sensors

MDPI

Article

LPAI—A Complete AIoT Framework Based on LPWAN Applicable to Acoustic Scene Classification Scenarios

Xinru Jing [1,2], Xin Tian [1] and Chong Du [1,*]

[1] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: duc@sari.ac.cn

Abstract: Deploying artificial intelligence on edge nodes of Low-Power Wide Area Networks can significantly reduce network transmission volumes, event response latency, and overall network power consumption. However, the edge nodes in LPWAN bear limited computing power and storage space, and researchers have found it challenging to improve the recognition capability of the nodes using sensor data from the environment. In particular, the domain-shift problem in LPWAN is challenging to overcome. In this paper, a complete AIoT system framework referred to as LPAI is presented. It is the first generic framework for implementing AIoT technology based on LPWAN applicable to acoustic scene classification scenarios. LPAI overcomes the domain-shift problem, which enables resource-constrained edge nodes to continuously improve their performance using real data to become more adaptive to the environment. For efficient use of limited resources, the edge nodes independently select representative data and transmit it back to the cloud. Moreover, the model is iteratively retrained on the cloud using the few-shot uploaded data. Finally, the feasibility of LPAI is analyzed, and simulation experiments on the public ASC dataset provide validation that our proposed framework can improve the recognition accuracy by as little as 5% using 85 actual sensor data points.

Keywords: AIoT; domain adaptation; edge intelligence; LPWAN

check for updates

Citation: Jing, X.; Tian, X.; Du, C. LPAI—A Complete AIoT Framework Based on LPWAN Applicable to Acoustic Scene Classification Scenarios. *Sensors* **2022**, *22*, 9404. https://doi.org/10.3390/s22239404

Received: 3 November 2022
Accepted: 29 November 2022
Published: 2 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Low-Power Wide Area Network (LPWAN) is a wireless network that is an essential part of wireless communication for the Internet of Things (IoT) [1] and has received extensive attention in recent years [2]. LPWAN is designed to allow low-power and long-range transmission, which makes LPWAN have the following features. First, it has strict transmission rate and payload length limitations. Second, the edge nodes in LPWAN are required to operate for long periods without battery replacement, and are usually ultra-low-power embedded devices with about a few megabytes of memory resources, thus they have extremely limited computing resources compared to mainstream lightweight networks with millions of parameters [3]. Moreover, low-power microcontrollers use MHz-level frequencies, while mobile devices employ GHz-level frequencies, so their computational resources are also scarce. Among all LPWAN technologies, LoRa is the most widely used [4]. Unlike other solutions, LoRa is deployed in unlicensed bands so that users can build networks independently. As a result, anyone can have complete ownership and control of the LoRa network at a low operational cost.

LPWAN has been used in medical treatment [5,6] and traffic monitoring [7], especially in acoustic scene classification (ASC) scenarios. ASC refers to recognizing different indoor and outdoor scenarios via acoustic signals recorded by sensors. Many studies utilize AIoT technologies to optimize the performance of ASC tasks [8]. Zualkernan et al. [9] have designed a system for monitoring bat species based on echolocation audio. For scenarios such as wildlife monitoring, which requires the long-term deployment and challenging maintenance, LPWAN is the most effective network option. Implementing

AIoT technology in these acoustic scenarios allows offloading computational tasks to nodes located at the edge of the network, thus effectively reducing the amount of data transmission in the medium and extending the lifetime of the edge nodes, which aligns with the needs of LPWAN. Subsequently, this brings several benefits [10,11], including ultra-low latency [12], reduced power consumption [13,14] and improved network reliability and data security [14,15].

However, due to the transmission characteristics of LPWAN, there remain challenges to the implementation of AIoT technologies in LPWAN. In particular, it is difficult for the cloud to utilize data from the sensors for model updates after deploying intelligent models, resulting in still poor recognition performance of edge nodes. This occurs due to several reasons.

- The domain shift problem exists between the environment and the data available in the cloud. Domain shift is caused by sensor type changes and deployment locations, resulting in differences between source and target domains, thus weakening the intelligence capabilities of the edge nodes.
- It is costly for LPWAN to upload a single packet of raw data back to the server. For example, the LoRa payload comprises only about a hundred levels of bytes. Capturing a 1-second voice recording at 16Khz can produce 64K bytes of raw data. Therefore, hundreds of LoRa packets would need to be sent to deliver them all to the cloud, which creates a huge overhead.
- From a data volume perspective, existing mobile-network-based frameworks support the backhaul of all target domain data, which is almost impossible for LPWAN. Moreover, even if only a small amount of data is uploaded, it is ineffective to retrain based on these few-shot samples using traditional methods.

There has been much research on AIoT frameworks in recent years. Huawei Technologies Co. et al. [16–18] proposed mobile device frameworks designed for high-dimensional data such as images, which are not suitable for the more restrictive LPWAN. Other frameworks, including Chiu et al. [19], Zualkernan et al. [9] and Chang et al. [20] are based on LPWAN; however, these studies do not consider updating the intelligence model according to the environment. They have still not implemented edge intelligence for LPWAN in this sense.

To increase the suitaility of LPWAN edge nodes for the environment, we propose LPAI, the first generic framework for implementing AIoT technology based on LPWAN applicable to ASC scenarios. LPAI not only realizes the intelligence of edge nodes in LPWAN, but more importantly, it allows resource-constrained edge nodes to continuously improve their performance using raw data, thus becoming more adaptive to the environment. The main contributions can be summarized as follows:

- LPAI enables the intelligence of edge nodes and reduces bias between the source and the target data. This enables edge nodes to be applied in real-world scenarios and continuously utilize unlabeled target data from the environment to improve their recognition performance.
- A data screening mechanism suitable for LPWAN is designed to improve the recognition performance of edge nodes. In this mechanism, edge nodes make independent decisions and only upload compressed features that help improve performance. Furthermore, iterative retraining is performed in the cloud based on few-shot compressed features returned by the nodes.
- We evaluate multiple datasets in ASC scenarios, proving that LPAI can become a general AIoT framework of LPWAN applicable to the ASC scenarios.

2. Related Work

2.1. Domain Adaptation

In computer vision, domain shift problems are solved using domain adaptation (DA) techniques. With DA techniques, the bias caused by the target data collected by actual sensors can be better mitigated, forcing the data distributions of the two domains to be

similar [21–29]. Researchers have gradually veered toward wearables [30] and mobile devices [31,32], but rarely toward the more restrictive LPWAN.

2.2. AIoT Frameworks

We summarize the research on AIoT frameworks for mobile devices and LPWAN, as shown in Table 1. Huawei Technologies Co. [16] proposed an open-source deep learning training and inference framework called MindSpore. E. Raj et al. [17] implemented a deep learning model for human pose estimation and tracking based on MindSpore in computer vision. Rong et al. [18] designed a collaborative computing platform between edge devices and the cloud to support continuous model evolution and system updates. The foregoing studies are based on mobile networks. They support the uploading of all target domain data regardless of resources. Chiu et al. [19] propose an AIoT precision feeding management system based on LoRa network to improve the existing automatic feeding system in the mark. Chang et al. [20] proposed an intelligent assistive system based on wearable smart glasses and smart cane. However, LPWAN is only responsible for transmitting GPS information of where a fallen person is located. The wearable smart glasses implement the intelligence task.

Table 1. The latest research on AIoT frameworks and the differences in LPAI .

