proposed a multi-channel CNN to recognize emergency situations for hearing-impaired individuals. Tran and Tsai [29] proposed SirenNet that can identify emergency vehicles. Almaadeed *et al.* [3] proposed a support vector machine (SVM)-based road surveillance application to detect car crashes and tire skidding on roads.

Deep learning-based ESC is actively being studied to accurately classify complex environmental noises that contain numerous overlapping sounds. The CNN is the most commonly used deep learning model architecture for the ESC. Early CNN-based ESC research classified sounds using a simple 2D CNN model after feature extraction to obtain time-frequency representations. Piczak [22] used a simple two-layer CNN to classify sounds by extracting sound characteristics using a log-scaled mel-spectrogram. Salamon and Bello [25] proposed methods to increase the accuracy of 2D CNN models by applying various data augmentation techniques. Zhang *et al.* [35] improved the classification accuracy for noisy data by performing feature extraction using a spectrogram energy triggering algorithm.

Recently, studies have been conducted to achieve a higher classification accuracy by applying model ensembles or the latest CNN architectures, such as attention modules and transformer methods. Li *et al.* [17] demonstrated accuracy rates of 92.2% and 92.6% for the UrbanSound8K (US8K) and ESC-10 datasets, respectively, using a deep learning model that combined logmel-CNN and end-to-end raw-CNN through ensemble learning. Guzhov *et al.* [13] proposed a model, called ESResNet, that simultaneously used the attention mechanism and residual network techniques, achieving 97% and 91.5% accuracy on the ESC-10 and 50 datasets, respectively. Gazneli *et al.* [11] presented an end-to-end audio transformer (EAT) model that performed sound classification using a 1D convolution stack and a transformer encoder block. The EAT-S and EAT-M models proposed in this study achieved accuracy rates of 95.25% and 96.3%, respectively, on the ESC-50 dataset. These studies achieved accuracy rates close to 95% accuracy for human-performed ESC-10 data classifications [23]. Despite their high accuracy, these complex models require preliminary feature extraction or have numerous model parameters; hence, directly applying them to on-device deep learning is challenging.

Da Silva *et al.* [8] and Arce *et al.* [4] evaluated the accuracy and execution time of classical machine learning methods and lightweight 2D CNNs on Raspberry Pi 3B devices, respectively. However, both studies required feature extraction for inference and achieved classification accuracies of less than 80%. Wyatt *et al.* [34] and Elliott *et al.* [9] developed transformer-based models applicable to edge devices and tested them on the Raspberry Pi Zero and Samsung Galaxy S9 devices, respectively. These on-device ESC studies with lightweight architectures demonstrated the potential for ESC on edge devices. However, the inference time was excessively long owing to the computational overhead of the log-scaled mel-spectrogram used as the feature extraction algorithm. Overall, further improvements are required in on-device ESC research because the model accuracy is low or additional computation is required for feature extraction.

3 Design of the proposed system architecture

In this section, we describe the proposed system components for the on-device ESC and its development procedure. As shown in Figure 1, the proposed system consists of edge AI nodes with a lightweight deep learning model and a FIWARE-based server. In each sound monitoring period, an edge AI node performs tasks such as recording ambient noise, calculating noise levels, classifying sounds using a lightweight deep learning model, and transmitting the classification results to a FIWARE-based server. The FIWARE-based server stores the sound analysis results in an internal database and visualizes the urban sound monitoring results using these data. Using the proposed system, users can monitor in real-time the types of sound generated around the locations where edge AI nodes are installed.

We also developed a lightweight deep learning model that can perform real-time inference with high accuracy on the edge AI nodes. To evaluate the model in various urban sound contexts, we generated an integrated dataset that included 17 classes of urban sound contexts from four different public datasets. The classification accuracy, number of operations, and size of the model parameters were used as metrics to evaluate both the sound classification ability and computational efficiency of the model trained on our integrated dataset. In addition, post-training quantization (PTQ) was used to accelerate the inference of the developed model.

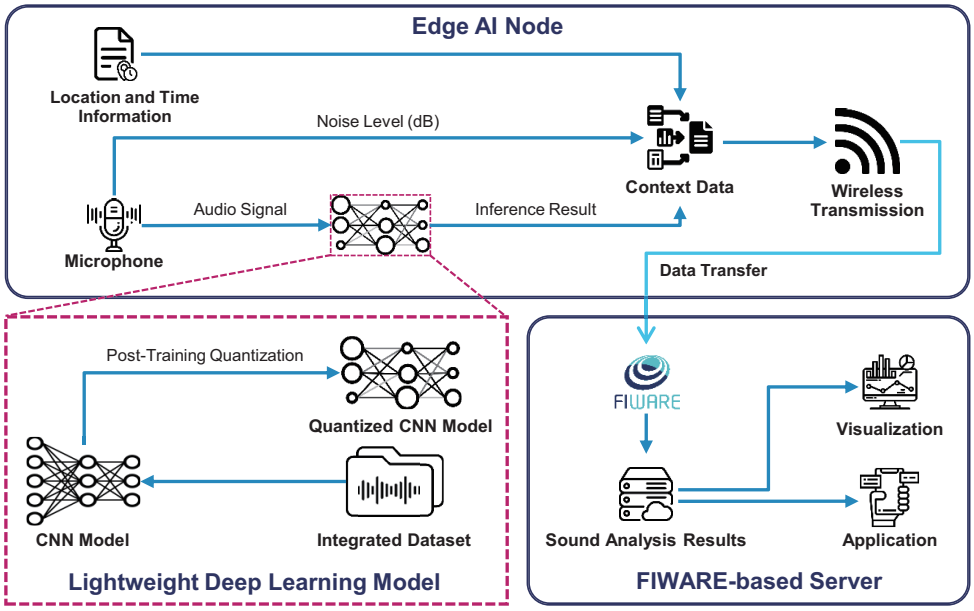


Figure 1: Architecture design of the proposed system.

3.1 Edge AI node

The edge AI node classifies the sounds recorded at its installed location and delivers the classification results to the FIWARE-based server. Since a single edge AI node has a limited coverage area, multiple nodes must be deployed to minimize unmonitored areas. Therefore, the cost-effectiveness of the edge AI nodes is an important factor.

We selected the Raspberry Pi Zero 2W as the target platform for the on-device inference and data processing. Despite its limited resources, the CPU-based edge device allows for a fast inference response for lightweight deep learning models. Furthermore, it is priced at USD 15 with an onboard wireless communication module. Therefore, we constructed a low-cost, low-latency, on-device ESC system using the Raspberry Pi Zero 2W. To reduce the required components of the edge AI node and their cost, we used an omnidirectional microphone that could collect sound from all directions equally well using only one microphone. Although its sensitivity might be lower than that of a directional microphone, it was sufficient for our system to classify relatively loud urban noise.

Since edge AI nodes are generally installed outdoors, we used a waterproofed polyvinyl chloride (PVC) enclosure to protect their internal components. Although an edge AI node is intended to be powered by an external power source, it can operate alternatively with a power bank when external power is unavailable. Even with a power bank, an edge AI node can be built for less than USD 50. Therefore, edge AI nodes can be installed at multiple locations at a modest cost, enabling real-time noise monitoring and analysis without unmonitored spots. A detailed view of an edge AI node and a list of the required components and their costs for constructing an edge AI node are presented in Figure 2 and Table 1, respectively.

