Article

An Enhancement Method in Few-Shot Scenarios for Intrusion Detection in Smart Home Environments

Yajun Chen [1], Junxiang Wang [1,*], Tao Yang [2], Qinru Li [3] and Nahian Alom Nijhum [4]

[1] School of Electronic Information Engineering, China West Normal University, Nanchong 637001, China; scnccyj@cwnu.edu.cn
[2] Education and Information Technology Center, China West Normal University, Nanchong 637001, China; yangt@cwnu.edu.cn
[3] School of Computer Science, China West Normal University, Nanchong 637001, China; qrli@cwnu.edu.cn
[4] School of Software Engineering, China West Normal University, Nanchong 637001, China; 31nahian@gmail.com
* Correspondence: wangjx@stu.cwnu.edu.cn

Abstract: Different devices in the smart home environment are subject to different levels of attack. Devices with lower attack frequencies confront difficulties in collecting attack data, which restricts the ability to train intrusion detection models. Therefore, this paper presents a novel method called EM-FEDE (enhancement method based on feature enhancement and data enhancement) to generate adequate training data for expanding few-shot datasets. Training intrusion detection models with an expanded dataset can enhance detection performance. Firstly, the EM-FEDE method adaptively extends the features by analyzing the historical intrusion detection records of smart homes, achieving format alignment of device data. Secondly, the EM-FEDE method performs data cleaning operations to reduce noise and redundancy and uses a random sampling mechanism to ensure the diversity of the few-shot data obtained by sampling. Finally, the processed sampling data is used as the input to the CWGAN, and the loss between the generated and real data is calculated using the Wasserstein distance. Based on this loss, the CWGAN is adjusted. Finally, the generator outputs effectively generated data. According to the experimental findings, the accuracy of J48, Random Forest, Bagging, PART, KStar, KNN, MLP, and CNN has been enhanced by 21.9%, 6.2%, 19.4%, 9.2%, 6.3%, 7%, 3.4%, and 5.9%, respectively, when compared to the original dataset, along with the optimal generation sample ratio of each algorithm. The experimental findings demonstrate the effectiveness of the EM-FEDE approach in completing sparse data.

Keywords: data enhancement; few-shot data; smart home; generative adversarial networks; intrusion detection

Citation: Chen, Y.; Wang, J.; Yang, T.; Li, Q.; Nijhum, N.A. An Enhancement Method in Few-Shot Scenarios for Intrusion Detection in Smart Home Environments. *Electronics* **2023**, *12*, 3304. https://doi.org/10.3390/electronics12153304

Academic Editor: Domenico Ursino

Received: 21 June 2023
Revised: 19 July 2023
Accepted: 30 July 2023
Published: 31 July 2023

1. Introduction

With the development of Internet of Things (IoT) technology, the application of smart home scenarios is becoming increasingly widespread [1]. The number of smart home devices connected to home networks rapidly increases, leading to a surge in network scale and data traffic. These factors exacerbate security threats such as network attacks and privacy breaches, presenting new security challenges [2]. Additionally, different types of smart home devices differ in various aspects, posing unique security challenges for each device. For example, smart locks may face security issues like password cracking and fingerprint recognition attacks, while smart appliances may encounter concerns related to electrical safety and control signal security.

There is currently a limited amount of research focusing on smart home security, with most studies primarily concentrating on hardware design and passive defense measures. For instance, the authors of [3] propose a method for privacy risk assessment and risk control measures to address privacy security concerns. The authors of [4] describe a provable security authentication scheme to ensure the security of the smart home environment. These studies, which are passive defense mechanisms, can improve the security of smart homes to some extent, but they do not fully address all the security issues.

Intrusion detection, as a typical representative of active defense, is one of the critical technologies for safeguarding the security of smart home systems [5]. It overcomes the limitations of traditional network security techniques in terms of real-time responsiveness and dynamic adaptability. Monitoring and identifying abnormal behavior in network traffic enables timely detection and prevention of malicious attacks. Therefore, designing an efficient intrusion detection model is of paramount importance in ensuring the security of smart home systems. Traditional machine learning-based intrusion detection algorithms are relatively straightforward to train, widely adopted, and demonstrate high efficiency and reliability in practical applications [6,7]. On the other hand, intrusion detection algorithms based on deep learning exhibit superior detection performance, but their exceptional performance relies heavily on a significant amount of training data [8–10].

Currently, there is no comprehensive framework for research on smart home security, and it still faces several challenges [11,12]. Due to the varying attack frequencies of different devices in smart homes, there is an imbalance in collecting network traffic data, with some devices having a deficient proportion of attack data compared to normal data. The insufficient quantity of data makes it difficult to effectively train intrusion detection models, resulting in a decline in their performance [13,14]. Therefore, this paper proposes an enhancement method, EM-FEDE, applied to smart home intrusion detection in few-shot scenarios. Firstly, the EM-FEDE method analyzes the historical intrusion detection records of smart homes to determine whether there are features indicative of device types and data types in the captured data and then adaptively extends the features to achieve format alignment of device data. Secondly, the EM-FEDE method performs data cleaning operations to reduce noise and redundancy by removing duplicate entries and normalizing the data. Furthermore, the method adjusts the random sampling mechanism to ensure the diversity of the few-shot data obtained through sampling. Finally, the processed sampling data is used as input for the CWGAN, a variant of GAN that improves data generation through modifications in the loss function and optimization algorithms. The Wasserstein distance, which measures the dissimilarity between two probability distributions, is employed by CWGAN to calculate the loss between the generated data (fake data) and the real data. Based on this loss, the CWGAN is adjusted, and the generator of the CWGAN outputs effectively generates data. The main contributions of this paper are as follows:

- This paper proposes a feature enhancement module to improve the data quality in the dataset by analyzing historical intrusion detection records of smart homes, adaptively extending feature columns for the smart home devices dataset, and performing data cleaning on the dataset;
- This paper proposes a data enhancement module to generate valid data to populate the dataset using conditional Wasserstein GAN to realize the operation of data enhancement for few-shot data;
- The effectiveness of the EM-FEDE method is evaluated using a typical smart home device dataset, N-BaIoT. The performance of the original dataset and the expanded dataset using the EM-FEDE method on each intrusion detection model is compared to conclude that the classifier's performance is higher for the expanded dataset than the original dataset;
- The experiments demonstrate that expanding the dataset using the EM-FEDE method is crucial and effective in improving the performance of attack detection. This work successfully addresses the problem of few-shot data affecting the performance of intrusion detection models.

2. Related Works

2.1. Intrusion Detection Methods for Smart Homes

Intrusion detection methods for smart homes have gained significant attention in recent years as a popular research direction in the field of smart homes, and many scholars have conducted relevant research [15–17]. Many methods utilize sensors and network communication functions within smart home devices to detect intrusions by monitoring user behavior, device status, and other relevant data.

In 2021, the authors of [18] proposed an intrusion detection system that uses bidirectional LSTM recursive behavior to save the learned information and uses CNN to perfectly extract data features to detect anomalies in smart home networks. In 2021, the authors of [19] proposed a two-layer feature processing method for massive data and a three-layer hybrid architecture composed of binary classifiers in smart home environments to detect malicious attack environments effectively. In 2022, the authors of [20] proposed an intelligent two-tier intrusion detection system for the IoT. Using the feature selection module combined with machine learning, both flow-based and packet-based, it can minimize the time cost without affecting the detection accuracy. In 2023, the authors of [21] proposed an effective and time-saving intrusion detection system using an ML-based integrated algorithm design model. This model has high accuracy, better time efficiency, and a lower false alarm rate. In 2023, the authors of [14] proposed a transformer-based NIDS method for the Internet of Things. This method utilizes a self-attention mechanism to learn the context embedding of input network features, reducing the negative impact of heterogeneous features.

Even though numerous scholars have obtained commensurate outcomes pertaining to the issue of smart home security, such research endeavors were executed with ample data and did not consider the predicament of limited samples attributable to the shortage of data emanating from various devices in smart homes. As a result, it is difficult for intrusion detection models to assimilate the data feature, and the suggested models of the research endeavors above are unsuitable for situations involving few-shot data.

2.2. GAN-Based Data Enhancement Methods

In machine learning and deep learning, the size of the dataset is a critical factor affecting the performance of the model. However, obtaining large-scale labeled datasets will require a large workforce and resources. Researchers have been exploring data enhancement techniques to expand the dataset and improve model performance. Among these techniques, data enhancement methods using Generative Adversarial Networks (GAN) [22] proposed by Ian J. Goodfellow et al. in 2014 have gained significant attention. GAN-based data enhancement methods have shown promising results in enhancing the performance of intrusion detection models by generating generated data that can be used to supplement the limited labeled dataset.

GAN consists of a discriminator network and a generator network. The goal of the discriminant network is to accurately determine whether a sample is from real or fake data. The purpose of the generator network is to generate samples whose sources cannot be distinguished by the discriminant network. In GAN, the Generator uses random noise Z as input data, and its output is fake sample data $G(z)$. The discriminator receives real sample data x and fake sample data $G(z)$ as inputs and obtains loss by determining whether the data is real or fake by using the backpropagation algorithm to update the GAN parameters based on the loss function.

In recent years, more and more research has applied GAN for data enhancement to improve the performance and robustness of machine learning models. In 2021, the authors of [23] proposed using the ACGAN model to solve the problem of the imbalanced distribution of 1D intrusion detection sample data, which improved the average detection accuracy of some classification models. In 2021, the authors of [24] proposed an improved DCGAN model with higher stability and sample balance to achieve higher classification accuracy for a few samples. In 2022, the authors of [25] proposed a new generation of methods that use a class of classification models to determine the authenticity of facial images. This method improves cross-domain detection efficiency while maintaining source-domain accuracy. In 2023, the authors of [26] proposed an attention-self-supervised learning-aided classifier generative adversarial network algorithm to expand the samples to improve the defect recognition ability of small sample data sets. In 2023, the authors of [27] proposed a generative model for generating virtual marker samples by combining supervised variational automatic encoders with Wasserstein GAN with a gradient penalty. This model can significantly improve the prediction accuracy of soft sensor models for small-sample problems.

Although scholars have made many achievements using GAN for data enhancement, their applications are mainly carried out on images. In network security, there is still a lack of research on data enhancement using GAN. In addition, the implementation of GANs for data augmentation in the field of smart home intrusion detection has not been fully explored, thereby limiting their potential to solve problems in this field.

3. EM-FEDE Method

3.1. Problem Analysis

Figure 1 shows a typical smart home environment. A diverse array of smart home devices is linked to a gateway, which in turn is connected to the Internet via routers, and the data collected by these devices is subsequently sent to terminals for user analysis.