Reference	Proposed	Based on LPWAN	Rely on the Backbone Network	Consider Model Update
[16]	An open-source deep learning training and inference framework	×	✓	×
[17]	A deep learning model for human pose estimation and tracking	×	✓	×
[18]	A collaborative computing platform between edge devices and the cloud	×	✓	✓
[9]	A system to monitor bat species based on echolocation audio.	✓	×	×
[19]	An AIoT precision feeding management system to measure water surface fluctuations in areas of fish pellet application	✓	×	×
[20]	An intelligent-assistance system for visually impaired people to achieve the goals of aerial obstacle avoidance and fall detection.	✓	×	×
[33]	A wearable assistive system to help visually impaired consumers safely use marked crosswalks	✓	✓	×
[34]	A flood observation system based on LoRa to integrate with the Internet	✓	✓	×
LPAI	A generic framework for implementing AIoT technology based on LPWAN applicable to the acoustic scene classification (ASC) scenarios	✓	×	✓

3. Methods

3.1. Overview

LPAI is a general AIoT framework based on LPWAN for acoustic scene classification scenarios. Like the traditional AIoT framework, LPAI contains four layers from top to bottom: the application, platform, transport, and perception. The overall framework diagram of LPAI is in Figure 1. The application layer contains urban acoustic scenarios. Acoustic sensors are deployed at the entrance of construction sites to monitor incoming and outgoing traffic flow during unconventional hours, effectively discouraging illegal

site operations. In this scenario, the input is the wave data collected by the edge nodes through acoustic sensors. After model inference, the output is the scene which the acoustic data belong to, such as traffic or pedestrian. Subsequently, the edge node determines whether the current data need to be uploaded by our proposed data screening algorithm. When a certain amount of data from the target domain is collected in the cloud, it is retrained with our cloud-based retraining algorithm. A new model that is more applicable to the environment is eventually obtained, thus continuously improving the recognition performance of nodes in real-world applications.

In LPAI, the platform layer comprises the cloud platform and operating system that handles complex intelligent tasks. The transport layer is mainly responsible for uplink and downlink data transmission. Furthermore, the perceptron layer is responsible not only for sensing, execution, and control, but also for executing small AI computational tasks and making independent decisions.

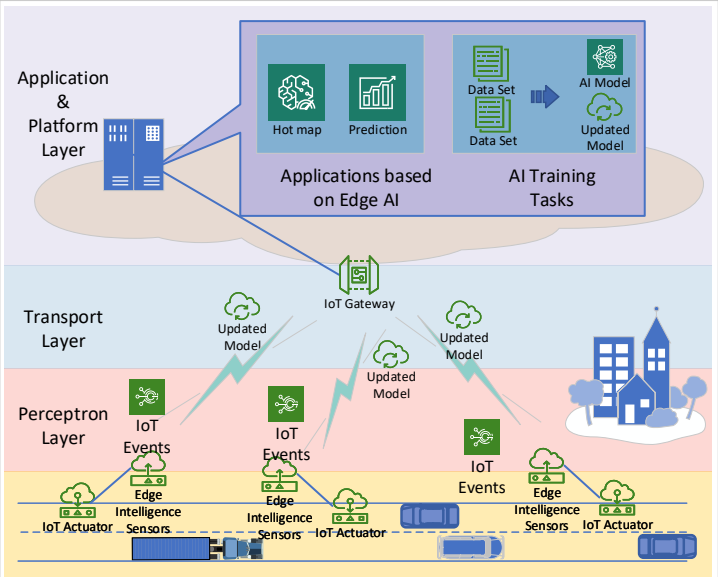


Figure 1. The overall framework of LPAI. It is divided into four layers, namely the application layer, platform layer, transport layer and perception layer.

3.2. The Transport Layer

The transport layer is first introduced to describe the data flow of the framework. The transport layer is based on the standard LPWAN protocol to ensure generality of the LPAI. The transmission task is divided into uplink and downlink, as summarized below.

- The uplink uploads from the edge side to the cloud. To save resources, it mainly uploads compressed features of raw data from the environment and the statistical events of edge nodes.
- The downlink includes the model to be updated and the spatial distribution of the source domain, providing the basis for nodes to filter the data independently.

In LPAI, the complete data flow process after implementing edge intelligence is as follows. An edge node periodically uploads the statistical information to the cloud to notify when a vehicle is detected. At the same time, it only uploads the compressed features that are more helpful for model updates. After a certain number of compressed features are uploaded, a new model is generated in the cloud. The newly produced model will then be sent down to the edge node via the downlink of the LPWAN protocol. Thus, a closed-loop complete framework is constructed. The complete data flow process is shown in Figure 2.

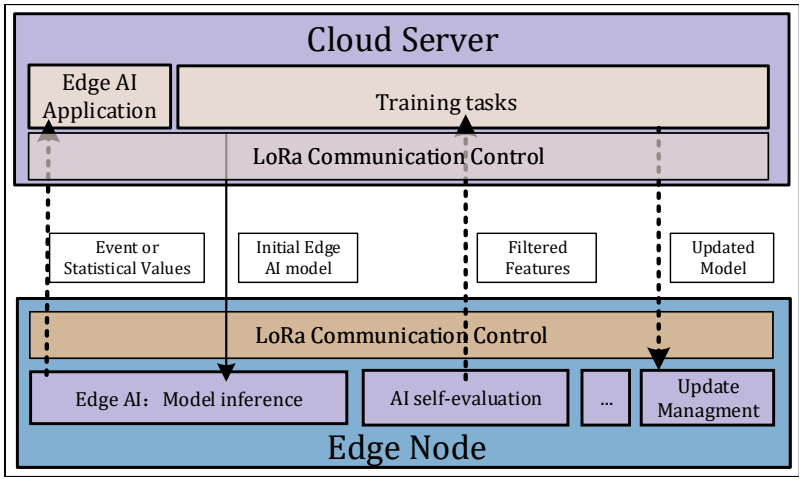


Figure 2. Data flow in the transport layer. In LPAI, the transport layer is mainly responsible for uplink and downlink data transmission. The arrows in the figure represent the data flow. The solid black line represents the initialization model that needs to be deployed locally on the node before the system operates, and the dashed line represents the data transmitted through the LPWAN protocol after deployment. Among them, the compressed features processed by the edge node and the response events are uploaded in the uplink, and the model to be updated is transmitted in the downlink.

3.3. The Platform Layer

In LPAI, the platform layer provides complex computing tasks in the cloud. Its tasks are listed below, and are also shown in Figure 3:

- Generate a model that can classify acoustic scenes to enable edge intelligence for LPWAN.
- Perform a cloud-based retraining algorithm that can receive features from the target domain and retrain a new model using both the source domain and unlabeled target domain features.
- Describe the spatial distribution of the existing source features.

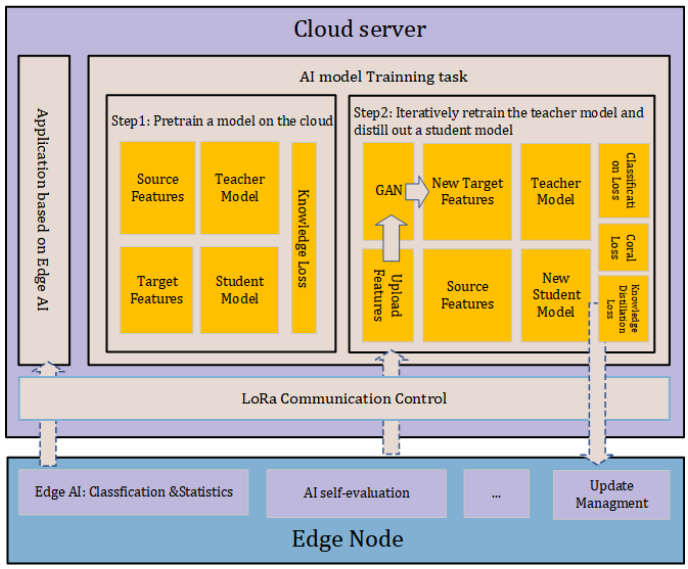


Figure 3. Tasks on the cloud. The task is divided into two steps, the first step is to train an initial model S_{init} to implement edge intelligence. The second step is to retrain a teacher model with the target domain features, and then distilling a student model. Finally, the student model is redistributed to the nodes.