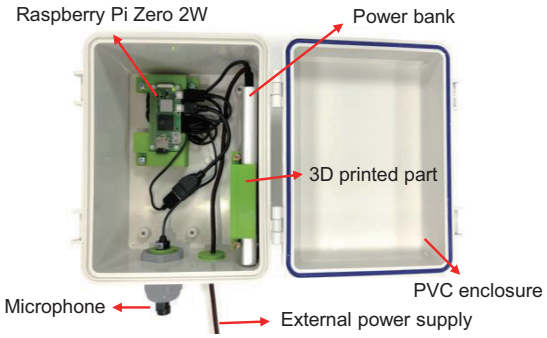


Figure 2: Detailed view of the components inside an edge AI node.

Table 1: Components of an edge AI node.

Component	Specification
Raspberry Pi zero 2W	CPU: 1 GHz quad-core, 64-bit ARM Cortex-A53 Connectivity: 2.4GHz 802.11 b/g/n wireless LAN Size: 65 mm × 30 mm Price: USD 15
Microphone	Frequency range: 100 Hz – 16 KHz Sensitivity: -51dB – -43dB Size: 70 mm × 15 mm Price: USD 8
PVC enclosure	Size: 150 mm × 200 mm × 120 mm Price: USD 4.2
3D printed parts for fastening	Material: PLA Weight: 44 g Price: USD 3.7
Power bank	Battery capacity: 10000 mAh (3.7 V) Output: 5.1 V / 2 A Size: 71.2 mm × 147 mm × 14.2 mm Price: USD 18.83

3.2 FIWARE-based server

FIWARE [7] is an open-source smart city framework used in various smart city applications, such as smart agriculture [18, 36] and smart homes [10]. FIWARE consists of three main parts: core context management, interface to IoT, and context processing. The core context management is responsible for data updates, query handling, and message publishing/subscription management. The interface part, IoT agents, is a middleware that facilitates data transmission between the core context management and sensors or actuators. The IoT agent receives data using common IoT-level protocols such as JSON, HTTP, and MQTT, and converts the data to the NGSI format for transmission to the server or vice versa. IoT agents also enable the existing protocols deployed in low-level sensor and actuator solutions to be easily integrated into FIWARE. The context processing component supports data analysis and visualization for user convenience.

The structure of the proposed FIWARE-based server is illustrated in Figure 3. We utilized the Orion Context Broker (OCB) to build the FIWARE server and manage the historical data using QuantumLeap and CrateDB. The data stored in CrateDB were visualized using Grafana [37] for monitoring systems.

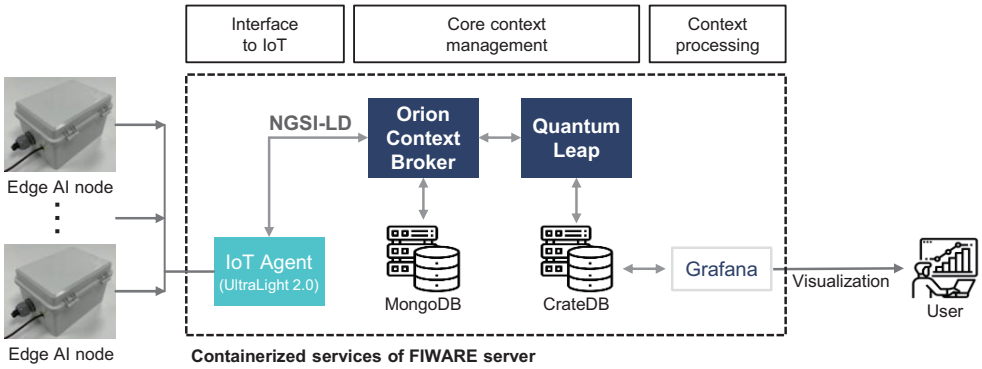


Figure 3: FIWARE-based server architecture for the proposed system.

3.3 Deep learning model development

To effectively monitor urban sounds, a lightweight deep learning model should be capable of classifying a wide range of sounds, including emergency-related sounds and traffic, industrial, and residential noises. A single public ESC dataset cannot cover all the classes of urban sounds because of insufficient data. Therefore, we constructed an integrated dataset of urban sound-related data from US8K [24] and ESC-50 [23], nonverbal sound data from the AI-Hub (Korea) [38], and urban sound data from the AI-Hub (Korea) [40] for model training. Table 2 presents the origin of the data for each class as well as the number of audio clips for each class. We divided the dataset into 80%:10%:10% training, validation, and test datasets, respectively.

Table 2: Summary of the dataset for deep learning model development.

Category	Class	Data source				Number of audio clips
		US8K	ESC-50	Nonverbal data	Urban Sound data	
Traffic noise	car	-	-	-	1639	1639
	motorcycle	-	-	-	4206	4206
	horn	236	40	-	7171	7447
	train	-	40	-	2085	2125
	airplane	-	40	-	1761	1801
	helicopter	-	40	-	3225	3265
Industrial noise	pile driver	-	-	-	3169	3169
	generator	-	-	-	2286	2286
	drilling	871	-	-	-	871
	compressor	-	-	-	2077	2077
	jackhammer	894	-	-	-	894
Residential noise	dog	752	40	-	2069	2861
	cat	-	40	-	1995	2035
Emergency sound	gunshot	113	-	3016	-	3129
	siren	904	40	-	1912	2856
	screaming	-	-	509	-	509
	crying	-	-	1432	-	1432
Number of audio clips		3770	280	4957	33595	42602
Total duration		4.06 Hours	0.39 Hours	4.13 Hours	66.66 Hours	75.24 Hours
Average length of each clip		3.88 s	5.00 s	3.00 s	7.14 s	-

Since the integrated dataset used for model training in this study was imbalanced, learning the features of classes using a small number of samples could be difficult. To address this problem, data augmentation was required for the classes that constituted a small portion of the dataset. We employed the sliding window technique illustrated in Figure 4 as the augmentation technique, which is commonly used for audio data. By setting various stride sizes, this technique could generate multiple new data samples from the original audio signal with a more balanced class distribution. In this approach, a smaller stride size resulted in a larger overlap between adjacent samples, resulting in more samples

being obtained. Figure 5 shows the total audio length by class before and after data augmentation and the stride ratio set differently according to the original sound sample length for augmentation. These results indicated that data augmentation led to a more balanced distribution of audio samples per class.

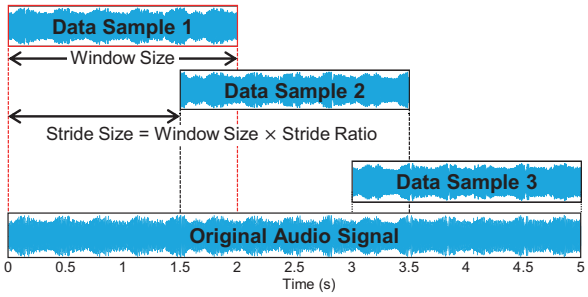


Figure 4: Data augmentation based on the sliding window technique.