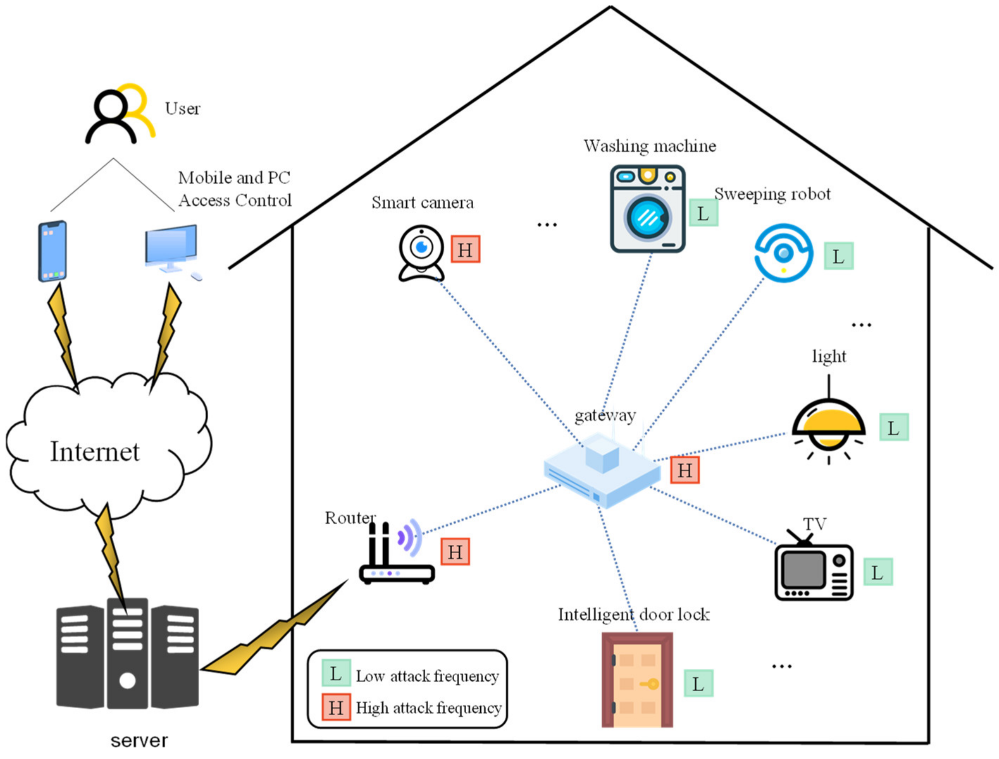


Figure 1. Smart home topology diagram.

Smart home devices differ in functionality and operational characteristics, exhibiting distinct working hours and data throughput. Attackers take various factors into account, such as device usage frequency and attack complexity, when exploiting vulnerabilities in different types of devices. This leads to the devices being subjected to varying frequencies of network attacks. The N-BaIoT dataset [28] of typical IoT devices shows the variance in data throughput and traffic collected among different devices. As depicted in Figure 2, a smart doorbell device (device1) and a smart camera device (device2) exhibit different data throughput, with device1 having a lower data throughput. Consequently, the amount of data collected by device1 is significantly less than that collected by device2 (338,599 vs.

1,075,936). Moreover, the number of data points generated by different attack behaviors also varies based on the attack frequencies of the devices. For instance, Figure 3 shows that attack1 (a UDP attack by the Gafgyt botnet) and attack2 (a UDP attack by the Mirai botnet) both utilize vulnerabilities to carry out DDoS attacks. However, attack2 is more effective and straightforward, resulting in a higher frequency of occurrence. Therefore, attack1 has far fewer data samples (255,111 vs. 1,229,999).

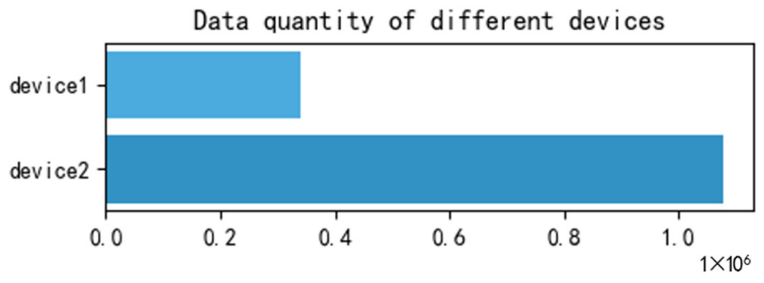


Figure 2. Number of data points from different devices.

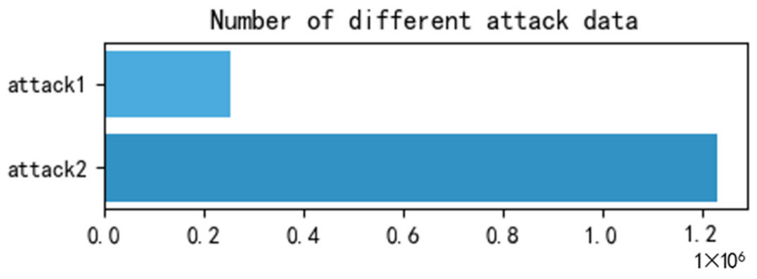


Figure 3. Number of different attack data.

Through the utilization of authentic datasets, information retrieval, and prior knowledge, the present study presents an account of the operational and safety conditions of various commonplace smart home devices in Table 1. The tabulation highlights that distinctive devices within the smart home setting exhibit assorted data throughput and attack frequencies. Additionally, diverse categories of smart devices are susceptible to differing attack behaviors, which results in a dissimilar amount of attack-related data. This situation leads to marked discrepancies in the data collected between various devices and attack types. For example, in the case of smart light bulbs, detecting and identifying attacks on these devices effectively is challenging due to the limited amount of attack data available. This scarcity of data is a result of the relatively low number of attacks that have been observed on this particular type of device. On the other hand, for smart door lock devices that experience a high frequency of attacks, more attack data is typically collected. However, there may still be instances of infrequent attack behaviors of a specific type (such as DDoS attacks commonly observed on smart cameras). These infrequent attack behaviors generate a small amount of attack data, which can be considered a sample size. As the tally of interconnected smart home devices continues to increase, these disparities become more prominent. Accordingly, during the process of flow data collection, specific devices are often unable to generate sufficient attack data, which impairs the efficacy of intrusion detection models. This limitation ultimately has a bearing on the overall security and stability of the smart home environment. Therefore, addressing the challenge of few-shot data resulting from a shortage of attack-related data is a critical research direction in the field of smart home device network security.

Table 1. Smart device working status table.

Devices	Working Hours (h)	Data Throughput	Frequency of Attack
Router	24	Larger	Higher
Gateway	20	Larger	Higher
Light	14	Smaller	Lower
TV	8	Larger	Lower
Intelligent door lock	3	Smaller	Lower
Floor sweeper	2	Smaller	Lower
Washing machine	2	Smaller	Lower
Smart camera	24	Larger	Higher

We propose the EM-FEDE method, as depicted in Figure 4. The method consists of three modules: the feature enhancement module, the data enhancement module, and the intrusion detection module. The feature enhancement module is assigned the task of processing the raw data by optimizing and filtering it. The data enhancement module focuses on generating samples, thereby expanding the dataset by adding fake samples. The intrusion detection module is responsible for identifying attacks and is trained using the expanded dataset, resulting in an improved ability of the model to classify and recognize various types of attacks. The EM-FEDE method employs symbols and their meanings, as listed in Table 2.

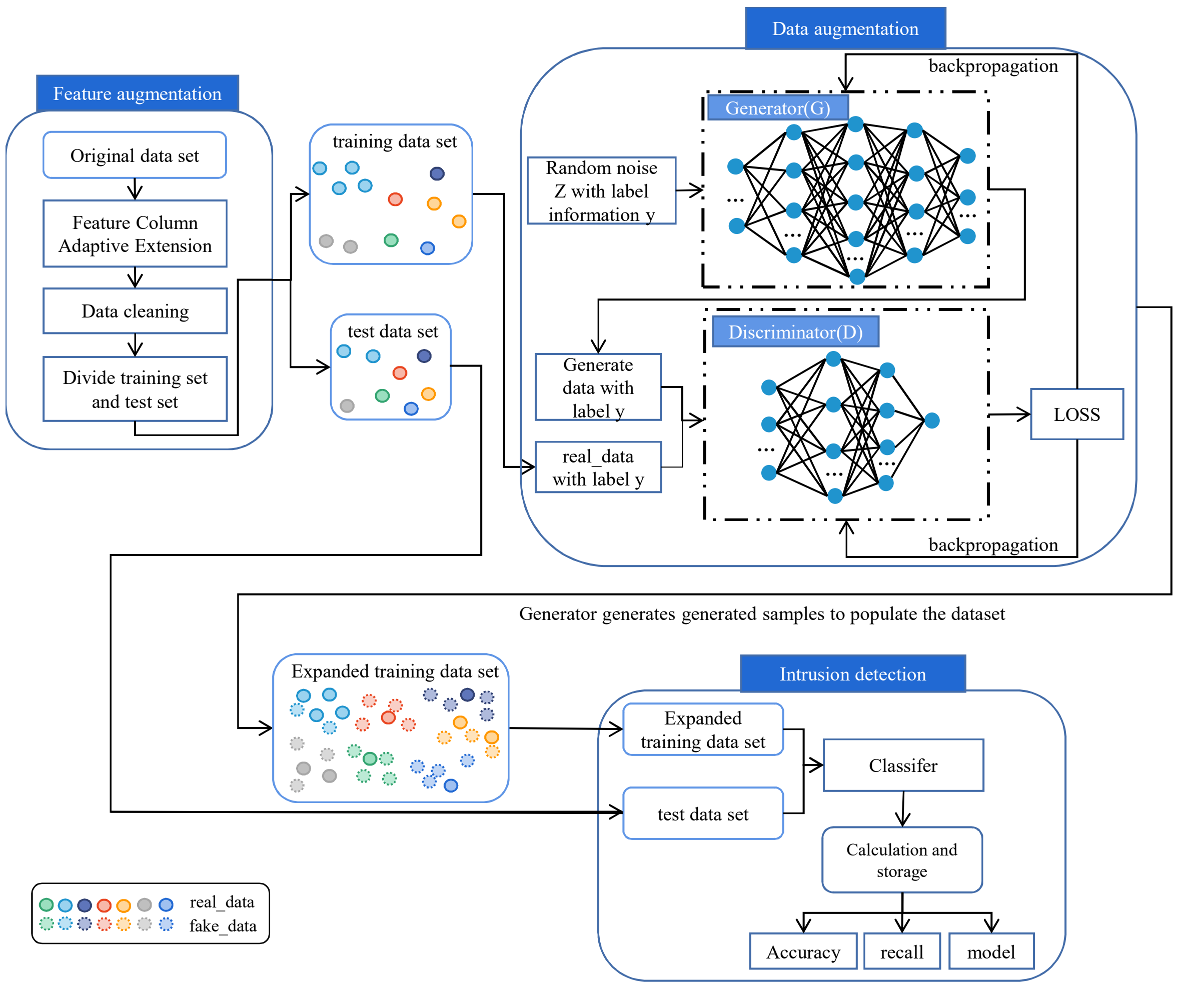


Figure 4. EM-FEDE method.