3.3.1. Training an Initial Model

Initially, a portion of the labeled source features is on the cloud. This part of the source features needs supervised training to get an initial model M_{init}. Unlike traditional deep learning, this model performs simple classification tasks, usually two- [35], three- or four-class problems. In general, the more complex the classification task, the higher the training cost. Subsequently, M_{init} can then be compressed and prepared for deployment on resource-constrained devices.

According to Hinton's knowledge distillation method [36], a deep and complex network can transfer to a small and shallow network, also known as the teacher–student model. Inspired by the recent model distillation technique, we design a soft target loss in order to transfer novel knowledge. Considering an example $x_i^t \in \mathcal{D}$, the teacher network can produce class probabilities by a softmax layer that converts its logits $z_i = \{z_i^1, \ldots, z_i^K\}$ to an output of probability y_i. The temperature T is put into the standard softmax function as shown in Equation (1), where T is a parameter denoting temperature, set to 1 in a standard softmax. It converts the logit values to pseudo-probabilites. The teacher model's knowledge is transferred to the student model through Equation (4), using cross-entropy as the loss function, where L_{soft} in Equation (2) refers to the cross entropy between the soft labels of the teacher p_j^T and the student q_j^T when the temperature is T. The higher the temperature, the greater the attention paid to negative labels. Since the teacher's judgment is not entirely correct, L_{hard} in Equation (3) is introduced to prevent overfitting. It represents cross-entropy between the student's soft and hard labels at temperature $T = 1$. When the teacher's judgment is incorrect, the student learns by referring to the correct label. Finally, the student model S_{init} is distributed to the nodes.

$$y_i(x|t) = \frac{e^{\frac{z_i(x)}{T}}}{\sum_j e^{\frac{z_j(x)}{T}}} \tag{1}$$

$$\mathcal{L}_{soft} = -\sum_j^N p_j^T \log(q_j^T) \tag{2}$$

$$L_{hard} = -\sum_j^N C_j \log(q_j^{'}), C_j \in \{0, 1\} \tag{3}$$

$$\mathcal{L}_{KD} = \alpha \mathcal{L}_{soft} + (1 - \alpha)\mathcal{L}_{hard} \tag{4}$$

3.3.2. Cloud-Based Retraining Algorithm

Because the number of newly uploaded target features is much smaller than the existing source domain, a feature-specific augmentation method is required to prevent poor generalization performance. The classical methods used for the augmentation of acoustic are usually specific to the raw data [37,38], not the features. Generative Adversarial Networks (GAN) are a popular research area in computer vision. GAN can expand target features by creating new features using its generator. Moreover, they can also achieve excellent performance in the case of few-shot learning. The structure of GAN is displayed in Figure 4. In the retraining phase, the training and test sets are the source domain f_{s_i} and the expanded target features $f_{new_t_i}$.

A pseudo-label is added to each unlabeled target feature in each training epoch. Because these pseudo-labels are still unreliable, features with confidence greater than γ_{server} are selected iteratively and fed into the neural network. As the epoch increases, the model performance will improve, and the number of target features involved in training will increase, further enhancing the overall recognition performance. In a previous work, many loss functions were designed to decrease the distance between the source and target domains, such as MMD [39]. In LPAI, a simple coral loss [40] is used based on the two-norm distance of the covariance matrix between the two domains. As in Equation (5), d represents the batch size of each round of training. The source and target domain covariance matrices

calculate as Equations (6) and (7), where **1** represents a column vector. This part is shown in Algorithm 1.

$$\mathcal{L}_{DA} = \frac{1}{4d^2}\|C_s - C_t\|_2^2 \tag{5}$$

$$C_s = \frac{1}{N_s - 1}(X_s^T X_s - \frac{1}{X_s}(\mathbf{1}^T X_s)^T(\mathbf{1}^T X_s)) \tag{6}$$

$$C_t = \frac{1}{N_t - 1}(X_t^T X_t - \frac{1}{X_t}(\mathbf{1}^T X_t)^T(\mathbf{1}^T X_t)) \tag{7}$$

$$\mathcal{L} = \mathcal{L}_{classification} + \alpha\mathcal{L}_{DA}, \text{where } 0 \leq \alpha \leq 1. \tag{8}$$

Algorithm 1 Cloud-Based Retraining Algorithm

1: Given a classification task with C categories.
Input: Source features f_{s_i} and some uploaded features f_{t_i}.
2: Perform data augmentation on f_{t_i} and obtain $f_{new_t_i}$. Use a portion of f_{s_i} and $f_{new_t_i}$ as the training set $\mathcal{X}_{retrain}$ to train a teacher model.
3: **for** each epoch **do**
3: Obtain samples x_t with a confidence level higher than γ_{server}.
3: Calculate the coral loss $\mathcal{L}_{DA}$.
3: Calculate the total loss in Equation (8).
4: **end for**
5: **Output:** A new Teacher model $\mathcal{M}_s$.

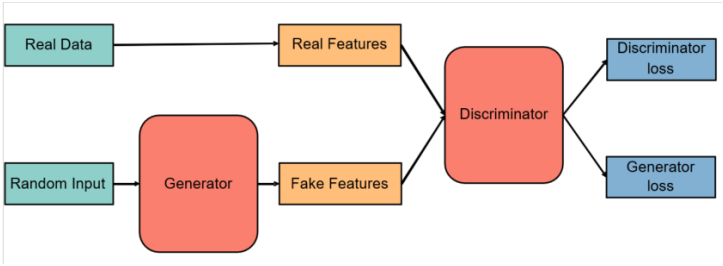


Figure 4. The structure of GAN. The generator is responsible for generating corresponding instances to deceive the discriminator. In this way, samples can be generated from a given domain. Real features represent the newly uploaded target domain features.

The teacher model is transferred to a new student model $\mathcal{M}_t$ after retraining. The identical model compression method mentioned in Equation (4) is applied here. Beyer et al. [41] showed that knowledge distillation is more effective when the inputs to the teacher and student models are consistent, so $\mathcal{X}_{retrain}$ is still used as the input to the student for distillation. The complete compression process is in Algorithm 2.

Algorithm 2 Compression Algorithm

Input: Use $\mathcal{X}_{retrain}$ as the training set and a teacher model $\mathcal{M}_s$.
1: **for** each epoch **do**
1: Calculate the soft loss $\mathcal{L}_{soft}$.
1: Calculate the hard loss $\mathcal{L}_{hard}$.
1: Calculate the total loss in Equation (4).
2: **end for**
3: **Output:** A new student model $\mathcal{M}_t$.

3.3.3. Data Distribution

The cloud requires computation of source domain features prior to being sent to edge nodes in order to provide a basis for them to determine whether to submit the data.

However, storing all the source domain features is far beyond the memory capacity of an edge node in LPWAN. The cloud only calculates the normalized average vector $\overline{d}_{s_i}$ of each category in the source domain to conserve memory overhead. As for how the nodes use $\overline{d}_{s_i}$, this will be continued in Section 3.4.3.

3.4. The Perception Layer

Unlike traditional AIoT, the perception layer in LPAI handles more additional tasks. Therefore, edge nodes are supposed to schedule the tasks rationally within the limited resources. Figure 5 shows the task flow diagram of the edge node. The tasks are divided into three main parts:

- The edge node collects real-world data and performs inference.
- During each inference cycle, the edge node determines whether the current features after feature extraction needs to be sent back to the server.
- The edge node receives a new model and loads it into memory.