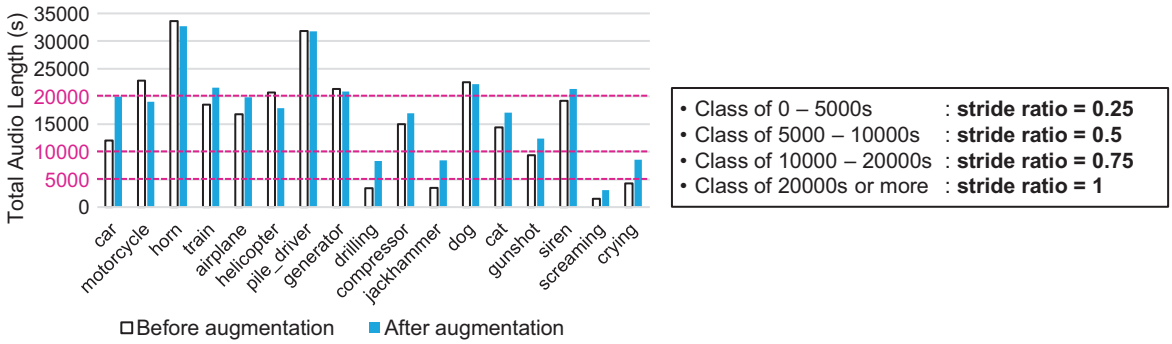


Figure 5: Comparison of data class distribution in the integrated dataset before and after data augmentation.

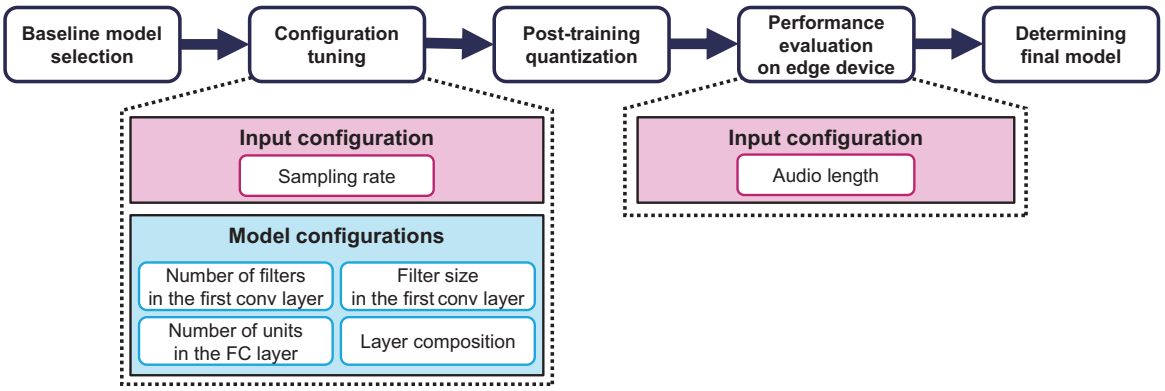


Figure 6: Procedure for building our lightweight deep learning model architecture.

Figure 6 depicts the model selection process for constructing a lightweight deep learning model with high accuracy for real-time inference on edge AI nodes. First, we examined the accuracy, number of parameters, and computational cost of the two prospective baseline models, and selected the best model based on the comparison results. Next, we adjusted the input sampling rate and model configuration of the selected baseline model to minimize the number of parameters and operations. We used PTQ to reduce the model size while maintaining the inference performance on edge AI nodes. For the quantized model, we conducted experiments on various input audio lengths to select a sufficiently short input audio length for emergency detection while achieving a high noise classification performance. Finally, we deployed the final lightweight deep learning model on our prototype edge AI node and verified its operation.

We selected the 1D CNN model proposed by Abdoli *et al.* [1] and the 2D CNN model proposed by Salamon and Bello [25] as prospective baseline models for our on-device ESC system because they have a relatively small number of model parameters. Table 3 lists the properties of the two baseline models.

Table 3: Properties of candidate 1D CNN and 2D CNN models.

	1D CNN [1]	**2D CNN [25]**
Input audio length	1 s	3 s
Input shape	16000 × 1	128 × 128
Sampling rate	16 KHz	Original sampling rate
#Params	257 K	241 K
Optimizer	Adadelta	Adam
Feature extraction	N/A (Raw signals)	Log-scaled mel-spectrogram

To determine the final baseline model after comparing the performances of the candidate baseline models under the same conditions, we evaluated and compared their performances in terms of classification accuracy, number of model parameters, and number of operations required in mega floating-point operations (MFLOPs) for different input audio lengths. The evaluation results are listed in Table 4.

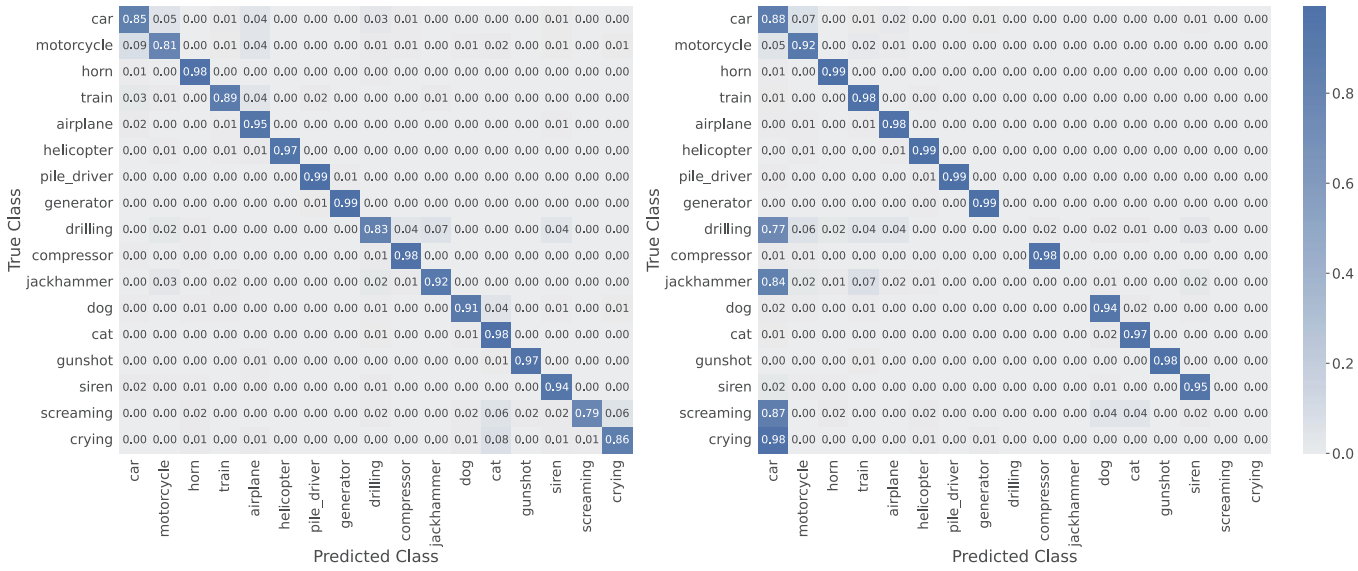
Table 4: Comparison results of candidate baseline models with varying input audio lengths.

		Input audio length				
		1 s	1.5 s	2 s	2.5 s	3 s
1D CNN	Input Size	16000	24000	32000	40000	48000
	Test Accuracy	89.57%	91.10%	92.40%	93.49%	93.68%
	#Params	257,473	388,545	519,617	650,689	765,377
	MFLOPs	35.57	54.54	73.57	92.61	111.42
2D CNN	Input Size	128×128	128×128	128×128	128×128	128×128
	Test Accuracy	58.96%	82.14%	83.15%	90.31%	90.92%
	#Params	241,889	241,889	241,889	241,889	241,889
	MFLOPs	115.60	115.60	115.60	115.60	115.60

As presented in Table 4, the 1D CNN model exhibited a higher classification accuracy than the 2D CNN model for the same input audio length. Furthermore, the normalized confusion matrices, as shown in Figure 7, indicated that the 2D CNN had considerably low accuracy in certain classes with a small amount of data, such as drilling, jackhammer, screaming, and crying, whereas the 1D CNN accurately classified all the class data. Since the 2D CNN model always resizes the input to 128×128 using a log-scaled mel-spectrogram, the number of parameters and computational demand were the same irrespective of the input audio length. By contrast, the number of parameters and computational demand of the 1D CNN increased linearly with the input audio length because it used the input data directly without feature extraction. However, the computational demand of the 1D CNN was lower than that of the 2D CNN for all input audio lengths, as shown in Table 4. Additionally, although the 1D CNN model had more parameters than the 2D CNN model, the number of model parameters remained less than one million even for the 3 s input audio length, sufficient to meet the tight memory and computing resource constraints of edge devices. Therefore, we selected the 1D CNN model as the final baseline model.