Table 2. Symbols used in the EM-FEDE method.

Symbols	Description
R	Historical intrusion detection records.
$SearchF(x)$	Used to determine the existence of the device class and the data class in x. It returns 1 if features are present and 0 otherwise.
Insert()	Insert operation.
$Class\_Label(x)$	Obtain the corresponding class from the information in x.
a_i	The device class feature column.
bi	The data class feature column.
$LabelEncoding(x)$	The function is used for mapping during the process of numericalization in x.
$FE\_Duplicate(x)$	The function is used for removing duplicate data in x.
$FE\_Normalization(x)$	The function is used for normalizing the data in x.
L	1-Lipschitz function.
P_{real}	Real data distribution.
P_z	Data distribution of input noise.
G(z)	Fake sample data generated by the generator.
$D(x)$	The probability that the discriminator determines that x belongs to the real data.
Z	Noise vector of the a priori noise distribution P_z.
$\prod(P_{real}, P_g)$	Joint probability distribution of real data and generated data.
$Fake\_data$	Generated data with label y_fake.

3.2. Feature Enhancement

This section focuses on the specific implementation of the EM-FEDE method in terms of feature enhancement.

In Section 3.1, it was discussed that various smart home devices exhibit differing data throughput and attack frequencies. Once the traffic data from these devices is captured, it is typically stored in a pcap file format. The format of the pcap file is shown in Figure 5. While the pcap file contains information such as timestamps, source addresses, and destination addresses, it lacks the ability to indicate device and data classes, resulting in the inability to label traffic. As a result, intrusion detection models that utilize supervised learning methods cannot directly utilize this data for training purposes. This challenge is also present in the N-BaIoT standard dataset, which includes eleven types of data collected from nine types of IoT devices (including one type of normal data and ten types of attack data). The dataset fails to provide feature columns that indicate the device class of each sample and distinguish the data class. To address the challenge of being unable to use raw data for training intrusion detection models and to improve data quality, this study proposes the EM-FEDE method's feature enhancement module. This module achieves feature enhancement through R analysis, feature-adaptive expansion, and data cleaning, thereby optimizing the data and indicating missing class features.

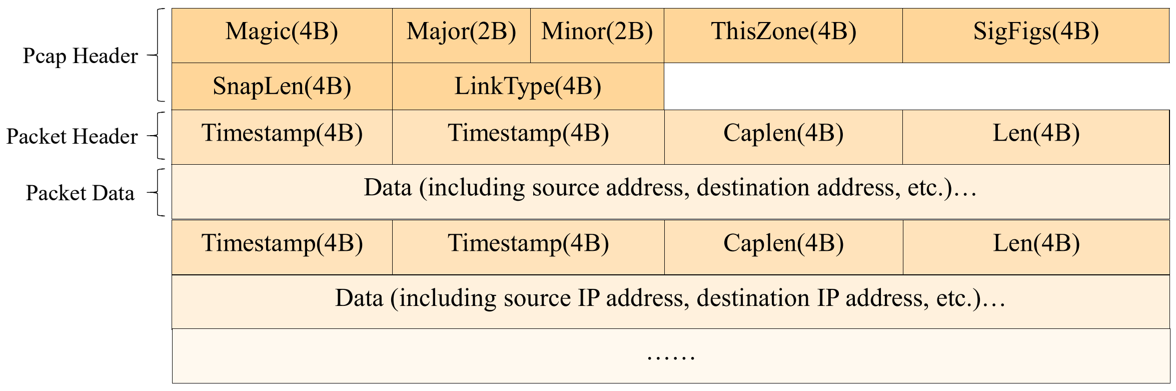


Figure 5. Common pcap file format.

To address the inability to directly use the raw data for training intrusion detection models and improve data quality, this study proposes the feature enhancement module of the EM-FEDE method. It achieves feature enhancement through *R* analysis, feature-adaptive expansion, and data cleaning. This process optimizes the data and enables the identification of missing class features.

The following is the specific process of feature enhancement:

Step 1. The *LF* is used to indicate the device class, and the data class in *R* is determined by Equation (1). If *LF* = 1, go to Step 3; if *LF* = 0, go to Step 2.

$$LF = SearchF(R); \tag{1}$$

Step 2. If direct prior knowledge (*E*) is available regarding the class of device and data, the device class feature (*ai*) and data class feature (*bi*) can be added to *R* through *E*. In the absence of such knowledge, the captured traffic data is analyzed to gather relevant information. As different attacks take place at different timestamps and distinct source IP addresses represent unique device characteristics, the timestamp and source IP address are treated as prior knowledge *E*. The device class feature (*ai*) and data class feature (*bi*) are then added to *R*, utilizing *E*. The specific equations pertaining to this process are illustrated in (2) and (3).

$$[ai, bi] = Class\_Label(E), \tag{2}$$

$$R = [R \cup Insert(ai) \cup Insert(bi)]; \tag{3}$$

Step 3. Numerical, de-duplication, and normalization of *R* by Equations (4)–(6).

$$R = LabelEncoding(R), \tag{4}$$

$$R = FE\_Duplicate(R), \tag{5}$$

$$R = FE\_Normalization(R), \tag{6}$$

In this step, we utilized Equation (4) to carry out numerical operations to convert non-numerical data in R into numerical data for the purpose of training the model. To tackle the problem of duplicate data, we applied Equation (5) to eliminate redundant data and minimize its impact on the results during data analysis. Additionally, we normalized the data using Equation (6) to ensure that all feature data was of the same magnitude and reduce the influence of noise on the results;

Step 4. Divide the training set and the test set, and output.

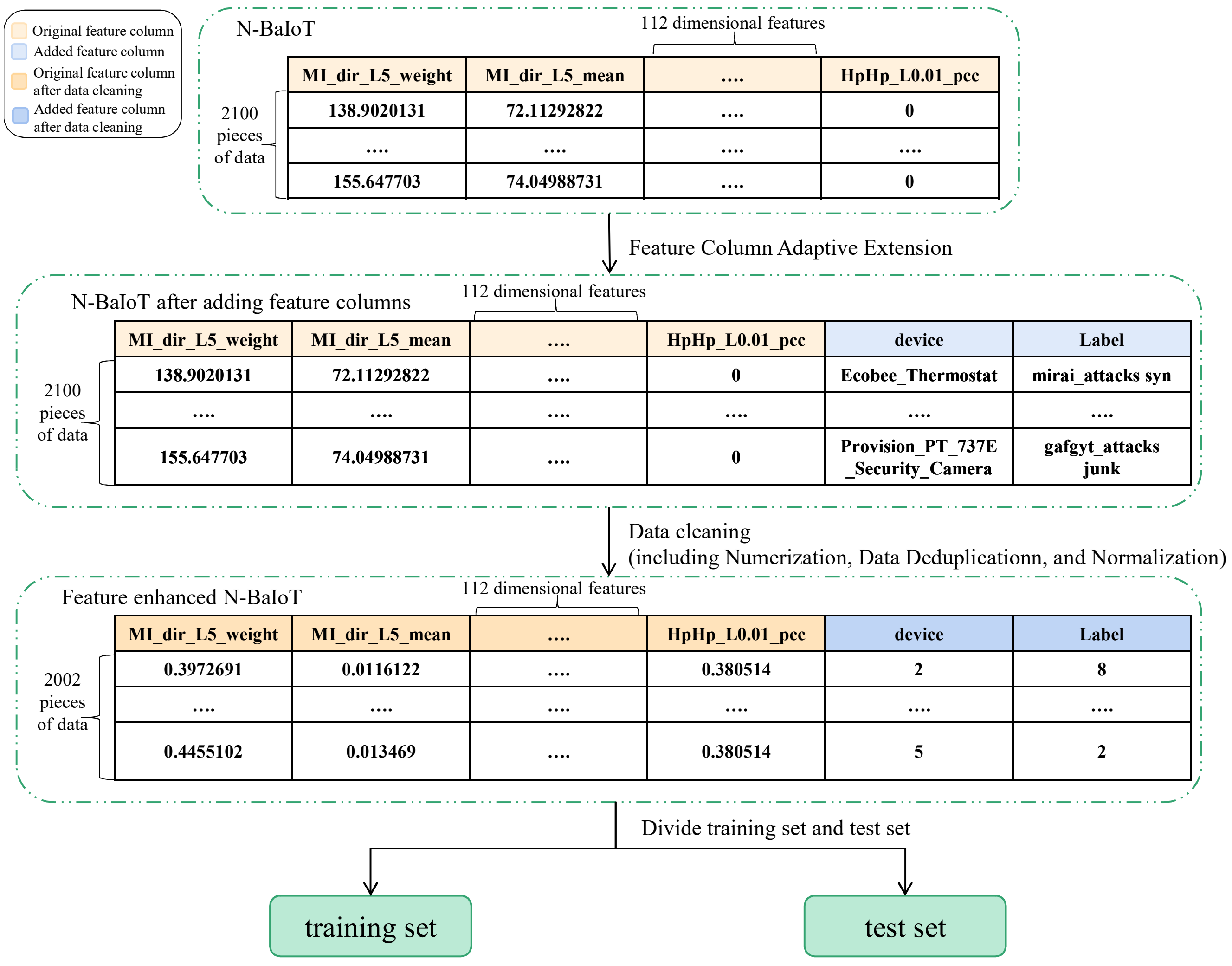
Figure 6 illustrates the process of feature enhancement using the N-BaIoT dataset as an example. The dataset contains *data1* = {138.9020131, 72.11292822, . . ., 0}, where 138.9020131 represents the value of *MI_dir_L5_weight*, 72.11292822 represents the value of *MI_dir_L5_mean*, 0 represents the value of *HpHp_L0.01_pcc*. There are 115 dimensional features in *data1*, and the specific steps are as follows:

Step 1. There is no common feature used to indicate the device class and attack class in N-BaIoT, *LF* = 0, so jump to Step 2;

Step 2. The N-BaIoT dataset contains prior knowledge *E* that enables us to determine the device class and data class of the dataset. Based on Equations (2) and (3), we added feature columns "*device*" and "*Label*". *Data1* corresponds to mirai_attacks syn attacks on the Ecobee_Thermostat device. As a result, we obtain *data1′* = {138.9020131, 72.11292822, . . ., 0,"Ecobee_Thermostat","mirai_attacks syn"};

Step 3. The obtained *data1′* contains non-numerical data, so it is numericalized by Equation (3) to obtain *data1′* = {138.9020131, 72.11292822, . . ., 0, 2, 8}, where 2 represents the value of *device*, 8 represents the value of *Label*. Then the *data1′* is de-duplicated and normalized by Equations (4) and (5), and finally the *data″′* = {0.3972691, 0.0116122, . . ., 0.380514, 2, 8} after feature enhancement is obtained;

Step 4. Output the training set and test set.