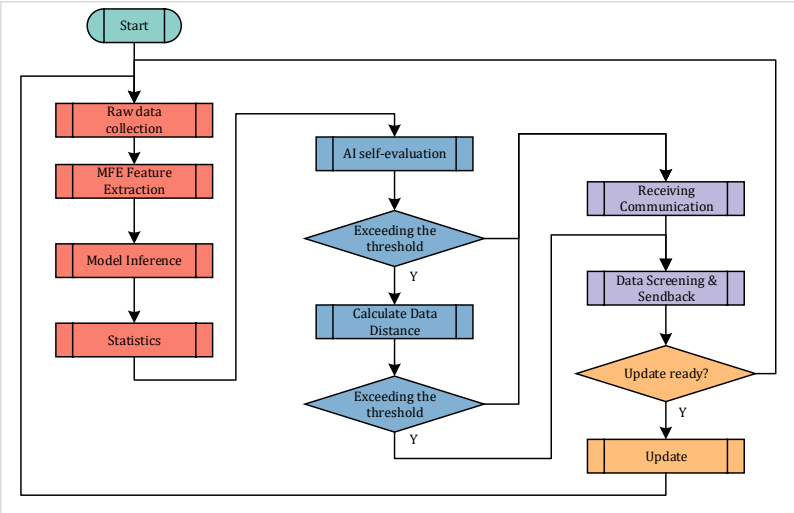


Figure 5. Node State Diagram. The leftmost boxes show regular tasks of the node, i.e., constantly collecting data for inference. The middle section displays the intelligent decision-making function specific to LPAI. Furthermore, the right section displays the data interaction process with the upper layer network.

3.4.1. Edge Intelligence

In LPAI, a neural network model is trained in the cloud using the Keras framework and then converted into hexadecimal data suitable for embedded devices with the help of Tensorflow Lite. This tool is used for deploying deep learning models on mobile and embedded devices, consisting of a converter and an interpreter. It also accomplishes network optimizations, such as quantization during the conversion process. The interpreter is responsible for transforming a .tflite file into a format that can be deployed on mobile devices and embedded microcontrollers. The hexadecimal model is then loaded into the device with the help of an open-source inference framework provided by Edge impulse to implement model inference.

3.4.2. Feature Extraction

To significantly reduce the amount of data returned, important information needs to be extracted from the raw data. The traditional spectrogram method divides the window into multiple overlapping frames and then computes the FFT for each frame [42]. The size and number of frames can be adjusted with the parameters Frame length and Frame stride. For example, with a window of 1 s, frame length of 0.02 s and stride of 0.01 s, it will create 99 time frames. An FFT is then calculated for each frame. Finally, the noise floor value is applied to the power spectrum. In edge impulse, to adapt to non-voice data, two additional

steps are added. After computing the spectrogram, a triangular filter is applied on the Mel scale to extract frequency bands. The main idea is to extract more features in low and less in high frequencies. The final step is to perform local mean normalization of the signal, applying the noise floor value to the power spectrum.

3.4.3. Data Screening Algorithm

The edge nodes of LPAI can screen representative features autonomously. As mentioned above, the average vector $\overline{d}_{s_i}$ is pre-stored in the node as a baseline for the source domain data. Y. Kim et al. [43] mentioned that the pseudo-labels are more accurate for data closer to the source domain, while the opposite is true for data farther away from the source domain, as shown in Figure 6. The algorithm of Yang et al. and Y. Kim et al. [31,43] cannot be directly applied to LPWAN because of the limitations of embedded devices. Hence, a more ingenious implementation is designed.

In each round of inference, the edge node collects raw acoustic data through sensors, acquires f_{t_j} after feature extraction, and then infers the classification result by forward propagation. The node decides whether to upload the feature if the confidence level is higher than γ_{nodes}. γ_{nodes} can be interpreted as the probability that the node has γ_{nodes} to trust its judgment. It should be set lower than γ_{server} because the features that confused the nodes should be selected and retrained.

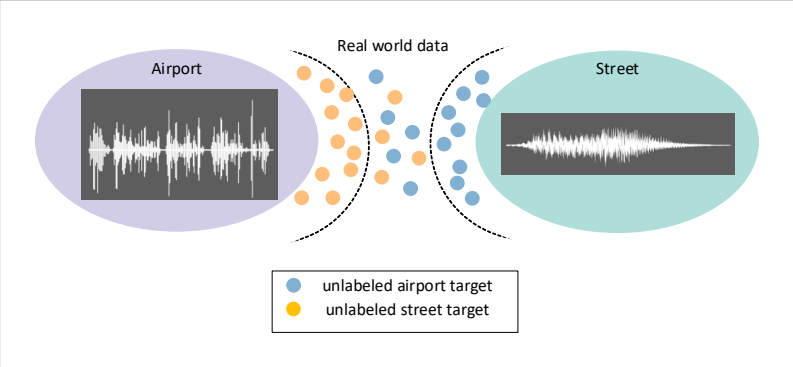


Figure 6. Taking binary classification as an example, the labeled source domain data can be classified by the class boundary, but in the target domain, the data closer to the class boundary has a higher probability of being correctly classified, while the data farther away from the class boundary is difficult to distinguish. Assigning pseudo-labels to all target data generates numerous errors.

Next, the distance d_{f_j} between f_{t_j} and each $\overline{d}_{s_i}$ is calculated. However, an edge node cannot record each f_{t_j} and sort them based on the distance as Y. Kim et al. did. Even storing the reduced-dimensional features is still impossible for low-power devices. Therefore, a more straightforward approach is taken to find the target features that confuse the nodes. It is to design a temporary variable d_{temp}, which is the average of d_{f_j}. It represents the average distance of all the target domain data from the source domain. Whenever a edge node obtains a f_{t_j}, it caculates the distance d_{f_j} and update d_{temp}. To ensure the correctness of the pseudo-labeling while electing features confused the nodes, the final features to be uploaded are those near d_{temp}, which is to satisfy Equation (9). Because the number of newly data is as few as possible, η is as close to 0 as possible. The judgment process is shown in Algorithm 3. Its computational complexity is $\mathcal{O}(n)$, where n is the total number of data points in the target domain.

$$(1-\eta) \times d_{temp} < d_{f_{t_i}} < (1+\eta) \times d_{temp} \tag{9}$$

Algorithm 3 Data Screening Algorithm On the Node

Given $\overline{d}_{s_i}$ from the server, $i = 1 \ldots C$, where C is the number of categories.
Input: A new target data:
Obtain f_{t_j} after feature extraction.
Perform inference.
if Its confidence level is greater than γ_{nodes} **then**
  for each f_{t_j} **do**
    Calculate the distance between each f_{t_j} and the $\overline{d}_{s_i}$ of the corresponding pseudo-label category.
    Update the average distance d_{temp} once.
    if Equation (9) is satisfied **then**
      Output: upload f_{t_j}.
    end if
  end for
end if

4. Evaluations and Results

In this section, we conduct extensive experiments to demonstrate the ability of LPAI to enable edge nodes to enhance their capabilities in an environment based on few-shot data. We validate it with three acoustic scene classification datasets using LoRa, a typical network in LPWAN. Experiments are based on a four-classification scenario. We have deployed LoRa nodes at a school intersection to monitor the operation of shuttles. Its hardware diagram is shown in Figure 7.

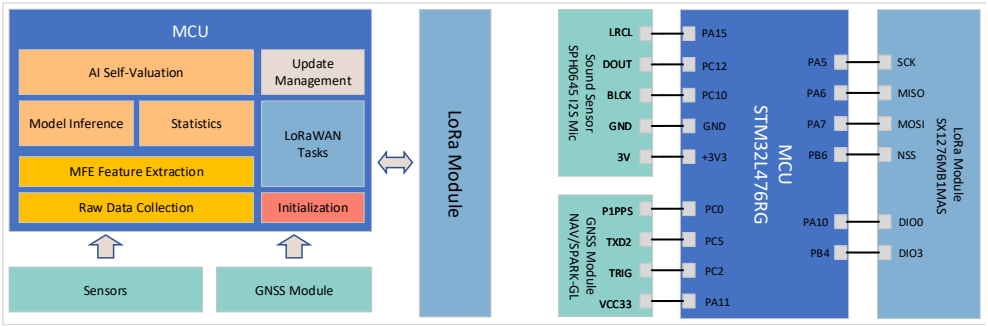


Figure 7. A simple memory management diagram for edge nodes.