Larger and deeper models are generally more accurate; however, resource constraints limit their practical implementation on edge devices. Therefore, in Section 4, we further elaborated the model structure by experimenting with the configuration items shown in Figure 6 to achieve high classification accuracy while maintaining low computational costs.

After fine-tuning the different model configuration options to enhance the accuracy of the deep learning model, we used PTQ to compress the model for lightweight edge devices. PTQ quantizes pre-trained deep learning models by reducing the number of bits used to express each weight and activation value. Although this technique may result in accuracy degradation owing to rounding errors during quantization and dequantization processes, it can provide several benefits, including a reduced memory footprint, latency, and improved throughput [2]. Figure 8 illustrates the process of



(a) 1D CNN with 3 s of audio

(b) 2D CNN with 3 s of audio

Figure 7: Normalized confusion matrices of 1D CNN and 2D CNN.

mapping a 32-bit real value (x) in the range of $[x_{min}, x_{max}]$ to a B-bit quantized value (x_q) in the range of $[-2^{B-1}, 2^{B-1}-1]$. The quantization transformation is expressed as follows [33]:

$$S = \frac{2^B - 1}{x_{max} - x_{min}} \tag{1}$$

$$Z = -round(x_{min} \times S) - 2^{B-1} \tag{2}$$

where S and Z denote the scale factor and the zero-point, respectively.

$$x_q = S \times x + Z \tag{3}$$

$$x = \frac{1}{S}(x_q - Z) \tag{4}$$

Equations 3 and 4 represent the quantization and dequantization functions, respectively. Additional processing overhead may arise during the scale factor and zero-point calculations, as well as the quantization and dequantization operations.

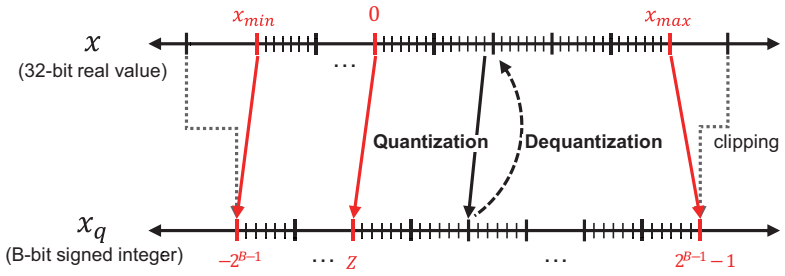


Figure 8: Quantization mapping process between the 32-bit real value and the B-bit signed integer.

To analyze the inference speed, power consumption, and accuracy of the model for an actual system deployment, we measured the performance of the model after PTQ on a prototype edge device. We determined the final deep learning model structure for deployment in the proposed system based on the experimental results obtained from the prototype. In Section 4, we present a detailed discussion of the entire model optimization procedure after the baseline model setup.

4 Model tuning and performance evaluation on edge devices

In this section, we analyze the effects of model tuning and quantization options on the performance of the deep learning model, and verify the operation of the proposed system. The model was tuned in the following order: input sampling rate adjustment, model configuration tuning, quantization, and input audio length selection. We determine the input audio length in the final design stage because it had a significant impact not only on the model accuracy and inference speed but also on the monitoring interval of the system.

We implemented the model and training process using the TensorFlow framework and deployed the trained model on an edge device using the TensorFlow Lite framework. Table 5 summarizes the model training environment. We used a batch size of 32 and 100 epochs, and an Adadelta optimizer with an initial learning rate of 1.0 for all training. The inference performance of the edge device was measured using tflite-runtime 2.10.0 on a Raspberry Pi Zero 2W.

Table 5: Hardware and software specifications for deep learning model development.

Type	Specification
OS	Ubuntu 20.04
CPU	Intel Xeon ® W-2235
RAM	32GB
GPU	NVIDIA RTX A4000 (16GB memory)
CUDA version	11.3
CUDNN version	8.2.1
TensorFlow version	2.10.0

4.1 Input sampling rate adjustment

Increasing the input data size for the baseline 1D CNN improved the classification accuracy, as described in Section 3.3. However, this increased the computational cost. Therefore, we intended to reduce the computational cost by adjusting the input size, that is, by downsampling the input audio data. Hence, we compared the accuracy and computational cost of the model at the sampling rates of 4, 8, and 16 KHz. In the comparison, we did not consider the input audio length and sampling rate combinations that resulted in an input data size of less than 8000 because convolution operations cannot be performed when the input size for convolution is smaller than the filter size.

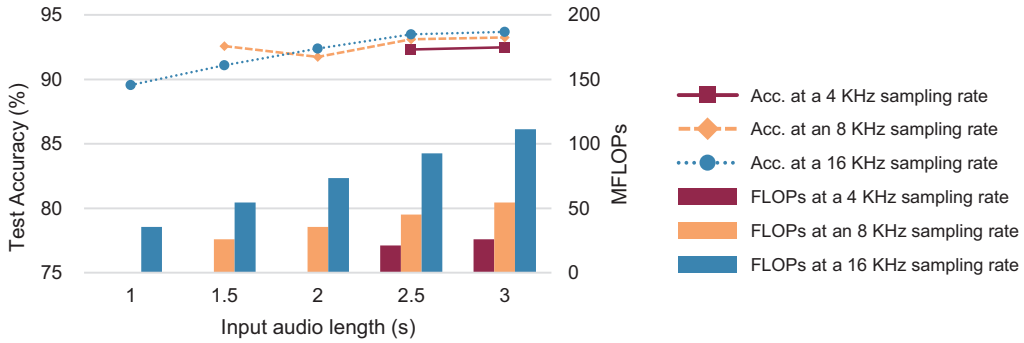


Figure 9: Accuracy and FLOPs of the baseline model for varying input audio lengths and sampling rates.

Figure 9 shows the accuracy and computational cost represented as MFLOPs, for each input audio length and sampling rate of the baseline 1D CNN. When the input audio length was fixed, a higher sampling rate increased both the accuracy and computational cost. According to the results shown in Figure 9, the changes in the sampling rate had a greater impact on the computational cost than on the accuracy. Therefore, employing a lower sampling rate may be more beneficial in terms of the hardware resource utilization of the edge platform. However, utilizing a 4 KHz sampling rate decreased the classification accuracy at input audio lengths of 2.5 s and 3 s by 1–3%p compared to using an

8 or 16 KHz sampling rate because an accurate feature of the audio data could not be extracted. Furthermore, if the input audio length was less than 2 s, using a 4 KHz sampling rate required an overall change in the model structure; therefore, we decided not to use a 4 KHz sampling rate.

The average accuracies of the sampling rates of 8 KHz and 16 KHz were nearly identical. Since using 8 KHz results in an approximately 2× reduction in the computational cost compared to using 16 KHz, we selected a sampling rate of 8 KHz for the proposed system.