Original feature column
Added feature column
Original feature column after data cleaning
Added feature column after data cleaning
N-BaIoT
112 dimensional features
2100 pieces of data
Feature Column Adaptive Extension
N-BaIoT after adding feature columns
112 dimensional features
2100 pieces of data
Data cleaning (including Numerization, Data Deduplicationn, and Normalization)
Feature enhanced N-BaIoT
112 dimensional features
2002 pieces of data
Divide training set and test set
training set
test set

MI_dir_L5_weight	MI_dir_L5_mean	....	HpHp_L0.01_pcc
138.9020131	72.11292822	....	0
....	....	....	....
155.647703	74.04988731	....	0

MI_dir_L5_weight	MI_dir_L5_mean	....	HpHp_L0.01_pcc	device	Label
138.9020131	72.11292822	....	0	Ecobee_Thermostat	mirai_attacks syn
....	....	....	....	....	....
155.647703	74.04988731	....	0	Provision_PT_737E _Security_Camera	gafgyt_attacks junk

MI_dir_L5_weight	MI_dir_L5_mean	....	HpHp_L0.01_pcc	device	Label
0.3972691	0.0116122	....	0.380514	2	8
....	....	....	....	....	....
0.4455102	0.013469	....	0.380514	5	2

Figure 6. Dataset N-BaIoT feature enhancement process.

Feature enhancement is beneficial for reducing the superfluous information present in the data by processing and transforming the original features. This leads to the normalization of data, enhances its quality and usability, and provides a more dependable foundation for subsequent data enhancement and intrusion detection.

3.3. Data Enhancement

This section presents a detailed account of the practical implementation of data enhancement utilizing the EM-FEDE method. The data enhancement framework is shown in Figure 7.

Figure 7 is composed of three primary components. The first component is the input section, where the original data is enhanced by feature enhancement and utilized as input for the subsequent model training. The second component constitutes the CWGAN section. It contains two key components: the generator and the discriminator. The generator generates a variety of fake samples, while the discriminator is responsible for distinguishing between real and fake samples. The third component is the output section, where the generator produces diverse and authentic fake samples following iterative training of the CWGAN. These samples are then integrated into the original dataset, resulting in the enhanced dataset as the final output.

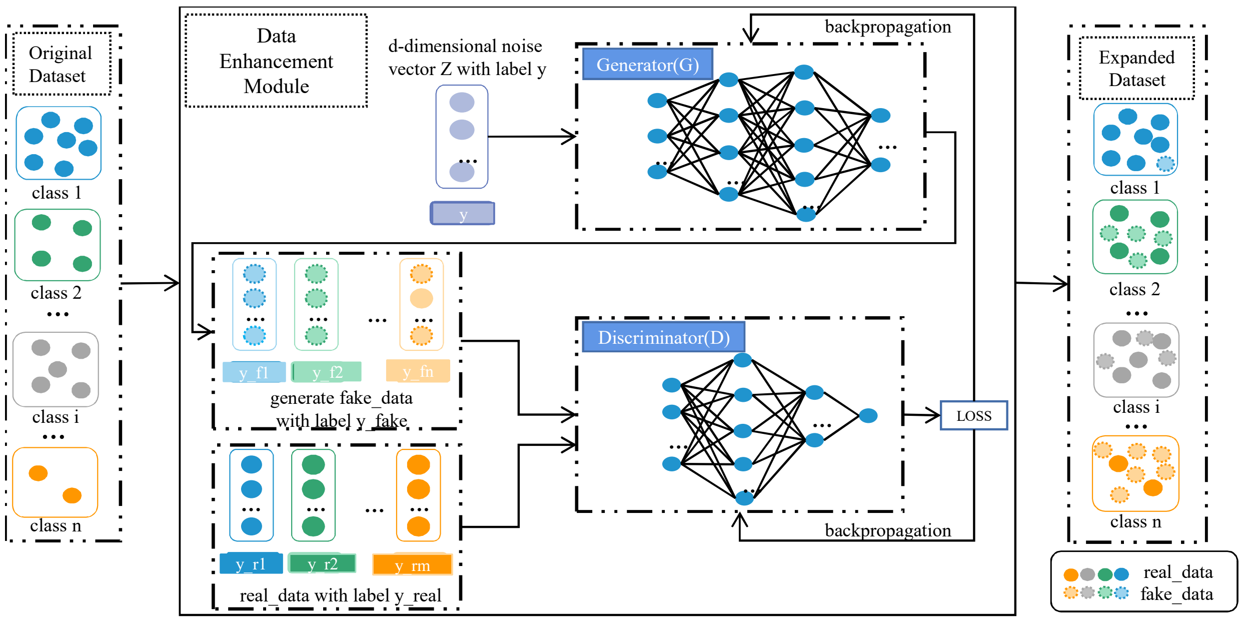


Figure 7. Data enhancement framework.

To quantify the disparity between the real data distribution and the fake data distribution, the EM-FEDE method employs the Wasserstein distance, which is expressed as Equation (7):

$$Wasserstein(P_{real}, P_z) = inf_{\gamma\sim\prod(P_{real},P_z)} E_{(x,y)\sim\gamma}[||x-y||], \tag{7}$$

That is, for any joint probability distribution γ there exist edge probability distributions P_{real} and P_g. Two sample points, x and y, can be sampled from the edge distribution, and the value of the Wasserstein distance is a lower boundary on the expectation of the x and y distances.

The Wasserstein distance is a metric that quantifies the dissimilarity between two probability distributions. It is smaller when the distributions are more similar. Even if the two distributions have no overlap, the Wasserstein distance can still be computed, unlike the Jensen–Shannon divergence, which cannot handle this case. This property has been leveraged by the EM-FEDE method, which incorporates the Wasserstein distance into the loss function of the CWGAN. As a result, the neural network structure is improved, and the objective function is represented by an equation:

$$L_{CWGAN} = E_{x\sim P_{real}}[D(x|y)] - E_{z\sim P_z}[D(G(z|y))], \tag{8}$$

where $D \in L$, x is the sample from the real data distribution, P_{real}, and y is the conditional variable, i.e., the class characteristics of the data.

The following are the main steps of the data enhancement process:

Step 1. The training set in R, after undergoing the feature enhancement process, is utilized as the training data for the CWGAN. The generator and discriminator, both of which employ multilayer perceptron models, are defined as two neural network models. Equation (8) is employed to determine the objective function of the EM-FEDE method;

Step 2. Training the discriminator. The process of training the discriminator is illustrated in Figure 8. It involves inputting a set of randomly generated *fake_data* samples and *real_data* samples of sizes n and m, respectively, into the discriminator. The loss values of both sets of data are computed using Equation (9) and subsequently used to update the discriminator's parameters:

$$\begin{aligned} Loss_{Discriminator} &= maxL_{CWGAN} = min(-L_{CWGAN}) \\ &= E_{z\sim P_z}[D(G(z|y))] - E_{x\sim P_{real}}[D(x|y)] \end{aligned}; \tag{9}$$

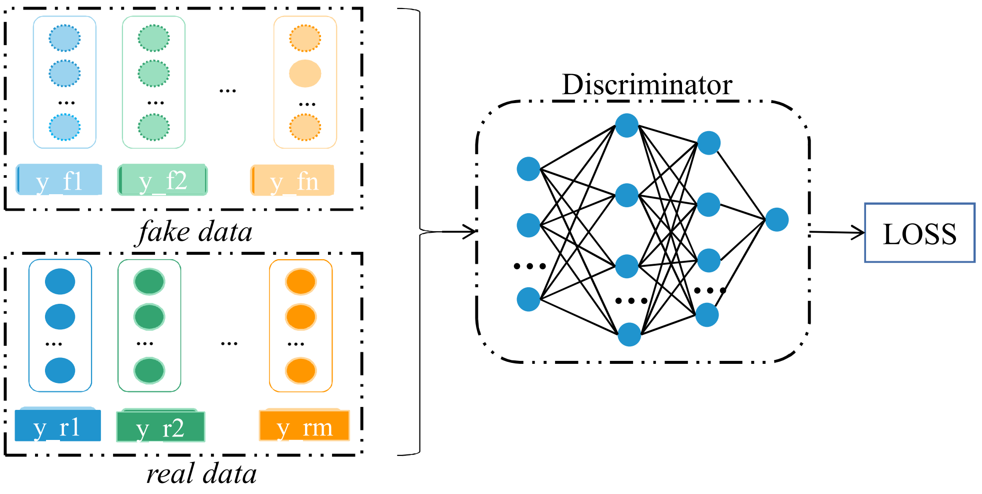


Figure 8. Discriminator process.

Step 3. Training the generator. The process of training the generator is illustrated in Figure 9. The generator is trained by generating a d-dimensional noise vector Z with label y as input, producing a set of *fake_data* samples of size n. These *fake_data* samples, along with the *real_data* samples, are then input into the discriminator. The loss value for this set of *fake_data* is computed using Equation (10), and the generator's parameters are updated accordingly.

$$Loss_{Generator} = min_{CWGAN} = -E_{z\sim P_z}[D(G(z|y))]; \tag{10}$$

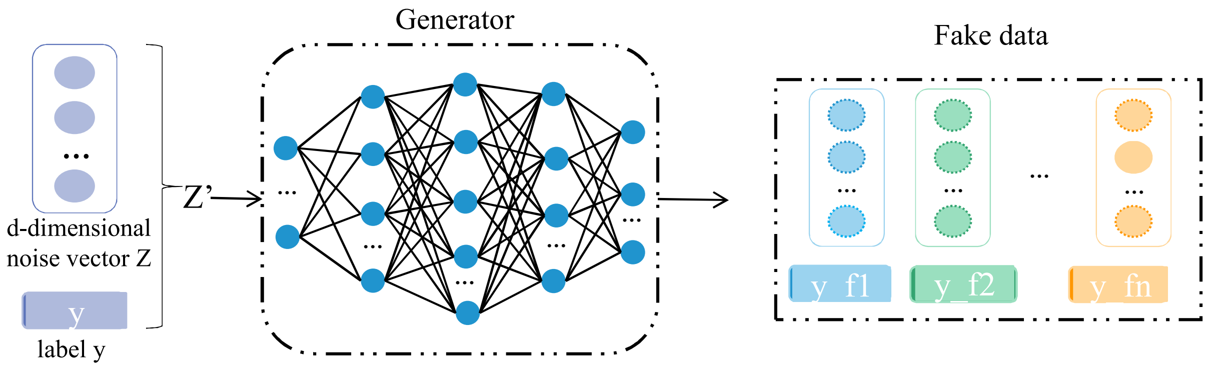


Figure 9. Generator process.