The evaluation is divided into three parts: the first part evaluates the effectiveness of the cloud-based retraining phase. It is assumed that all target domain features have been uploaded, so the model is trained with the source domain data and a portion of the target domain data and also tested on the remaining target domain data. Moreover, the second part evaluates the effectiveness of Algorithm 3 for screening the dataset. The results of uploading all target data and the random screening method are used as benchmarks. Suppose the model is retrained after collecting 85 points of data to mimic the operation pattern of nodes in the environment. In the last section, we combine experimental and theoretical analysis to explore the impact on LoRa network performance.

4.1. Datasets

- TAU Urban Acoustic Scenes 2020 Mobile, development dataset [44]. It contains recordings from 12 European cities in 10 different acoustic scenes using 4 different devices. Because only four-classification questions are discussed, four scenarios commonly used in LPWAN are chosen: metro, metro station, street pedestrian and traffic. However, changes in the environment can lead to differences between domains, and the evaluation is divided into five domains according to the collection location, namely A1 (Barcelona, Helsinki), B1 (Lisbon, London), C1 (Milan, Lyon), D1 (Paris,

Prague), E1 (Stock, Vienna). This dataset is divided into several source-domain and target-domain datasets according to different locations. The baseline system for the 2020 DCASE Challenge provides an accuracy of 54.1%.

- Urban sound 8K [45]. The dataset contains 8732 labeled sound excerpts from 10 categories of urban sounds. Take four of them: marked as A2. Refer to paper [45] for baseline results.
- Dataset AOB [46]. This dataset uses a convolutional neural network to collect and manually edit an audio dataset for urban sound event classification for a master's thesis. Take four of them: marked as A3.

After experiments, we discovered that an edge node could recognize a dataset up to 2 s in length in our scheme. Considering power consumption, recognition performance and other factors, the audio length was determined to be 1 s. Therefore, the above three datasets were downsampled to 16 kHz and cut into 1-second fragments before training. After feature extraction, the original dataset with a length of 1 s and a size of 64K bytes were compressed into feature data with 4K bytes. This means that the quantity of data transmitted over the network is compressed by a factor of 16.

4.2. Setup and Implementation Details

4.2.1. Evaluating Cloud-Based Retraining Algorithms

In this section, we evaluate the effectiveness of the retraining phase. We first train an initial teacher model with the source domain data to evaluate its performance on the target domain, providing a baseline result. The structure of the teacher model is shown in Figure 8. We use 80% of the source domain as the training set and test the remaining 20%. The reason for designing this experiment is to demonstrate that the domain-shift problem exists and to provide a baseline for subsequent experiments to prove the validity of our approach.

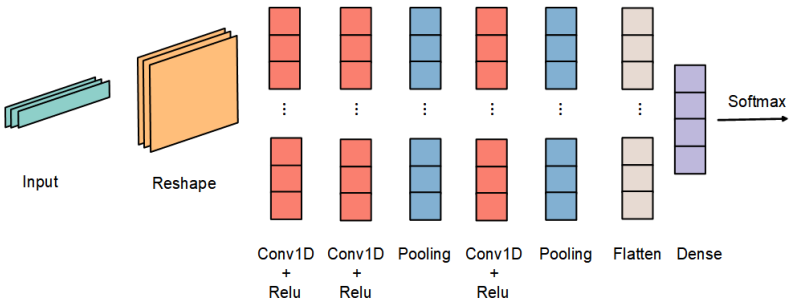


Figure 8. Teacher model structure. This structure comprises multiple 2D convolutional layers, pooling layers, and dropout layers. The convolutional layer represents the core part of recognition and is used to extract different features from the data. The pooling layer is used to reduce information redundancy and model computation, reduce the difficulty of network optimization, and prevent network overfitting.

Next, we evaluate the effectiveness of Algorithm 1. We assume that all unlabeled target domain data are uploaded to the cloud. Unlike the traditional approach, we use 80% of the unlabeled data as the training set and 20% as the test set. The parameters are set as follows, γ_{nodes} is 0.5, and γ_{server} is 0.7. Adam is employed as the optimizer. Its learning rate is 0.0001, β_1 is 0.9, β_2 is 0.99, and the iteration epoch is 20.

Subsequently, we evaluate the effectiveness of the cloud-based retraining phase by assuming all target domain feature data are uploaded to the cloud. The reason for this design is firstly to evaluate that our method can reduce the bias between different domains, and secondly to provide a control for subsequent evaluation of the data screening algorithm. Therefore, the inputs to this part of the algorithm are all labeled source domain features and all unlabeled target domain features. Unlike the traditional approach, we use 80% of the unlabeled data as the training set and 20% as the test set. The parameters are set as follows: γ_{nodes} is 0.5, and γ_{server} is 0.7. Adam is used as the optimizer with a learning rate

of 0.0001, β_1 is 0.9, β_2 is 0.99, and the iteration period is 20. The output of the algorithm is a teacher model with optimal performance.

Finally, the performance after compression was evaluated. In this part, the input of the student model is the same as in the previous step, i.e., the input of the teacher model, and after iterative training, an optimal student model is obtained. In addition, the student model is finally updated to the edge node. The structure of the student model is shown in Figure 9.

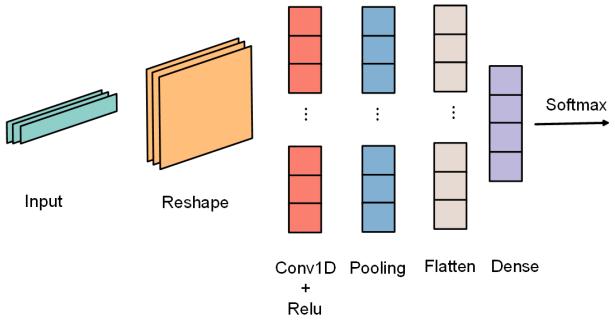


Figure 9. Student model structure. This structure consists of two-dimensional convolution, pooling and dropout layers, with a smaller number of parameters than the teacher model and suitable for low-power embedded devices.

The results of the first part of the experiment are presented in Table 2. All results are measured according to two criteria: accuracy and Kappa coefficient. Accuracy is the most common metric for classification problems and is used to measure the model's accuracy on the test set. The Kappa coefficient is commonly used for multi-classification issues and is employed to measure consistency. Consistency refers to whether the model predicts the same results as the actual classification results. It usually lies between 0 and 1, and a larger Kappa coefficient indicates greater consistency. The experiments were conducted on three data sets, and group A1–B1 means that A1 is the labeled source domain and B1 is the unlabeled target domain data. The first two columns record the model performance trained with the source domain data on the source test set and the target domain. The performance of the source model on different target domains is recorded to facilitate comparison with the results of subsequent experiments. The last three columns show the performance after our cloud-based retraining algorithm. The performance of the model in the source domain, the target domain, and the student model in the target domain after retraining is recorded, respectively. The last column records the performance of the model deployed on the edge nodes after compression. Taking group A1–B1 as an example, we can see from S (before DA) and T (before DA) that, as expected, the performance of the model trained on the source domain data degrades on the target domain data, which indicates that the domain-shift problem arises from changes in the environment. After retraining, the data of S (after DA) are generally better than S (before DA), which means that the performance of the model does not degrade, or indeed even slightly increases on the source domain data. Furthermore, the performance of the target domain data is also greatly improved. That is, our approach makes the recognition performance of edge nodes better. Finally, the recognition accuracy remains the same before and after compression from the last column. Most importantly, a model of this size can be easily deployed on embedded devices. Table 3 shows the time and space complexity. The time complexity of the model is measured by floating point operations (FLOPs), and the space complexity is measured by the number of model parameters. The compression ratio is approximately 0.4.