4.2 Model configuration tuning

Subsequently, we adjusted the configuration of the baseline model with the input sampling rate set to 8 KHz, as previously determined. The model configuration items targeted for tuning were the number of filters and their size in the first convolutional layer, the number of units in the fully connected (FC) layer, and the composition of the convolutional layers that did not significantly alter the computational cost of the baseline model. To assess the impact of each item on the performance, we measured the accuracy of the model by combining various numbers of filters (8, 16, 24) and filter sizes (32, 64, 96) for the first convolutional layer, as well as different numbers of units (64-32, 128-64, 192-96) for the FC1-FC2 layers. The default configuration of the baseline model consisted of 16 filters of size 64 for the first convolutional layer and 128-64 units for the FC1-FC2 layers.

In addition, we evaluated the accuracy of two different convolution layer compositions. Figure 10 shows a comparison of the structures. Figure 10 (a) shows the most frequently used compositions in the sequence of convolution, batch normalization, and activation [16] (BN-first), which adjusts the distribution of data before running a non-linear activation function. By contrast, the composition illustrated in Figure 10 (b) executes batch normalization after a non-linear activation function following the convolution operation (ReLU-first). We noted that the structures of these layer compositions did not affect the number of model parameters or computational cost.

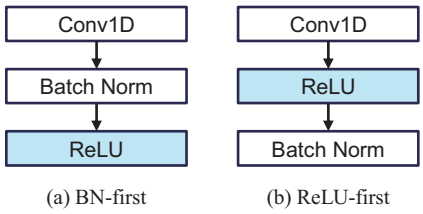
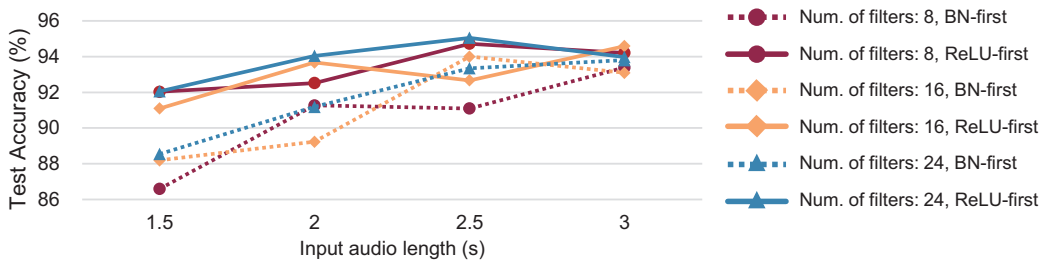


Figure 10: Layer composition for model architecture tuning.

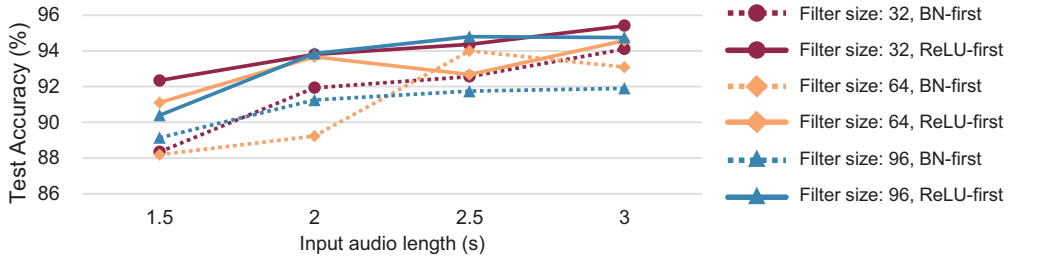
Figures 11 (a)–(c) show the classification accuracy based on the number of filters in the first convolutional layer, filter size in the first convolutional layer, and the number of units in the FC layer, respectively. The results indicated that the ReLU-first composition consistently achieved notably higher accuracy than the BN-first composition in most cases. The effects of the other configuration options, as depicted in Figure 11, on the model accuracy were not significant when the layer composition was set to ReLU-first. By contrast, as the input audio length increased, the accuracy of the model tended to increase, irrespective of the model configurations. Therefore, we selected the model configuration with the highest accuracy for each input audio length and determined the final input audio length after the subsequent model quantization step. Table 6 lists the accuracy, number of parameters, and computational costs of the best model configuration for each audio length.

Table 6: Model configurations achieving the best accuracy for each input audio length.

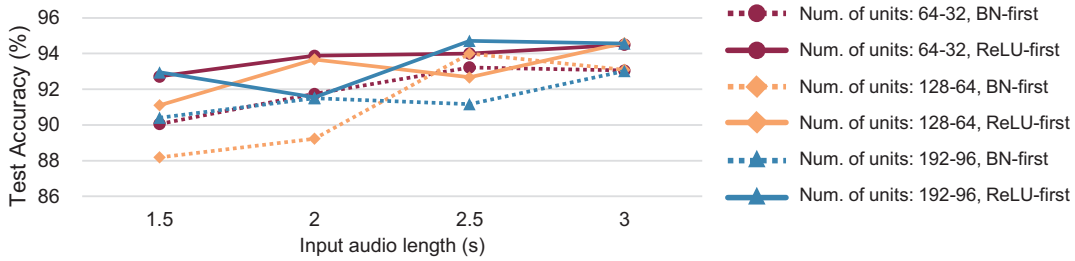
Input audio length		1.5 s	2 s	2.5 s	3 s
Model configurations	# of filters in the first conv layer	16	24	24	16
	Filter size in the first conv layer	64	64	64	32
	# of units in FC1-FC2 layers	192-96	128-64	128-64	128-64
	Layer composition	ReLU-first	ReLU-first	ReLU-first	ReLU-first
# of parameters		235.6 K	266.2 K	331.8 K	388.0 K
MFLOPs		26.13	51.89	65.57	42.33
Accuracy		92.95%	94.04%	95.05%	95.41%



(a) Accuracy based on the number of filters in the first convolutional layer and layer composition



(b) Accuracy based on the filter size in the first convolutional layer and layer composition



(c) Accuracy based on the number of units in FC layers and layer composition

Figure 11: Accuracy comparison of different model configurations.