Step 4. The training process iterates Steps 2 and 3 repeatedly until the predetermined number of iterations or loss of convergence is reached. Conversely, by generating a new set of *fake_data* and returning *R* = [*R* ∪ *fake_data*].

Following the process of data enhancement, the imbalanced original dataset is enriched with fake data, effectively ensuring a more even distribution of data across all classes within the dataset.

In the EM-FEDE method, the computational cost of feature enhancement is negligible, so its computational complexity depends mainly on the CWGAN part of the data enhancement module. For the EM-FEDE method, the gradients of the generator and discriminator need to be computed and updated. In each epoch, O($|g_\omega| + |g_\theta|$) floating-point operations are required (where g_ω is the gradient of the generator and g_θ is the gradient of the discriminator), and thus its overall complexity is O($|R|\cdot(|g_\omega| + |g_\theta|)\cdot Ne$)(where $|R|$ is the training dataset and Ne is the total number of training times). Algorithm 1 gives the detailed algorithmic flow of the EM-FEDE method.

Algorithm 1: EM-FEDE

```
Input: $\alpha$ = 0.0005, the learning rate; $n$ = 50, the batch size; $c$ = 0.01, the clipping parameter; $\omega_0$, initial discriminator parameters; $\theta_0$, initial generator parameters; $Ne$ = 1000, the training cycles.
Output: Expanded $R$
Process:
1. Calculate $LF$ by Equation (1)
2. If $LF$ = 0
3. Add feature columns that are helpful for classification to $R$ through Equations (2)–(4)
4. Numerization, de-duplication, and normalization by Equations (5)–(7)
5. Divide the processed $R$ into training sets and test sets
6. End if
7. While $\theta$ has not converged or $epoch < Ne$ do
8. $epoch$++
9. Sample of $m$ noise samples$\{z_1, \ldots, z_n\} \sim P_Z$ a batch of prior data
10. Sample of $m$ examples$\{(x_1,y_1), \ldots, (x_n,y_n)\} \sim P_{real}$ a batch from the real data
11. Update the discriminator D by ascending its stochastic gradient ($g_\omega$)
12.     $g_\omega = \nabla\omega\left[\frac{1}{m}\sum_{i=1}^{m} f_\omega(x_i|y_i) - \frac{1}{m}\sum_{i=1}^{m} f_\omega(g_\theta(z_i|y_i))\right]$
13.     $\omega = \omega + \alpha * RMSProp(\omega, g_\omega)$
14.     $\omega = clip(\omega, -c, c)$
15. Sample of $m$ noise samples$\{z_1, \ldots, z_m\} \sim P_Z$ a batch of prior data.
16. Update the generator G by ascending its stochastic gradient ($g_\theta$)
17.     $g_\theta = -\nabla\theta\frac{1}{m}\sum_{i=1}^{m} f_\omega(g_\theta(z_i|y_i))$
18.     $\theta = \theta - \alpha * RMSProp(\theta, g_\theta)$
19. End while
20. Generate sample data for each class through the generator to populate $R$
21. Train the expanded $R$ on different classifiers to obtain various evaluation indicators
```

4. Results

4.1. N-BaIoT Dataset Description

The N-BaIoT dataset, released in 2018, consists of network traffic samples extracted from nine real IoT devices, featuring normal traffic from these devices and five varieties of attack traffic from the gafgyt and mirai botnet families. Figures 10 and 11 illustrate the differences in the data distribution across different traffic types and devices.

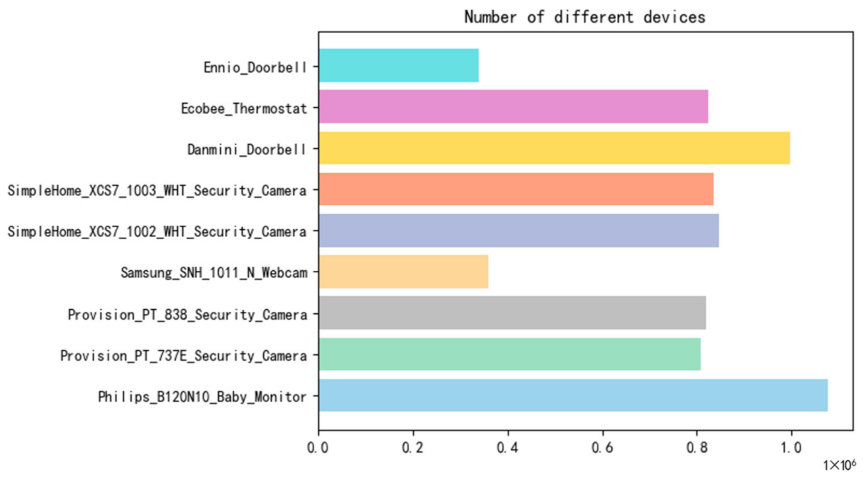


Figure 10. Distribution of the number of different devices.

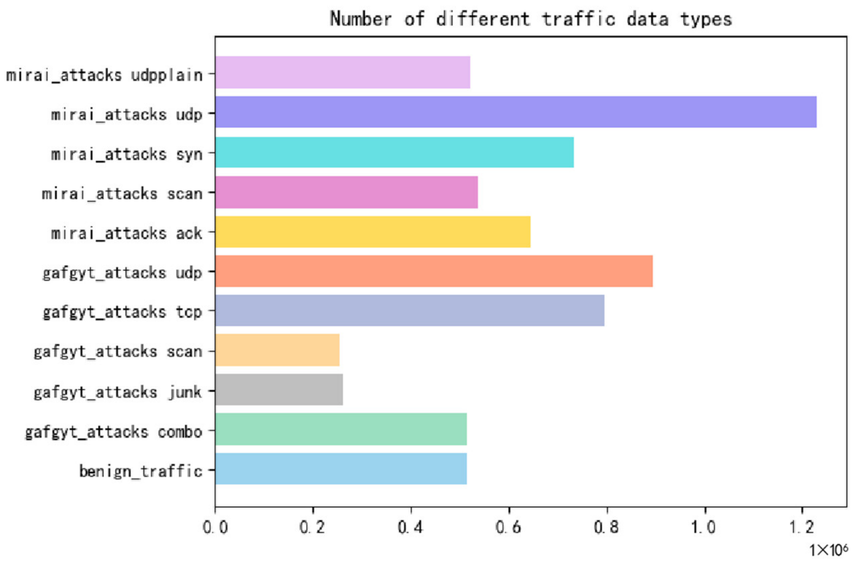


Figure 11. Distribution of the number of different traffic types.

The N-BaIoT dataset comprises extracted functions derived from raw IoT network traffic information. Upon receipt of each packet, a synopsis of the protocol and the host's behavior is computed with respect to the transmission of each packet. The contextual information of the data packet is then represented by a set of statistical features that are generated whenever a data packet arrives. Specifically, the arrival of each data packet leads to the extraction of 23 statistical features from five distinct time windows, namely, 100 ms, 500 ms, 1.5 s, 10 s, and 1 min. These five 23-dimensional vectors are subsequently concatenated into a single 115-dimensional vector.

The N-BaIoT dataset has been obtained in a real-world IoT setting, thus ensuring a high level of authenticity and representativeness. It serves as a standardized dataset that can be used by researchers to evaluate and enhance the efficacy of intrusion detection systems for IoT devices.

4.2. Data Preprocessing

The normalization method used in Equation (6) is Min–Max normalization. The specific formula for normalization is shown below:

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \tag{11}$$

where x_i is the current feature, x_{min} is the minimum eigenvalue in the same dimension, and x_{max} is the maximum eigenvalue in the same dimension.

To replicate the scarcity of data in smart home devices in the real world and to guarantee that the dataset gathered from sampling includes samples from all categories, this research employs stratified sampling. This method ensures that each sample has an equal opportunity to be selected while maintaining the randomness of the samples. Ultimately, 2860 data samples were randomly chosen from the dataset as representative examples. The training and test data were then divided in a 7:3 ratio, and the sample distribution of the training and test sets can be found in Table 3.

Table 3. Sample distribution of training and test sets.

Traffic Type Name	Number of Training Sets	Number of Test Sets
benign_traffic	1054	325
gafgyt_attacks combo	136	60
gafgyt_attacks junk	122	75

Table 3. *Cont.*

Traffic Type Name	Number of Training Sets	Number of Test Sets
gafgyt_attacks scan	124	78
gafgyt_attacks tcp	97	56
gafgyt_attacks udp	91	51
mirai_attacks ack	76	42
mirai_attacks scan	83	40
mirai_attacks syn	76	42
mirai_attacks udp	73	49
mirai_attacks udpplain	70	40

4.3. Experimental Environment

In order to verify the feasibility of the model in this paper, experiments were conducted in the experimental environment shown in Table 4.

Table 4. Experimental environment configuration.

Category	Parameters
CPU	Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz
RAM	64 GB
Programming Tools	Jupyter Notebook
Programming Languages	Python3.8
Deep Learning Framework	Pytorch1.8
Machine Learning Platform	Weka3.9
Data Processing Library	Numpy, pandas, etc.

4.4. Network Structure

The structural details of the CWGAN model designed in this article are presented in Table 5. The structural details of the MLP and CNN classifiers are shown in Table 6.

Table 5. Model parameters of the generator and discriminator.

G/D	Structure	Size
Generator	Input layer	50
	Hidden layer 1(Tanh())	128
	Hidden layer 2(Tanh())	256
	Hidden layer 3(Tanh())	128
	Output layer(Tanh())	116
Discriminator	Input layer	116
	Hidden layer 1(Tanh())	128
	Hidden layer 2(Tanh())	128
	Output layer	1

Table 6. Model parameters of the classifier.

Classifier	Structure	Size
MLP	Input layer	116
	Hidden layer 1(Tanh())	128
	Hidden layer 2(Tanh())	128
	Output layer	11
CNN	Input layer	116
	Conv1D(Relu())	32
	Pooling layer	32

Table 6. *Cont.*

Classifier	Structure	Size
CNN	Conv1D(Relu())	32
	Pooling layer	32
	Flatten	224
	Dense	50
	Dense	11

4.5. Results and Analysis

This study aimed to assess the effectiveness of the EM-FEDE method in enhancing the intrusion detection classifier. To accomplish this, machine learning and deep learning classification algorithms were employed to evaluate the dataset. The evaluation metrics used in this study, namely *Accuracy, Precision, Recall, and F1 Score*, are widely accepted in the field. The formulas for calculating each metric are provided below:

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP}, \tag{12}$$

$$Precision = \frac{TP}{FP + TP}, \tag{13}$$

$$Recall = \frac{TP}{FN + TP}, \tag{14}$$

$$F1\ Score = 2 * \frac{Prescision\ * \ Recall}{Prescision + Recall}, \tag{15}$$

where *TP* indicates the number of true positives; *TN* indicates the number of true negatives in the sample; *FN* indicates the number of false negatives; and *FP* indicates the number of false positives.