Table 2. The training results in the source and target domains, where S represents the test results of the teacher model in the source domain, T represents the results of the teacher model in all target domain data, and stu accuracy denotes the results of the student model, acc denotes the accuracy, and $\mathcal{K}$ is the Kappa coefficient.

	S (before DA)		T (before DA)		S (after DA)		T (after DA)		stu	
	acc	$\mathcal{K}$	acc	$\mathcal{K}$	acc	$\mathcal{K}$	acc	$\mathcal{K}$	acc	$\mathcal{K}$
A1–B1	66.48	0.5506	54.30	0.3906	69.95	0.5671	60.34	0.4688	60.08	0.4685
A1–C1	66.48	0.5506	59.12	0.4529	70.49	0.6183	64.46	0.5791	64.38	0.5790
A1–D1	66.48	0.5506	52.27	0.3606	66.79	0.5847	57.56	0.4102	57.76	0.4123
A1–E1	66.48	0.5506	56.88	0.4213	70.04	0.6092	62.17	0.4900	61.05	0.4853
B1–C1	66.98	0.5905	58.92	0.4969	69.74	0.6169	64.78	0.5217	62.34	0.5207
B1–D1	66.98	0.5905	55.91	0.4121	71.75	0.6136	60.17	0.4297	60.52	0.4304
B1–E1	66.98	0.5905	54.42	0.3915	71.59	0.6050	60.52	0.4712	60.35	0.4710
C1–D1	73.27	0.6409	55.59	0.4204	73.66	0.6833	58.60	0.4284	58.91	0.4302
C1–E1	73.27	0.6409	53.43	0.3675	73.79	0.6281	59.96	0.4022	59.97	0.4020
D1–E1	65.54	0.5444	59.91	0.3672	67.71	0.5641	63.53	0.4914	63.51	0.4915
A2–A3	76.32	0.8625	70.68	0.6754	79.13	0.8649	76.23	0.7714	76.41	0.7728
A3–A2	75.42	0.8160	70.32	0.5499	84.77	0.8158	76.68	0.7584	76.92	0.7613

Table 3. Algorithmic complexity of neural networks. Floating point operations (FLOPs) represent the time complexity and the number of model parameters measures the space complexity.

Model	FLOPs	Parameter Quantity
Teacher	3304	3340
Student	1762	1388

4.2.2. Evaluating Data Screening Algorithms

The effectiveness of the algorithm for screening the datasets is then evaluated. To highlight that our algorithm can select data that are more useful for model updates, we designed a random screening method and Algorithm 3 for comparison, where n data were randomly screened on the target domain data. We screened three times and took the average value with the same amount of data to ensure fairness and randomness. The results are shown in Figure 10 for A1–D1 group of the TAU dataset. We also evaluated the effect of the data augmentation method. In Figure 10, the blue line represents the performance of our algorithm. By default, 85 pieces of data are filtered by adjusting η in Equation (9), and then the target domain is expanded to the corresponding n using the data multiplication method.

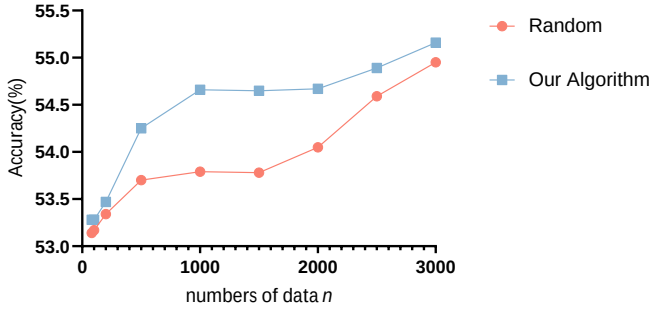


Figure 10. Randomly filter n pieces of data and use them to retrain the corresponding accuracy.

In addition, to emphasize the ability of our algorithm to render the nodes more applicable to the environment within the constraints of LPWAN, three control groups were set up to evaluate the effectiveness of Algorithm 3. The first one is the accuracy of the source domain model in all target domain data before retraining, the second one is the accuracy of the new model in the only 85 pieces of uploaded target features after implementing Algorithm 3, and the third one is the accuracy after uploading all target domain features.

Figures 10 and 11 show the performance of the data screening algorithm. In Figure 10, the accuracy steadily increases as n increases, implying that our algorithm performs better compared to random screening and can sort out higher quality data with the same amount of data. The effectiveness of the data augmentation method can still be seen from the figure, where the accuracy increases gradually as n increases after 85 s data are uploaded. In Figure 11, the effectiveness of our method can be clearly seen from the three control groups. On the one hand, uploading only 85 s of data collected from one node can significantly improve the recognition performance compared to not performing updates (before retraining in Figure 11). On the other hand, similar results are achieved in both evaluation criteria of accuracy and Kappa coefficient compared to uploading tens of thousands of data points. In particular, in some groups, such as A3–A2, the accuracy is improved by 5% and the Kappa coefficient by 0.17 compared to uploading all data. While there are still gaps with regard to uploading all the target features, these gaps are insignificant compared to the consumption of resources such as power. This is also in line with the usage scenario of low-power IoT.

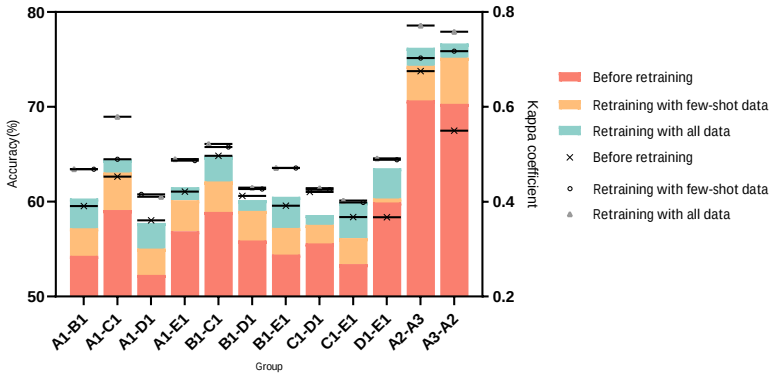


Figure 11. Data screening algorithm evaluation results. The bar chart indicates the accuracy before and after training, and the scatter plot indicates the Kappa before and after training. Each evaluation criterion is compared with three control groups, the first one is the accuracy of the model in the target domain without retraining, the second one is the result of uploading only 85 data points from the Algorithm 3, and the last one is the result of uploading all target domain features.

4.2.3. Evaluating Network Performance

Finally, we evaluate the network performance of the LPAI framework and its impact on the network in the standard LoRaWAN protocol through experiments and theoretical calculations, including the response time and the time required for model updates.

LPAI's network response time. Our nodes support two application modes of edge intelligence.

The first is the one-shot recognition mode. Namely, edge nodes perform inference and data screening only once. The process includes 1. Data acquisition: From collecting raw data to encoding using digital audio sensors. 2. Model inference: After extracting the raw data by MFE features, the classification result is then propagated by a neural network. 3. Data screening. The data screening algorithm determines whether these data need to be sent to the cloud. We use an Analog Discovery 2 oscilloscope to calculate the time required for each stage by flipping the I/O pin level, and the result is shown in Figure 12. The result is divided into three parts, where the first part is the process of obtaining raw data collected by the sensor. For continuous acquisition, the microcontroller acquires voice data in DMA dual mode, each buffer holds 0.25 s of audio data, and the double buffer pointers

are exchanged each time one of the buffers fills up. The 0.25 s of data collected is appended to the MFE's operation window, and subsequent inference operations are performed when the window is filled with 1 s of data. The second part is to perform model inference after feature extraction from the original data. The third part judges whether to upload the compression feature through the data screening algorithm. The experimental results show that for one second of acoustic data, the total execution time is about 2 s.