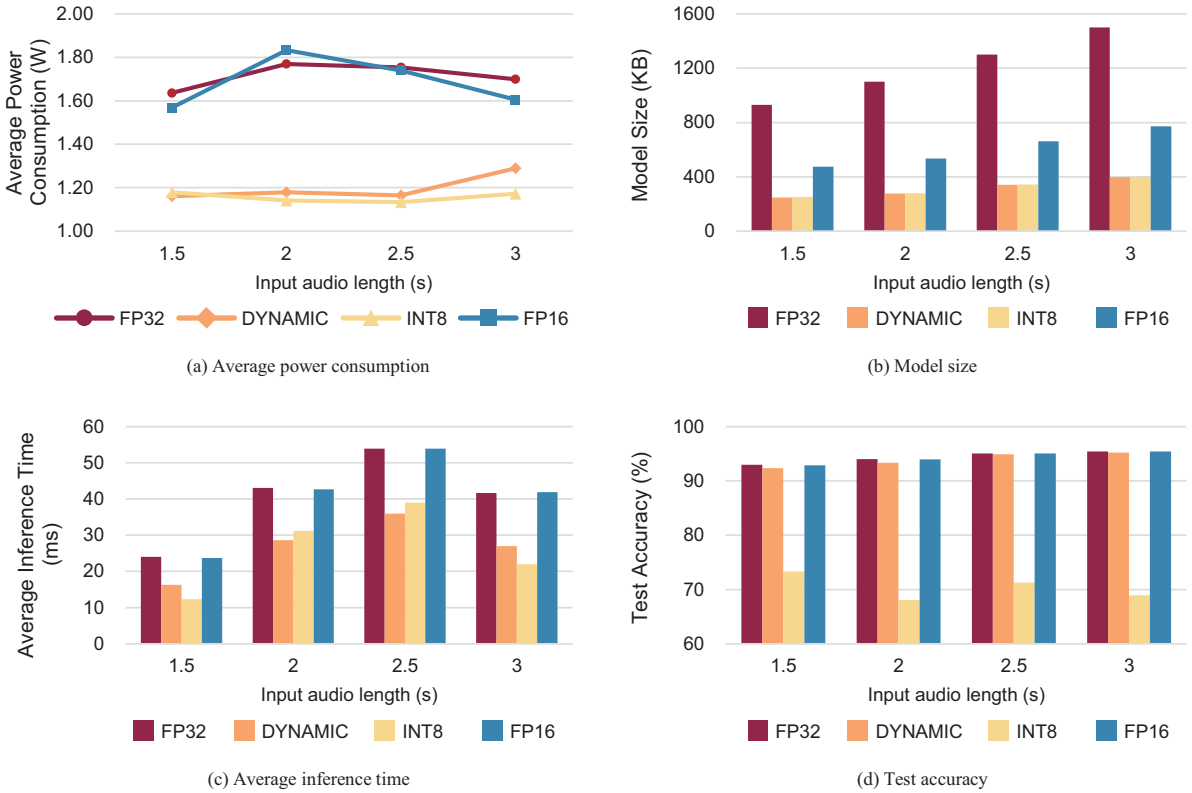
4.3 Quantization and input audio length selection

To reduce inference time, we applied three PTQ techniques, as presented in Table 7. We used the power consumption, model size, accuracy, and inference time as the performance metrics to determine the best PTQ technique. Figure 12 shows the performance comparison results of the different quantization techniques for the Raspberry Pi Zero 2W. In this study, we referred to the models without quantization as FP32, and those with dynamic range, full integer, and float16 quantization as DYNAMIC, INT8, and FP16, respectively.

Table 7: Post-training quantization techniques provided by TFLite [39].

Quantization Technique	Benefits	Hardware	Description
Dynamic range	4x smaller 2x–3x speedup	CPU	- Quantize the model weights to 8-bit integer - At inference, model performs operations that mix integer and float computation.
Full integer	4x smaller 3x+ speedup	CPU, Edge TPU	- Quantize the model weights, input, activations, and output to 8-bit integer - To calibrate model input/output and activations, a representative dataset is needed.
Float16	2x smaller GPU acceleration	CPU, GPU	- Quantize the model weights to float16 - At inference, model performs float32 operations.

Figures 12 (a) and (b) show the power consumption measured by the USB power meter during the inference and the size of each model, respectively. According to the results, the INT8 and DYNAMIC models had smaller model sizes and exhibited significantly lower power consumption than the FP32 and FP16 models. As listed in Table 7, the FP16 model was half the size of the FP32 model, whereas the DYNAMIC and INT8 models were approximately four times smaller. Figure 12 (c) indicates that the INT8 and DYNAMIC models had faster inference times of approximately 12–20 ms than the FP32 and FP16 models. However, as shown in Figure 12 (d), the INT8 model exhibited 19–26%p lower inference accuracy than the other models, despite the advantages of low power consumption and small



(a) Average power consumption

(b) Model size

(c) Average inference time

(d) Test accuracy

Figure 12: Performance comparison of post-training quantization techniques on a Raspberry Pi Zero 2W.

memory requirements, making it unsuitable for our system. The accuracy did not significantly differ among the three models, except for the INT8. Consequently, we decided to use the DYNAMIC model in the prototype system that had a smaller model size and lower power consumption than the FP32 and FP16 models while maintaining high accuracy.

After determining the PTQ technique, we investigated the most appropriate input audio length for our system by comparing the accuracy and inference time of the dynamic range quantized models for various input audio lengths. Figure 13 shows the comparison results of the accuracy and inference time for different input audio lengths. As the input audio length increased, the inference accuracy of the model increased; however, the time required to record the sound increased, resulting in a longer sound monitoring period. On the other hand, shorter input audio lengths can reduce the sound monitoring period while decreasing the inference accuracy. Therefore, among the four input audio lengths tested, we selected an input audio length of 2.5 s because it allowed us to shorten the monitoring period while providing fairly high accuracy. Figure 14 shows the final architecture of the proposed lightweight deep

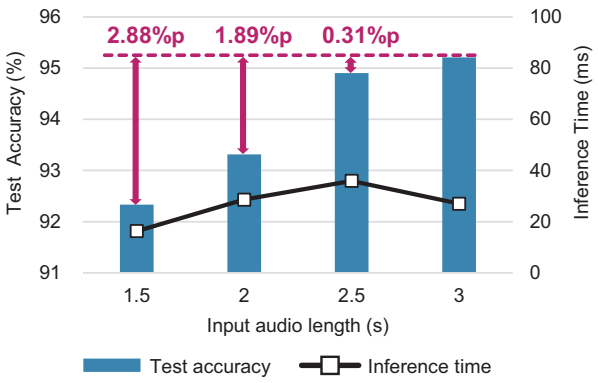


Figure 13: Accuracy and inference time of the dynamic range quantization model.

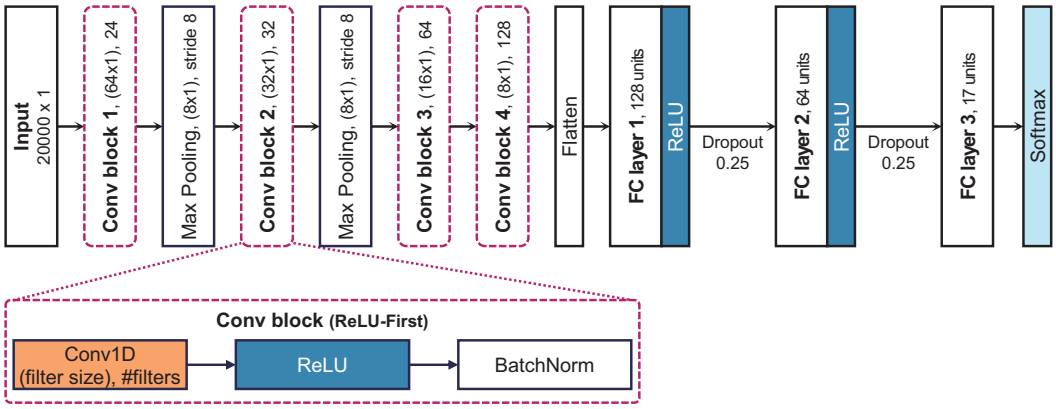


Figure 14: Final lightweight end-to-end CNN architecture for the proposed system.

learning model for urban sound classification. The proposed model is an end-to-end CNN that uses 2.5 s of audio data recorded at a sampling rate of 8 KHz without feature extraction.

4.4 Verification of the proposed system operation

We validated the performance and operation of the proposed system on a prototype consisting of an edge AI node that performed inference using the finalized model and a FIWARE-based server constructed on a high-end desktop computer. The entire processing flow in the edge AI node comprised six tasks: sound recording and saving in the WAV file format, loading the audio file, model inference, sound level calculation, data encapsulation for the FIWARE protocol, and data transfer to a FIWARE-based server.

Table 8: Time consumption of tasks measured on the prototype edge AI node.