To test the effectiveness of the fake samples, the intrusion detection classifier was trained on both the original training set and the training set enhanced by the EM-FEDE method. Subsequently, the enhancement effect of the EM-FEDE method was evaluated by assessing the comprehensive classification performance of the intrusion detection classifier using the test set. Multiple sets of data were generated for experiments, each with different ratios of fake samples, as documented in Table 7.

Table 7. Sample size at different generated sample ratios.

Dataset (Generated Sample Ratios)	Number of Fake Samples	Number of Samples after Expansion
x (Original sample size)	0	2002
2x	2004	4006 (2002 + 2004)
3x	4006	6008 (2002 + 4006)
4x	6118	8120 (2002 + 6118)
5x	8010	10,012 (2002 + 8010)
6x	10,012	12,014 (2002 + 10,012)
7x	12,014	14,016 (2002 + 12,014)
8x	14,016	16,018 (2002 + 14,016)
9x	16,018	18,020 (2002 + 16,018)
10x	18,009	20,011 (2002 + 18,009)

The distributions of the original data training set and the expanded training set are shown in Figures 12 and 13, respectively, using a generated sample ratio of 5x as an example.

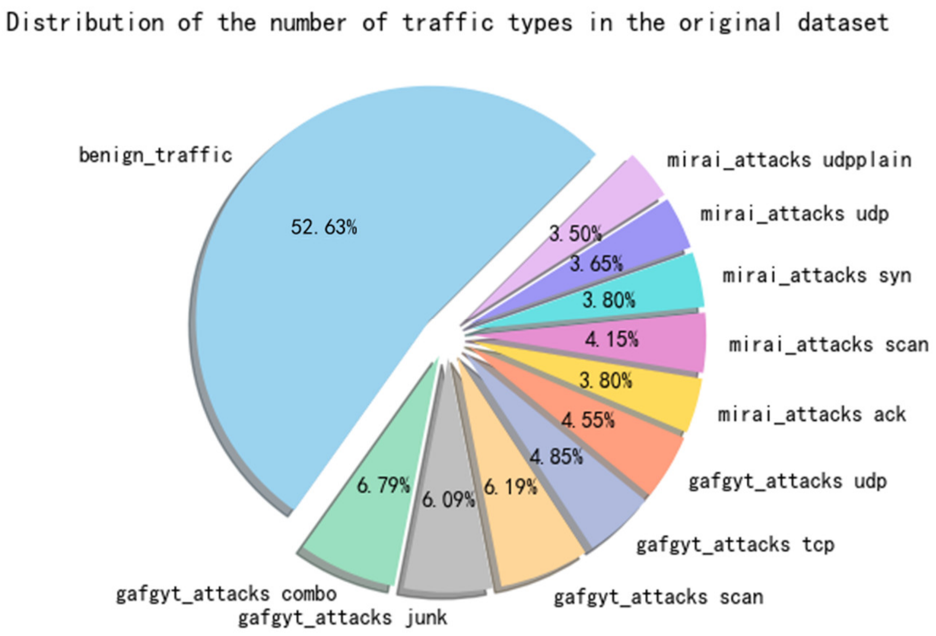


Figure 12. Data sample distribution of the original training set.

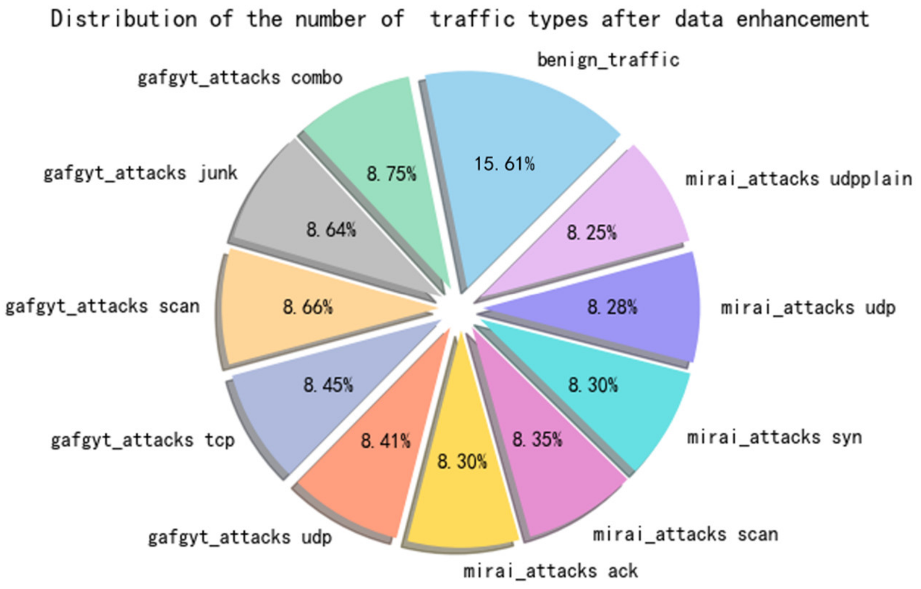


Figure 13. Sample distribution of training set data with a generated sample ratio of 5x.

The study conducted a comparison of multi-classification results between the original dataset and the generated sample ratio of 5x, as presented in Table 8. We evaluate the EM-FEDE method using various machine learning algorithms, where J48 is a decision tree algorithm, Random Forest is the Random Forest algorithm, Bagging is an integrated learning algorithm, PART is an algorithm that extracts rules in a dataset using incomplete decision trees, KStar is an instance-based classification algorithm, KNN is the K Nearest Neighbors algorithm, MLP is a Multi-Layer Perceptron Machine, and CNN is a Convolutional Neural Network. The results demonstrated that J48, Random Forest, Bagging, PART, KStar, KNN, MLP, and CNN showed an accuracy improvement of 16.4%, 4.9%, 10.7%, 9.2%, 4.9%, 4.4%, 3.1%, and 5.7%, respectively.

Table 8. Comparison of multi-classification results between the original dataset of size x and the mixed dataset with a generated sample ratio of $5x$ (the precision and F1 Score of some algorithms are unknown (?), which is due to the presence of Nan values in the calculation of precision, i.e., a denominator of 0). This scenario can occur when the algorithm fails to classify any sample into a particular class or when it wrongly classifies all samples in that class.

Dataset	Algorithm	Accuracy	Precision	Recall	F1 Score
N-BaIoT	J48	0.624	?	0.624	?
	Random Forest	0.755	?	0.756	?
	Bagging	0.655	?	0.655	?
	PART	0.673	?	0.673	?
	KStar	0.789	0.699	0.701	0.699
	KNN	0.768	0.773	0.768	0.771
	MLP	0.811	0.711	0.706	0.708
	CNN	0.712	0.726	0.673	0.698
N-BaIoT after EM-FEDE method processing	J48	**0.788**	**0.788**	**0.788**	**0.788**
	Random Forest	**0.804**	?	**0.804**	?
	Bagging	**0.762**	?	**0.762**	?
	PART	**0.765**	**0.795**	**0.765**	**0.779**
	KStar	**0.838**	**0.831**	**0.838**	**0.834**
	KNN	**0.812**	**0.796**	**0.812**	**0.803**
	MLP	**0.842**	**0.731**	0.678	0.703
	CNN	**0.769**	**0.828**	**0.736**	**0.779**

The evaluation of the experiments was carried out using various classification algorithms, including KNN, KStar, Bagging, PART, J48, Random Forest, MLP, and CNN. The evaluated results for datasets enhanced with different generated sample ratios are shown in Figure 14.

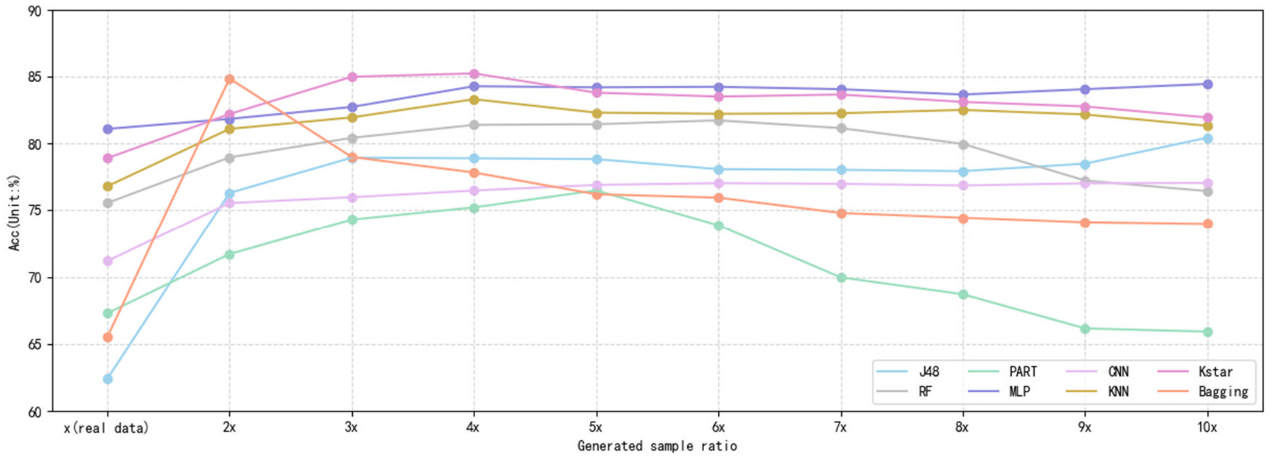


Figure 14. Experimental results.

As shown in Figure 14, the utilization of the EM-FEDE method has improved the accuracy of various classification algorithms. This improvement was observed when expanding the dataset compared to the original dataset. Additionally, the optimal sample ratio for achieving the best performance varies across different classification algorithms. With an increase in the number of generated samples, the accuracy of each classification algorithm gradually increases. The accuracy of J48 has increased from 62.39% (x) to 80.43% ($10x$), RF has increased from 75.55% (x) to 81.73% ($6x$), PART has increased from 67.28% (x) to 76.49% ($5x$), MLP has increased from 81.09% (x) to 84.45% ($10x$), CNN has increased from 71.18% (x) to 77.05% ($10x$), KNN has increased from 76.8% (x) to 83.31% ($4x$), KStar has increased from 78.9% (x) to 85.24% ($4x$), and Bagging has increased from 65.5% (x) to 84.86% ($2x$).

However, when the generated sample ratio becomes too large, the accuracy of some classification algorithms slightly decreases compared to a smaller generated sample ratio. The accuracy of RF decreases from 81.73% ($6x$) to 76.44% ($10x$), PART decreases from 76.49% ($5x$) to 65.91% ($10x$), KNN decreases from 83.31% ($4x$) to 81.32% ($10x$), KStar decreases from 85.24% ($4x$) to 81.94% ($10x$), and Bagging decreases from 84.86% ($2x$) to 73.98% ($10x$).