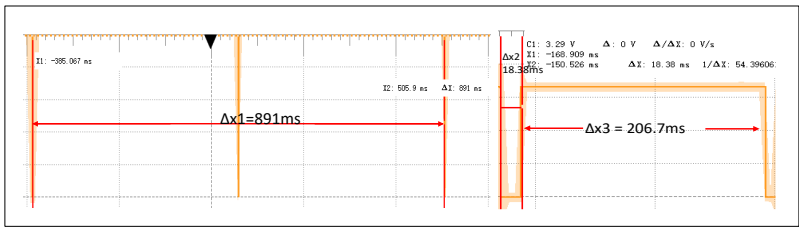


Figure 12. Response time for each task of the edge node. In the left figure, Δx_1 represents the time taken by the sensor to collect 0.5 s of data, in the right figure, Δx_2 represents the model inference time, and Δx_3 represents the time taken by the data screening algorithm.

The second is the periodic recognition mode: the edge nodes are continuously recognized at a fixed sampling interval over a long period. That is, the edge nodes count events continually within a day. Take the scenario of identifying a vehicle at an intersection as an example. Each event is expressed as a 1-bit Boolean value. Assuming that in the most extreme case: uploading statistics for every minute within one day requires 1440 bits of data. Therefore, the total payload uploaded per day is about 180 bytes. Using class C of the LoRaWAN protocol for theoretical calculation, the time T_{tx} required to upload 180 bytes in the CN470 frequency band is shown in Figure 13 and calculated by Equation (10) [47]. T_{tx} is composed of two parts: the preamble and the payload, calculated as Equations (11) and (12), where SF denotes the spreading factor, BW is the bandwidth, N_{pre} is the preamble length, N_{PHY} is the payload length, CR denotes the coding rate which can take values from 1 to 4, and PL is the payload rate that indicates the physical payload length in bytes. It can be seen from the figure that as the spreading factor increases, the data rate decreases, and the transmission time of each data packet increases. When the spreading factor is 11, the response time is 7733 ms, which requires approximately 13 min with a 1% duty cycle.

$$T_{tx} = T_{preamble} + N_{PHY} \times \frac{2SF}{BW} \tag{10}$$

$$T_{preamble} = \frac{2SF}{BW} \times (N_{pre} + 4.25) \tag{11}$$

$$N_{PHY} = 8 + \max\left[ceil\left[\frac{28 + 8PL + 16 \times CRC - 4SF}{4 \times (SF - 2DE)}\right] \times (CR + 4), 0\right] \tag{12}$$

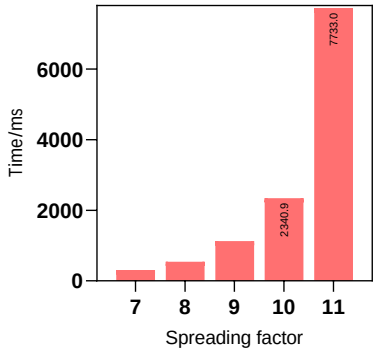


Figure 13. The LoRa node sends a 180-byte response time on the CN470 band.

Time to back propagate compressed features and model updates. LPAI only needs to upload 85 compressed features to ensure 5% accuracy improvement, and each compressed feature occupies 4k bytes. Since LoRaWAN has a constraint on the payload length of a single packet, in the worst case, where the SF is 11, the maximum payload length is 223 bytes, so a compressed feature requires 18 packets to be sent out. According to Equation (10), the time to backhaul a single 4KB dataset is about 10 min at a maximum duty cycle of 1%. Furthermore, for the model update part, consider the worst case, which is 30k bytes at a time, where the time required at an SF of 7 is 83 min. In these calculations, we assumed continuous packet transmission. In a real-world scenario, LoRa nodes only upload a few pieces of data per day. Therefore, complete data are only sent down after weeks or even months, and that amount matches the LPWAN scenario.

5. Conclusions

In this paper, we propose LPAI, the first generic LPWAN-based AIoT framework for acoustic scene classification. We demonstrate experimentally in LoRa networks that the recognition accuracy of LoRa nodes can be improved by 5% by uploading only 85 compressed data points. In addition, we analytically prove the feasibility of LPAI, which bears important practical implications. LPAI advances LPWAN to a new frontier. In future research, we will further improve LPAI. On one hand, we will evaluate other standard networks for LPWAN, such as sigfox, with longer transmission distances and more stringent conditions. On the other hand, LPAI should also be applied to the imaging-based fields to promote LPAI as a more general AIoT system.

Author Contributions: Methodology, X.J.; software, X.J.; validation, X.J.; formal analysis, X.T.; investigation, X.J. and X.T.; writing—original draft preparation, X.J.; writing—review and editing, X.J.; funding acquisition, C.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the "Strategic Priority Research Program" of Chinese Academy of Sciences (CAS), China, grant number XDC02070800, and "Shanghai Science and Technology Innovation Action Plan 2022" of The Science and Technology Commission of Shanghai Municipality (STCSM), China, grant number 22511100600.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://www.freepik.com/free-vector/sound-waves-collection_1014327.htm (accessed on 28 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
IoT	Internet of Things
AIoT	Artificial Intelligence of Things
LPWAN	Low–Power Wide Area Network
DA	Domain Adaptation
DCNN	Deep Convolutional Neural Network
ASC	Acoustic Scene Classification
SF	Spreading Factor

References

1. Song, Y.; Lin, J.; Tang, M.; Dong, S. An Internet of energy things based on wireless LPWAN. *Engineering* **2017**, *3*, 460–466. [CrossRef]
2. Mekki, K.; Bajic, E.; Chaxel, F.; Meyer, F. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express* **2019**, *5*, 1–7. [CrossRef]

3. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Adam, H. Mobilenets: Efficient 465 convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
4. Foubert, B.; Mitton, N. Long-Range Wireless Radio Technologies: A Survey. *Future Internet* **2020**, *12*, 13. Available online: https://www.mdpi.com/1999-5903/12/1/13 (accessed on 28 November 2022). [CrossRef]
5. Catherwood, P.A.; Steele, D.; Little, M.; Mccomb, S.; Mclaughlin, J. A Community-Based IoT Personalized Wireless Healthcare Solution Trial. *IEEE J. Transl. Eng. Health Med.* **2018**, *6*, 1–13. [CrossRef] [PubMed]
6. Petäjäjärvi, J.; Mikhaylov, K.; Yasmin, R.; Hämäläinen, M.; Iinatti, J. Evaluation of LoRa LPWAN technology for indoor remote health and wellbeing monitoring. *Int. J. Wirel. Inf. Netw.* **2017**, *24*, 153–165. [CrossRef]
7. Sharma, V.; You, I.; Pau, G.; Collotta, M.; Lim, J.D.; Kim, J.N. LoRaWAN-Based Energy-Efficient Surveillance by Drones for Intelligent Transportation Systems. *Energies* **2018**, *11*, 573. [CrossRef]
8. Abeßer, J. A Review of Deep Learning Based Methods for Acoustic Scene Classification. *Appl. Sci.* **2020**, *10*, 2020. [CrossRef]
9. Zualkernan, I.; Judas, J.; Mahbub, T.; Bhagwagar, A.; Chand, P. An aiot system for bat species classification. In Proceedings of the 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), BALI, Indonesia, 27–28 January 2021; pp. 155-160.
10. Zhang, J.; Tao, D. Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet Things J.* **2020**, *8*, 7789–7817. [CrossRef]
11. Xu, D.; Li, T.; Li, Y.; Su, X.; Tarkoma, S.; Jiang, T.; Crowcroft, J.; Hui, P. Edge intelligence: Empowering intelligence to the edge of network. *Proc. IEEE* **2021**, *109*, 1778–1837. [CrossRef]
12. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
13. Moschitta, A.; Neri, I. Power consumption assessment in wireless sensor networks. In *ICT-Energy-Concepts Towards Zero-Power Information and Communication Technology*; IntechOpen: London, UK, 2014.
14. Ghosh, A.; Chakraborty, D.; Law, A. Artificial intelligence in Internet of things. *Caai Trans. Intell. Technol.* **2018**, *3*, 208–218. [CrossRef]
15. Cui, L.; Yang, S.; Chen, F.; Ming, Z.; Lu, N.; Qin, J. A survey on application of machine learning for Internet of Things. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1399–1417. [CrossRef]
16. Huawei Technologies Co., Ltd. Huawei MindSpore AI Development Framework. In *Artificial Intelligence Technology*; Springer Nature: Singapore, 2022; pp. 137–162.
17. Raj, E.; Buffoni, D.; Westerlund, M.; Ahola, K. Edge MLOps: An Automation Framework for AIoT Applications. In Proceedings of the 2021 IEEE International Conference on Cloud Engineering (IC2E), San Francisco, CA, USA, 4–8 October 2021; pp. 191–200. [CrossRef]
18. Rong, G.; Xu, Y.; Tong, X.; Fan, H. An edge-cloud collaborative computing platform for building AIoT applications efficiently. *J. Cloud Comput.* **2021**, *10*, 1–14. [CrossRef]
19. Chiu, C.-C.; Liao, T.-L.; Chen, C.-H.; Kao, S.-E. AIoT Precision Feeding Management System. *Electronics* **2022**, *11*, 3358. [CrossRef]
20. Chang, W.J.; Chen, L.B.; Chen, M.C.; Su, J.P.; Sie, C.Y.; Yang, C.H. Design and Implementation of an Intelligent Assistive System for Visually Impaired People for Aerial Obstacle Avoidance and Fall Detection. *IEEE Sens. J.* **2020**, *20*, 10199–10210. [CrossRef]
21. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135–153. [CrossRef]
22. Yang, J.; Zou, H.; Zhou, Y.; Xie, L. Learning gestures from WiFi: A Siamese recurrent convolutional architecture. *IEEE Internet Things J.* **2019**, *6*, 10763–10772. [CrossRef]
23. Tzeng, E.; Hoffman, J.; Darrell, T.; Saenko, K. Simultaneous deep transfer across domains and tasks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4068–4076.
24. Motiian, S.; Piccirilli, M.; Adjeroh, D.A.; Doretto, G. Unified deep supervised domain adaptation and generalization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5715–5725.
25. Yan, H.; Ding, Y.; Li, P.; Wang, Q.; Xu, Y.; Zuo, W. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2272–2281.
26. Tao, Y.; Li, C.; Liang, Z.; Yang, H.; Xu, J. Wasserstein distance learns domain invariant feature representations for drift compensation of E-nose. *Sensors* **2019**, *19*, 3703. [CrossRef]
27. Yi, Z.; Zhang, H.; Tan, P.; Gong, M. Dualgan: Unsupervised dual learning for image-to-image translation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2849–2857.
28. Bousmalis, K.; Trigeorgis, G.; Silberman, N.; Krishnan, D.; Erhan, D. Domain separation networks. In Proceedings of the Advances in Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.
29. Ghifary, M.; Kleijn, W.B.; Zhang, M.; Balduzzi, D.; Li, W. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 597–613.
30. Chang, Y.; Mathur, A.; Isopoussu, A.; Song, J.; Kawsar, F. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proc. Acm Interact. Mobile Wearable Ubiquitous Technol.* **2020**, *4*, 1–30. [CrossRef]
31. Yang, J.; Zou, H.; Cao, S.; Chen, Z.; Xie, L. Mobileda: Toward edge-domain adaptation. *IEEE Internet Things J.* **2020**, *7*, 6909–6918. [CrossRef]
32. Li, H.; Chen, X.; Wang, J.; Wu, D.; Liu, X. DAFI: WiFi-based Device-free Indoor Localization via Domain Adaptation. *Proc. Acm Interact. Mobile Wearable Ubiquitous Technol.* **2021**, *5*, 1–21. [CrossRef]

33. Chang, W.J.; Chen, L.B.; Sie, C.Y.; Yang, C.H. An Artificial Intelligence Edge Computing-Based Assistive System for Visually Impaired Pedestrian Safety at Zebra Crossings. *IEEE Trans. Consum. Electron.* **2021**, *67*, 3–11. [CrossRef]
34. Sung, W.T.; Devi, I.V.; Hsiao, S.J. Early warning of impending flash flood based on AIoT. *Eurasip J. Wirel. Commun. Netw.* **2022**, *2022*, 1–18. [CrossRef]
35. Sudharsan, B.; Breslin, J.G.; Ali, M.I. Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices. In Proceedings of the 10th International Conference on the Internet of Things, Malmo, Sweden, 6–9 October 2020; pp. 1–8.
36. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
37. Kim, B.; Yang, S.; Kim, J.; Chang, S. QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design. *arXiv* **2022**, arXiv:2206.13909.
38. Yang, C.H.H.; Hu, H.; Siniscalchi, S.M.; Wang, Q.; Wang, Y.; Xia, X.; Zhao, Y.; Wu, Y.; Wang, Y.; Du, J.; et al. A lottery ticket hypothesis framework for low-complexity device-robust neural acoustic scene classification. *arXiv* **2021**, arXiv:2107.01461.
39. Gretton, A.; Borgwardt, K.M.; Rasch, M.; Schölkopf, B.; Smola, A.J. A kernel method for the two-sample-problem. In *Proceedings of the Advances in Neural Information Processing Systems;* MIT Press: Cambridge, MA, USA, 2007; pp. 513–520.
40. Sun, B.; Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 443–450.
41. Beyer, L.; Zhai, X.; Royer, A.; Markeeva, L.; Anil, R.; Kolesnikov, A. Knowledge distillation: A good teacher is patient and consistent. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–20 June 2022; pp. 10925–10934.
42. Gupta, S.; Jaafar, J.; Ahmad, W.W.; Bansal, A. Feature extraction using MFCC. *Signal Image Process. Int. J.* **2013**, *4*, 101–108. [CrossRef]
43. Kim, Y.; Kim, C. Semi-Supervised Domain Adaptation via Selective Pseudo Labeling and Progressive Self-Training. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 1059–1066. [CrossRef]
44. Mesaros, A.; Heittola, T.; Virtanen, T. A multi-device dataset for urban acoustic scene classification. *arXiv* **2018**, arXiv:1807.09840.
45. Salamon, J.; Jacoby, C.; Bello, J.P. A Dataset and Taxonomy for Urban Sound Research. In Proceedings of the 22nd ACM International Conference on Multimedia (MM '14), Association for Computing Machinery, New York, NY, USA, 3–7 November 2014; pp. 1041–1044. [CrossRef]
46. Ospina, A. Audio Event Classification using Deep Learning. Use case: Urban Sounds Events classification with Convolutional Neural Networks. Master's Thesis, Beuth University of Applied Sciences, Berlin, Germany, 2020.
47. Andrade, R.O.; Yoo, S.G. A comprehensive study of the use of LoRa in the development of smart cities. *Appl. Sci.* **2019**, *9*, 4753. [CrossRef]