Task	Avg.	Std. Dev.	Min.	Max.
Sound recording	2639.02 ms (96.06%)	4.83	2629.36 ms	2650.24 ms
WAV file loading	3.55 ms (0.13%)	0.26	3.03 ms	4.83 ms
Model inference	52.68 ms (1.92%)	5.79	38.41 ms	62.44 ms
Decibel calculation	4 ms (0.14%)	0.55	3.53 ms	5.52 ms
Data encapsulation	0.31 ms (0.01%)	0.15	0.24 ms	1.06 ms
Data transfer	47.71 ms (1.74%)	9.85	35.19 ms	83.99 ms
Total processing time	2747.27 ms (100%)	-	2709.76 ms	2808.08 ms

Table 8 lists the average, standard deviation, and minimum and maximum times required for each task measured 100 times at the edge AI node. As presented in Table 8, most of the time was spent recording and saving the acquired external sound into a WAV format file, whereas relatively little time was required to load the saved WAV file, calculate the sound level in decibels, and process the data for transferring to the FIWARE-based server.

Owing to our proposed lightweight deep learning model, the average inference time was 52.68 ms that occupied a very low proportion of the monitoring cycle. At the end of each monitoring period, the inference results, expressed as confidence scores for each class, and the decibel values of the recorded data were transferred to the FIWARE-based server over Wi-Fi at a frequency of 2.462 GHz and a bit rate of 72.2 MB/s. The size of data transferred to the server per monitoring cycle was approximately 500 bytes, and the data transmission took an average of 47.71 ms and a maximum of 83.99 ms. As the network channel environment affects the data transfer speed, the standard deviation of the data transfer time was the highest among all tasks; however, in all tests, we verified that the data were delivered to the FIWARE-based server without any loss.

Overall, the results in Table 8 demonstrated that the proposed system could achieve a monitoring period of approximately 3 s on the edge AI node, even in the worst-case case. Furthermore, the sound monitoring period of 3 s considerably shorter than that of previous edge device-based ESC studies [4, 34] that had a monitoring period of 5–6 s, excluding the time required for data transfer.

We also implemented a data visualization function on a FIWARE-based server for real-time decision-making in urban noise situations and emergencies. To verify the accuracy of the data visualization function, we played AudioSet [12] data that were not used for model training and testing near the installed edge AI node, and displayed the monitoring results on the Grafana dashboard of the FIWARE-based server.

Figure 15 illustrates the visualization results when the gunshot, drilling, and siren sounds were played for 4 minutes each. The top dashboard shows the confidence scores of the inference results, where the highest confidence scores of the class correspond to the given noise. The bottom dashboard in Figure 15 shows the decibels transmitted from the edge AI node. Since small sounds are likely to be background noises that need not be classified, the inference results were sent to the FIWARE-based server only when the recorded sound exceeded the predefined threshold decibel value. The World Health Organization (WHO) recommends that the average noise level should not exceed 40–55 decibels per hour to protect human health [31]. Based on this recommendation, we set the threshold to 40 dB. The visualization results, as shown in Figure 15, confirmed that the inference results were not delivered during silent intervals.

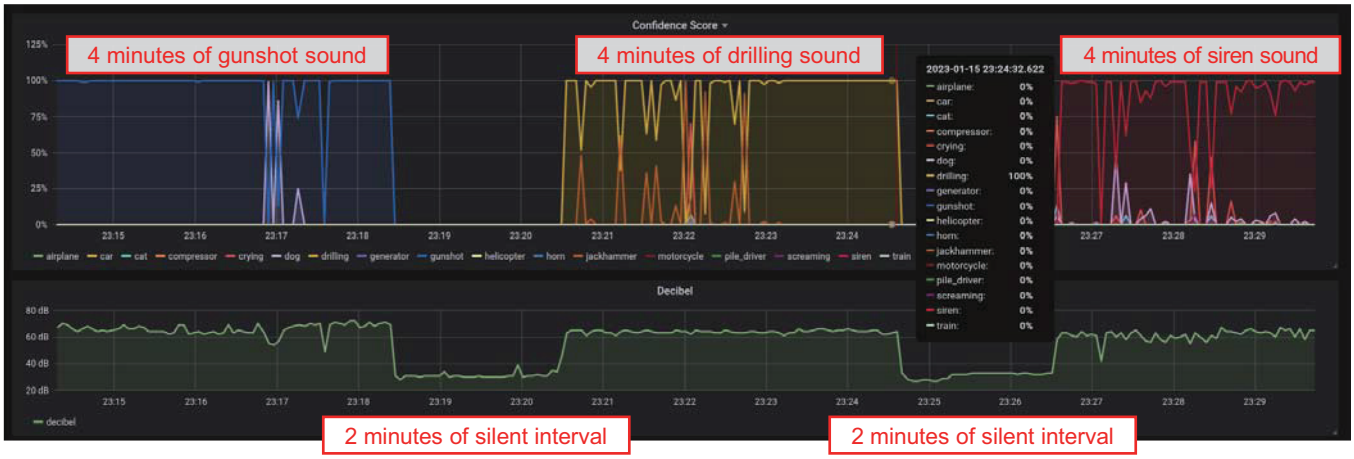


Figure 15: Real-time visualization results of urban sound monitoring and emergency detection on the prototype FIWARE-based server.

5 Conclusions

In this study, we developed a deep learning-based on-device sound monitoring system capable of categorizing and analyzing urban sounds in real-time. We first built an integrated dataset from four publicly available datasets to classify various types of urban sounds and detect emergencies. To develop a lightweight deep learning model for real-time inference on an edge AI node with tight hardware resource constraints, we trained the model on the integrated dataset and adjusted the audio input and model configurations through extensive performance evaluation to maintain higher accuracy while reducing the number of parameters.

Consequently, the proposed model showed a classification accuracy of 94.9% using only 331 K parameters on a prototype edge AI node built on a low-cost embedded platform, such as Raspberry Pi Zero 2W. Because of the low computational complexity of our model, the inference latency on our prototype edge AI node was only 62.44 ms in the worst case. Thus, we achieved a sound monitoring interval of 3 s that was sufficiently short to recognize urgent urban sounds. We also demonstrated that the contexts of sounds acquired from the edge AI node were accurately visualized on a FIWARE-based server for high-level decision-making on the detected sounds.

Funding

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1F1A1060231), and in part by the Korea Ministry

of Land, Infrastructure and Transport (MOLIT) as “Innovative Talent Education Program for Smart City”.

Conflict of interest

The authors declare no conflict of interest.

Acknowledgment

This study partly used datasets from “The Open AI Dataset Project (AI-Hub, Korea)”. All data information from the AI-Hub can be accessed at www.AIhub.or.kr.

References

[1] Abdoli, S.; Cardinal, P.; Koerich, A.L. (2019). End-to-end environmental sound classification using a 1D convolutional neural network, *Expert Systems with Applications*, 136, 252–263, 2019.

[2] Ahn, H.; Chen, T.; Alnaasan, N.; Shafi, A.; Abduljabbar, M.; Subramoni, H.; Panda, D.K. (2023). Performance characterization of using quantization for DNN inference on edge devices: Extended version, arXiv:2303.05016, 2023.

[3] Almaadeed, N.; Asim, M.; Al-Maadeed, S.; Bouridane, A.; Beghdadi, A. (2018). Automatic detection and classification of audio events for road surveillance applications, *Sensors*, 18(6), 1858, 2018.

[4] Arce, P.; Salvo, D.; Piñero, G.; Gonzalez, A. (2021). FIWARE based low-cost wireless acoustic sensor network for monitoring and classification of urban soundscape, *Computer Networks*, 196, 108199, 2021.