The accuracy of several classification algorithms such as RF, PART, KNN, KStar, and Bagging initially improves as the number of generated samples increases until they reach their optimal generated sample ratio, after which the accuracy decreases. This trend occurs due to the presence of fake data, which can negatively affect the quality of the data. The generator model aims to approximate the distribution of real data as closely as possible, but if the quantity of fake data becomes too large, the generator model can experience mode collapse. This phenomenon indicates that the fake data becomes excessively similar, and increasing the data further no longer improves the classifier's performance. Instead, it can lead to a decrease in classification accuracy due to noise in the fake data.

In contrast, J48, MLP, and CNN exhibit a gradual increase in accuracy. J48, a machine learning classifier based on feature partitioning, is typically sensitive to diversity and complexity. MLP and CNN, as deep learning classifiers, possess stronger representational and generalization capabilities. An increase in fake data leads to an increase in the training data for classifiers. This increase provides more opportunities for the classifiers to learn from different data distributions and features, leading to more complex and deeper levels of feature representation. Consequently, the classifiers' accuracy improves.

The variability in the best generated sample ratios is evident across different algorithms, as illustrated in Figure 14. Table 9 presents the accuracy of said ratios, in contrast to the original dataset, for various algorithms. The accuracy of J48, Random Forest, Bagging, PART, KStar, KNN, MLP, and CNN improved by 21.9%, 6.2%, 19.4%, 9.2%, 6.3%, 7%, 3.4%, and 5.9%, respectively. It is worth noting that the extended dataset demonstrated an overall higher accuracy in comparison to the original dataset when scaled to the best generated sample ratio of each algorithm.

Table 9. The accuracy of multi-classification is compared between the original data set and the mixed data set with the optimal generation sample ratio of each algorithm.

Algorithm	**Optimal Generation Sample Ratio nx ($1 \leq n \leq 10$)**	**Accuracy of the Original Dataset**	**Accuracy of the Mixed Dataset with the Optimal Generation Sample Ratio**	**The Percentage of Growth**
J48	$10x$	0.624	0.843	21.9%
Random Forest	$6x$	0.755	0.817	6.2%
Bagging	$2x$	0.655	0.849	19.4%
PART	$5x$	0.673	0.765	9.2%
KStar	$4x$	0.789	0.852	6.3%
KNN	$4x$	0.768	0.833	7%
MLP	$10x$	0.811	0.845	3.4%
CNN	$10x$	0.712	0.771	5.9%

SMOTE [29] is an oversampling method that generates new samples to expand the dataset based on the relationship between samples, and CGAN [30] is an extension of GAN for conditional sample generation. This part of the experiment examined the impact of different generated sample ratios on accuracy in J48 and Bagging for mixed datasets created using SMOTE, CGAN, and the proposed method. Additionally, we compared it with the same number of datasets containing only real data to prove the effectiveness of the proposed method in this paper. The experimental results are shown in Figures 15 and 16.

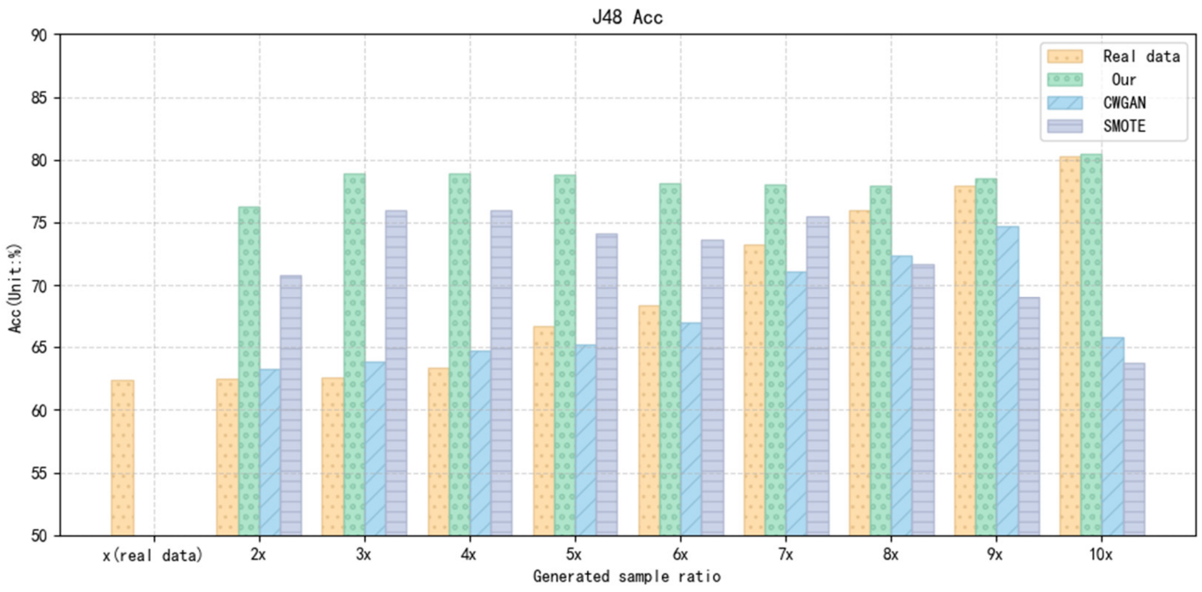


Figure 15. Accuracy of J48 for real and expanded datasets.

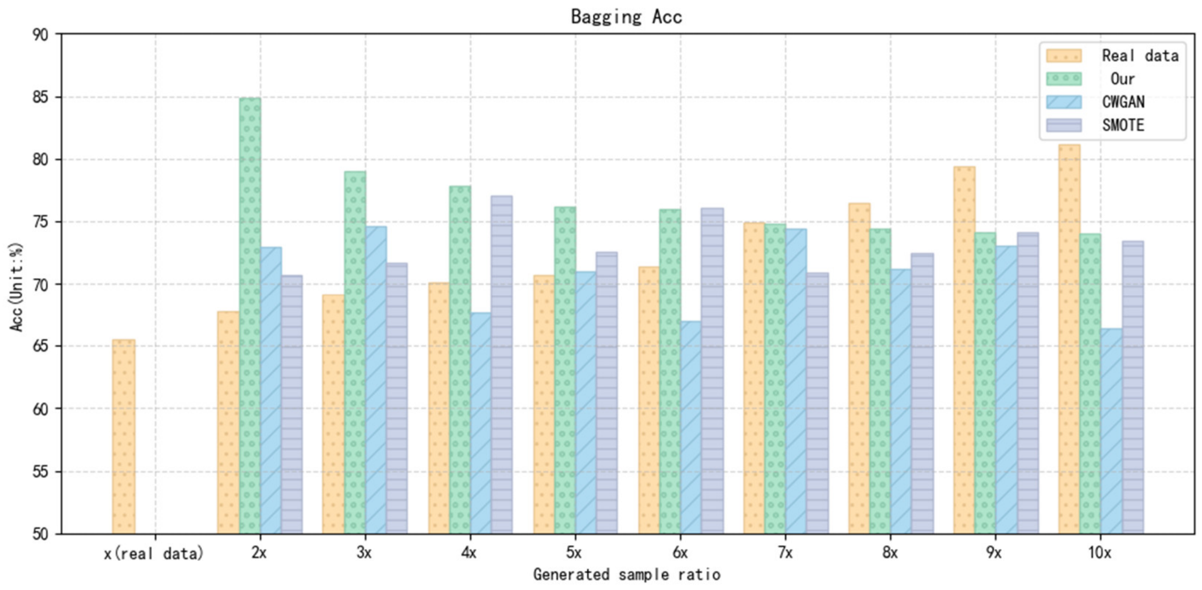


Figure 16. Accuracy of Bagging for real and expanded datasets.

Based on the results presented in Figure 15, in our method, it is evident that the accuracy of the enhanced dataset, which includes a combination of fake data and real data at the generated sample ratios of nx (n = 2, 3, . . ., 10), is superior to that of an equivalent number of instances from the original dataset. At lower generated sample ratios, the mixed dataset exhibits significantly improved accuracy on J48 in comparison to an equivalent number of instances from the real dataset. As the generated sample ratio increases, the accuracy of the mixed dataset on J48 exhibits fluctuation, albeit within a small range, and ultimately reaches a plateau. Although the mixed dataset continues to outperform the real dataset in terms of accuracy on J48, its advantage diminishes as the generated sample ratio becomes larger.

Regarding the CWGAN, at lower generated sample ratios nx (n = 2, 3, 4), the accuracy of the mixed dataset in J48 slightly improves compared to the same number of instances of the real dataset. However, for generated sample ratios of nx (n = 4, . . ., 10), the accuracy of the mixed dataset at J48 is lower than that of the equivalent number of real datasets, and the performance of the real dataset is significantly better than that of the mixed dataset as the generated sample ratio increases.

Regarding the SMOTE, at the generated sample ratio of *nx* (n = 2, . . ., 7), the accuracy of the mixed dataset in J48 is significantly higher than that of the real dataset with the same number of samples. At the generated sample ratio of *nx* (n = 8, 9, 10), the accuracy of the hybrid dataset starts to decrease and is lower than the equivalent number of real datasets.

Based on the experimental results, we can conclude that the J48 algorithm has more capacity for learning the supplementary feature information that is provided by the expanded dataset. This attribute of the algorithm contributes to an improved understanding of the dataset's traits and patterns, thereby leading to an enhancement of the classifier's performance. In addition to this, the introduction of a small quantity of artificial data has been observed to have a beneficial effect on the model's ability to generalize, and it can also serve to mitigate the effects of overfitting and noisy data. However, it should be noted that there is a threshold beyond which the quantity of artificially generated data becomes sufficient, and further increments of such data do not yield any improvement in the accuracy of the intrusion detection model.

The results in Figure 16 show that the accuracy of the mixed dataset with generated sample ratio *nx* (n = 2, . . ., 6) on the Bagging algorithm is better than that of the corresponding number of real datasets in the method of this paper. However, for the generated sample ratio *nx* (n = 7, . . ., 10), the accuracy of the mixed dataset is lower than that of the corresponding number of real datasets. The experimental results reveal that the optimal generated sample rate for the Bagging algorithm using the method in this paper is $2x$. Moreover, the accuracy of Bagging decreases and stabilizes as the generated sample rate increases.

Regarding the CWGAN, the accuracy of the mixed dataset is higher than the same number of instances of the real dataset for the generation sample rate *nx* (n = 2, 3, 5). However, for the generating sample ratio of *nx* (n = 4, 6, . . ., 10), the accuracy of the mixed dataset is lower than the accuracy of the same number of real datasets. The results indicate that the best generated sample ratio for the Bagging algorithm using the CWGAN is $3x$.