[5] Asdrubali, F.; D’Alessandro, F. (2018). Innovative approaches for noise management in smart cities: A review, *Current Pollution Reports*, 4(2), 143–153, 2018.

[6] Chachada, S.; Kuo, C.-C.J. (2014). Environmental sound recognition: A survey, *APSIPA Transactions on Signal and Information Processing*, 3, e14, 2014.

[7] Cirillo, F.; Solmaz, G.; Berz, E.L.; Bauer, M.; Cheng, B.; Kovacs, E. (2019). A standard-based open source IoT platform: FIWARE, *IEEE Internet of Things Magazine*, 2(3), 12–18, 2019.

[8] da Silva, B.; Happi, A.W.; Braeken, A.; Touhafi, A. (2019). Evaluation of classical machine learning techniques towards urban sound recognition on embedded systems, *Applied Sciences*, 9(18), 3885, 2019.

[9] Elliott, D.; Otero, C.E.; Wyatt, S.; Martino, E. (2021). Tiny transformers for environmental sound classification at the edge, arXiv:2103.12157, 2021.

[10] Fazio, M.; Celesti, A.; Márquez, F.G.; Glikson, A.; Villari, M. (2015). Exploiting the FIWARE cloud platform to develop a remote patient monitoring system, *2015 IEEE Symposium on Computers and Communications (ISCC)*, 264–270, 2015.

[11] Gazneli, A.; Zimerman, G.; Ridnik, T.; Sharir, G.; Noy, A. (2022). End-to-end audio strikes back: Boosting augmentations towards an efficient audio classification network, arXiv:2204.11479, 2022.

[12] Gemmeke, J.F.; Ellis, D.P.W.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R.C.; Plakal, M.; Ritter, M. (2017). Audio Set: An ontology and human-labeled dataset for audio events, *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 776–780, 2017.

[13] Guzhov, A.; Raue, F.; Hees, J.; Dengel, A. (2021). ESResNet: Environmental sound classification based on visual domain models, *Proceedings of 2020 25th International Conference on Pattern Recognition (ICPR)*, 4933–4940, 2021.

[14] Han, S.; Pool, J.; Tran, J.; Dally, W. (2015). Learning both weights and connections for efficient neural network, *Advances in Neural Information Processing Systems (NIPS)*, 28, 2015.

[15] Hinton, G.; Vinyals, O.; Dean, J. (2015). Distilling the knowledge in a neural network, arXiv:1503.02531, 2015.

[16] Ioffe, S.; Szegedy, C. (2015). Batch Normalization: Accelerating deep network training by reducing internal covariate shift, *Proceedings of the 32nd International Conference on Machine Learning*, 37, 448–456, 2015.

[17] Li, S.; Yao, Y.; Hu, J.; Liu, G.; Yao, X.; Hu, J. (2018). An ensemble stacked convolutional neural network model for environmental event sound recognition, *Applied Sciences*, 8(7), 1152, 2018.

[18] López-Riquelme, J.A.; Pavón-Pulido, N.; Navarro-Hellín, H.; Soto-Valles, F.; Torres-Sánchez, R. (2017). A software architecture based on FIWARE cloud for precision agriculture, *Agricultural Water Management*, 182, 123–135, 2017.

[19] Maijala, P.; Shuyang, Z.; Heittola, T.; Virtanen, T. (2018). Environmental noise monitoring using source classification in sensors, *Applied Acoustics*, 129, 258–267, 2018.

[20] Murshed, M.G.S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. (2021). Machine learning at the network edge: A survey, *ACM Computing Surveys*, 54(8), 1–37, 2021.

[21] Padhy, S.; Tiwari, J.; Rathore, S.; Kumar, N. (2019). Emergency signal classification for the hearing impaired using multi-channel convolutional neural network architecture, *Proceedings of 2019 IEEE Conference on Information and Communication Technology*, 1–6, 2019.

[22] Piczak, K.J. (2015). Environmental sound classification with convolutional neural networks, *Proceedings of 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6, 2015.

[23] Piczak, K.J. (2015). ESC: Dataset for environmental sound classification, *Proceedings of the 23rd ACM International Conference on Multimedia*, 1015–1018, 2015.

[24] Salamon, J.; Jacoby, C.; Bello, J.P. (2014). A dataset and taxonomy for urban sound research, *Proceedings of the 22nd ACM International Conference on Multimedia*, 1041–1044, 2014.

[25] Salamon, J.; Bello, J.P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification, *IEEE Signal Processing Letters*, 24(3), 279–283, 2017.

[26] Segura-Garcia, J.; Felici-Castell, S.; Perez-Solano, J.J.; Cobos, M.; Navarro, J.M. (2015). Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks, *IEEE Sensors Journal*, 15(2), 836–844, 2015.

[27] Stansfeld, S.A.; Matheson M.P. (2003). Noise pollution: Non-auditory effects on health, *British Medical Bulletin*, 68(1), 243–257, 2003.

[28] Tanweer, S.; Mobin, A.; Alam, A. (2016). Environmental noise classification using LDA, QDA, and ANN methods, *Indian Journal of Science and Technology*, 9(33), 1–8, 2016.

[29] Tran, V.-T.; Tsai, W.-H. (2020). Acoustic-based emergency vehicle detection using convolutional neural networks, *IEEE Access*, 8, 75702–75713, 2020.

[30] Tsalera, E.; Papadakis, A.; Samarakou, M. (2020). Monitoring, profiling, and classification of urban environmental noise using sound characteristics and the KNN algorithm, *Energy Reports*, 6, 223–230, 2020.

[31] World Health Organization (2018). *Environmental Noise Guidelines for the European Region*, World Health Organization. Regional Office for Europe, 2018.

[32] Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; Cheng, J. (2016). Quantized convolutional neural networks for mobile devices, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4820–4828, 2016.

[33] Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation, arXiv:2004.09602, 2020.

[34] Wyatt, S.; Elliott, D.; Aravamudan, A.; Otero, C.E.; Otero, L.D.; Anagnostopoulos, G.C.; Smith, A.O.; Peter, A.M.; Jones, W.; Leung, S.; Lam, E. (2021). Environmental sound classification with tiny transformers in noisy edge environments, *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 309–314, 2021.

[35] Zhang, H.; McLoughlin, I.; Song, Y. (2015). Robust sound event recognition using convolutional neural networks, *Proceedings of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 559–563, 2015.

[36] Zyrianoff, I.; Heideker, A.; Silva, D.; Kamienski, C. (2018). Scalability of an Internet of Things platform for smart water management for agriculture, *2018 23rd Conference of Open Innovations Association (FRUCT)*, 432–439, 2018.

[37] Grafana, [Online]. Available: https://grafana.com, Accessed on 23 June 2023.

[38] Natural and Artificial Occurrence Nonverbal Sound Datasets, [Online]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=644, Accessed on 23 June 2023.

[39] Post-training quantization, [Online]. Available: https://www.tensorflow.org/lite/performance/post_training_quantization#optimization_methods, Accessed on 23 June 2023.

[40] Urban Sound Dataset, [Online]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=585, Accessed on 23 June 2023.

Cite this paper as:

Park, J.; Yoo, T.; Lee, S.; Kim, T. (2023). Urban Noise Analysis and Emergency Detection System using Lightweight End-to-End Convolutional Neural Network, *International Journal of Computers Communications & Control*, 18(5), 5814, 2023.

https://doi.org/10.15837/ijccc.2023.5.5814