Regarding the SMOTE, the accuracy of the mixed dataset is higher for the generation sample ratio *nx* (n = 2, . . ., 6) compared to the same number of instances of the real dataset. For the generation sample ratio *nx* (n = 7, . . ., 10), the accuracy of the mixed dataset is lower than the accuracy of the same number of instances of the real dataset. From the experimental results, it can be concluded that the optimal generation sample ratio for Bagging on SMOTE is $4x$.

When the generated sample ratio *nx* (n = 7, . . ., 10) is too large, the accuracy of both the methods in this paper, CWGAN and SMOTE on Bagging, is lower than the equivalent number of real datasets. Despite the decrease in accuracy, the accuracy of this paper's method and SMOTE is still higher than that of the original dataset x. By comparing this paper's method, CWGAN, and SMOTE, it can be concluded that this paper's method exhibited better performance.

Based on our experimental results, we can conclude that utilizing fake data for data enhancement can significantly enhance the accuracy of the classifier, particularly when the expansion multiplier is small. However, the introduction of fake data may result in noise, and its proportion increases with the expansion multiplier. This difference between real and fake data can make it challenging to provide sufficient useful feature information, which can, in turn, impede the ability of the model to learn the data features. Ultimately, this can lead to a reduction in the accuracy of the classifier.

The SMOTE algorithm analyzes the minority class samples and manually synthesizes new samples to add to the dataset based on the minority class samples. This technique of generating new samples through oversampling helps prevent overfitting. However, it may generate the same number of new samples for each minority class sample, resulting in increased overlap between classes and the creation of samples that do not offer useful information. The CGAN method improves the data generation process by incorporating additional information to guide the model. However, the training process of CGAN is not very stable, and the quality of the generated data can vary. In contrast, the EM-FEDE method proposed in this paper uses the CWGAN approach to generate data with greater diversity. It also provides more informative samples and is more stable during training, resulting in higher-quality generated data compared to CGAN. To summarize, the effectiveness of the EM-FEDE method has been demonstrated, making it suitable for training datasets for intrusion detection models. However, it is crucial to consider that the optimal generated sample ratio may differ based on the particular algorithm and model in use. To attain the highest level of accuracy and performance for a given intrusion detection algorithm or model, it is essential to undertake a meticulous evaluation and selection of the most fitting generated sample ratio. This selection and evaluation process is crucial to guaranteeing optimal outcomes.

5. Discussion

The present article discusses the issue of few-shot data on smart home devices and the challenges this poses for intrusion detection models. Specifically, the study highlights how the security dataset collected from traffic information often lacks data, which limits the performance of intrusion detection models. To address this issue, the article proposes a method called EM-FEDE, which enhances the dataset and effectively mitigates the impact of few-shot data on intrusion detection performance, improving security in smart home environments. The study evaluates the performance of datasets enhanced with different generated sample ratios and analyzes the effect of using enhanced datasets for intrusion detection model training. Furthermore, the article examines the influence of different generated sample ratios on classification performance for specific classification algorithms. The results indicate that the optimal generated sample ratio may vary depending on the algorithm and model used. Based on the obtained results, it can be concluded that the proposed method shows promising performance in solving few-shot data. In addition to intrusion detection, it can be applied to different domains, such as sentiment analysis tasks where the samples of various sentiment categories are highly imbalanced and underwater target recognition tasks where the samples are too small to train an effective model.

In this paper, the specific details regarding the optimal expansion multiplier and the ratio of generated data to real data for various classification algorithms are not extensively explored. Thus, future studies will focus on optimizing the intrusion detection model by selecting more suitable classification algorithms to enhance detection accuracy. Additionally, further research will be conducted to determine the appropriate enhancement factors and ratios between generated and real data during the data enhancement process.

Author Contributions: Conceptualization, T.Y. and J.W.; methodology, Y.C. and J.W.; software, Y.C., T.Y. and J.W.; validation, J.W., T.Y. and Y.C.; formal analysis, J.W.; investigation, J.W.; resources, Y.C.; data curation, J.W.; writing—original draft preparation, J.W. and T.Y.; writing—review and editing, J.W., Q.L. and N.A.N.; supervision, T.Y. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Sichuan Science and Technology Program under Grant No. 2022YFG0322, China Scholarship Council Program (Nos. 202001010001 and 202101010003), the Innovation Team Funds of China West Normal University (No. KCXTD2022-3), the Nanchong Federation of Social Science Associations Program under Grant No. NC22C280, and China West Normal Universi-ty 2022 University-level College Student Innovation and Entrepreneurship Training Program Project under Grant No. CXCY2022285.

Data Availability Statement: Data are unavailable due to privacy.

Acknowledgments: Thanks to everyone who contributed to this work.

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

References

1. Cvitić, I.; Peraković, D.; Periša, M.; Jevremović, A.; Shalaginov, A. An Overview of Smart Home IoT Trends and related Cybersecurity Challenges. *Mob. Netw. Appl.* **2022**. [CrossRef]
2. Hammi, B.; Zeadally, S.; Khatoun, R.; Nebhen, J. Survey on smart homes: Vulnerabilities, risks, and countermeasures. *Comput. Secur.* **2022**, *117*, 102677. [CrossRef]
3. Wang, Y.; Zhang, R.; Zhang, X.; Zhang, Y. Privacy Risk Assessment of Smart Home System Based on a STPA–FMEA Method. *Sensors* **2023**, *23*, 4664. [CrossRef] [PubMed]
4. Wu, T.Y.; Meng, Q.; Chen, Y.C.; Kumari, S.; Chen, C.M. Toward a Secure Smart-Home IoT Access Control Scheme Based on Home Registration Approach. *Mathematics* **2023**, *11*, 2123. [CrossRef]
5. Li, Y.; Zuo, Y.; Song, H.; Lv, Z. Deep learning in security of internet of things. *IEEE Internet Things J.* **2021**, *9*, 22133–22146. [CrossRef]
6. Chkirbene, Z.; Erbad, A.; Hamila, R.; Gouissem, A.; Mohamed, A.; Guizani, M.; Hamdi, M. A weighted machine learning-based attacks classification to alleviating class imbalance. *IEEE Syst. J.* **2020**, *15*, 4780–4791. [CrossRef]
7. Zivkovic, M.; Tair, M.; Venkatachalam, K.; Bacanin, N.; Hubálovský, Š.; Trojovský, P. Novel hybrid firefly algorithm: An application to enhance XGBoost tuning for intrusion detection classification. *PeerJ Comput. Sci.* **2022**, *8*, e956.
8. Li, X.K.; Chen, W.; Zhang, Q.; Wu, L. Building auto-encoder intrusion detection system based on random forest feature selection. *Comput. Secur.* **2020**, *95*, 101851.
9. Wang, Z.; Liu, Y.; He, D.; Chan, S. Intrusion detection methods based on integrated deep learning model. *Comput. Secur.* **2021**, *103*, 102177. [CrossRef]
10. Tsimenidis, S.; Lagkas, T.; Rantos, K. Deep learning in IoT intrusion detection. *J. Netw. Syst. Manag.* **2022**, *30*, 8. [CrossRef]
11. Heartfield, R.; Loukas, G.; Budimir, S.; Bezemskij, A.; Fontaine, J.R.; Filippoupolitis, A.; Roesch, E. A taxonomy of cyber-physical threats and impact in the smart home. *Comput. Secur.* **2018**, *78*, 398–428. [CrossRef]
12. Touqeer, H.; Zaman, S.; Amin, R.; Hussain, M.; Al-Turjman, F.; Bilal, M. Smart home security: Challenges, issues and solutions at different IoT layers. *J. Supercomput.* **2021**, *77*, 14053–14089. [CrossRef]
13. Cao, X.; Luo, Q.; Wu, P. Filter-GAN: Imbalanced Malicious Traffic Classification Based on Generative Adversarial Networks with Filter. *Mathematics* **2022**, *10*, 3482. [CrossRef]
14. Wang, M.; Yang, N.; Weng, N. Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System. *Electronics* **2023**, *12*, 2100. [CrossRef]
15. Guebli, W.; Belkhir, A. Inconsistency detection-based LOD in smart homes. *Int. J. Semant. Web Inf. Syst. IJSWIS* **2021**, *17*, 56–75. [CrossRef]
16. Madhu, S.; Padunnavalappil, S.; Saajlal, P.P.; Vasudevan, V.A.; Mathew, J. Powering up an IoT-enabled smart home: A solar powered smart inverter for sustainable development. *Int. J. Softw. Sci. Comput. Intell. IJSSCI* **2022**, *14*, 1–21. [CrossRef]
17. Tiwari, A.; Garg, R. Adaptive Ontology-Based IoT Resource Provisioning in Computing Systems. *Int. J. Semant. Web Inf. Syst. IJSWIS* **2022**, *18*, 1–18. [CrossRef]
18. Elsayed, N.; Zaghloul, Z.S.; Azumah, S.W.; Li, C. Intrusion detection system in smart home network using bidirectional lstm and convolutional neural networks hybrid model. In Proceedings of the 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 9–11 August 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 55–58.
19. Shi, L.; Wu, L.; Guan, Z. Three-layer hybrid intrusion detection model for smart home malicious attacks. *Comput. Electr. Eng.* **2021**, *96*, 107536. [CrossRef]
20. Alani, M.M.; Awad, A.I. An Intelligent Two-Layer Intrusion Detection System for the Internet of Things. *IEEE Trans. Ind. Inform.* **2022**, *19*, 683–692. [CrossRef]
21. Rani, D.; Gill, N.S.; Gulia, P.; Arena, F.; Pau, G. Design of an Intrusion Detection Model for IoT-Enabled Smart Home. *IEEE Access* **2023**, *11*, 52509–52526. [CrossRef]
22. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 2672–2680.
23. Fu, W.; Qian, L.; Zhu, X. GAN-based intrusion detection data enhancement. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 2739–2744.
24. Zhang, L.; Duan, L.; Hong, X.; Liu, X.; Zhang, X. Imbalanced data enhancement method based on improved DCGAN and its application. *J. Intell. Fuzzy Syst.* **2021**, *41*, 3485–3498. [CrossRef]
25. Li, S.; Dutta, V.; He, X.; Matsumaru, T. Deep Learning Based One-Class Detection System for Fake Faces Generated by GAN Network. *Sensors* **2022**, *22*, 7767. [CrossRef] [PubMed]
26. Yang, W.; Xiao, Y.; Shen, H.; Wang, Z. An effective data enhancement method of deep learning for small weld data defect identification. *Measurement* **2023**, *206*, 112245. [CrossRef]
27. Jin, H.; Huang, S.; Wang, B.; Chen, X.; Yang, B.; Qian, B. Soft sensor modeling for small data scenarios based on data enhancement and selective ensemble. *Chem. Eng. Sci.* **2023**, *279*, 118958. [CrossRef]
28. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2019**, *17*, 12–22. [CrossRef]

29. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
30. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.