Hindawi
Security and Communication Networks
Volume 2022, Article ID 1761655, 12 pages
https://doi.org/10.1155/2022/1761655

Research Article

LGBM: An Intrusion Detection Scheme for Resource-Constrained End Devices in Internet of Things

Yong-Quan Cong,[1] **Ting Guan**,[2] **Ju-Fu Cui**,[3] **and Xiang-Guo Cheng**[4]

[1]*Qingdao Academy of Recruitment and Examination, Qingdao, China*
[2]*Educational Equipment & ICT Center, Qingdao, China*
[3]*Bank of Qingdao, Qingdao, China*
[4]*College of Computer Science and Technology, Qingdao University, Qingdao, China*

Correspondence should be addressed to Xiang-Guo Cheng; chengxg@qdu.edu.cn

Received 17 August 2022; Revised 19 September 2022; Accepted 28 September 2022; Published 11 October 2022

Academic Editor: Kai Wang

The intrusion detection schemes (IDSs) based on the Gradient Boosting Decision Tree (GBDT) face three problems: unbalanced training data distribution, large dimensionality of data features, and difficulty in model parameter optimization, which lead to weak monitoring capability and high false positive rate. For the problem of unbalanced training data distribution, we make the one-sided gradient oversampling algorithm to ensure the balance between the data of each category. To tackle the problem of the large dimensionality of data features, we develop a hierarchical cross-validation algorithm for binding mutually exclusive features. To address the problem of difficulty in model parameter optimization, we design a Bayesian optimization algorithm to make the model parameter search process more targeted and reduce the model training cost by establishing functional relationships between hyperparameters and target functions. The detailed experimental results show that the scheme can effectively solve the problems of data imbalance, high-dimensional data features, and low parameter finding efficiency, and improve the model's ability to monitor the attack behavior.

1. Introduction

Internet of Things (IoT) [1, 2] incorporates various types of acquisition or control sensors as well as mobile communications, intelligent analytics, and other technologies into various aspects of industrial production processes, making a large number of resource-constrained end devices gradually becoming first-class network entities [3, 4]. Compared to personal computers and cloud servers with large amounts of computing resources, end devices are usually close to the user side or in the transmission path and have a higher likelihood of being compromised by attackers. For example, an attacker can perform a side-channel attack on end devices by monitoring common information such as the time consumption and power consumption of end devices. Intrusion detection schemes are one of the most well-known security protection techniques in the traditional Internet domain [5–7]. However, since emerging resource-constrained network entities usually have limited computing power or insufficient power supply, mainstream intrusion detection techniques are hardly as effective as they were in the past. Therefore, it is necessary to design lightweight intrusion detection techniques to protect the security of resource-constrained end devices in IoT.

IDSs are mainly divided into two categories: traditional detection schemes and machine learning-based detection schemes. Traditional detection schemes suffer from weak monitoring capability [8, 9], high false positive rates [10, 11], difficult feature information collection [12], etc. To cope with these problems of traditional detection schemes, various IDSs based on machine learning [13] have been proposed one after another. These detection schemes first use machine learning algorithms to learn known attack types and then use training models to identify attacks with corresponding features. It can be broadly classified into the following two categories: (1) IDSs based on a single machine learning algorithm. Lippmann et al. [14] used a neural network composed of multilayer perceptions without hidden units to construct an anomaly detection system. The number of keyword occurrences in the Telnet session is first used as input to the neural network, and then the instances that are flagged as attacks are used as training data to train the multilayer perceptual neural network. Bivens et al. [15] used the TCP/IP data from DARPA to construct an anomaly detection system based on the multilayer perceptual neural network. It uses time windows to detect multiple packets as a group. However, the applicability of this scheme is very limited and the constructed model is simple and cannot handle large data volume, which leads to degradation of the model performance and has been rarely used in recent years. (2) IDS based on the integrated learning algorithm. Mousavi et al. [16] combined the grid search algorithm to reduce the number of input data and normal data matching the number of matches. Arif et al. [17] improved the recognition rate of the model on attack data by constructing the intrusion detection model with the help of principal component analysis unsupervised dimensionality reduction algorithm and Adaboost algorithm [18]. Nabila et al. [19] constructed a classifier by Random Forest and were able to identify four types of attacks, DOS, Probe, U2R, and R2L [20]. GBDT [21] is one of the most applied models for integrated learning to solve classification problems. IDSs [22–26] based on GBDT [27] are one of the most widely used means to defend against attacker intrusions today. However, this scheme usually requires integrated learning of multiple base models, and suffers from three problems: unbalanced training data distribution, large feature dimensionality, and difficulty in finding the optimal model parameters, which reduce the recognition accuracy, learning efficiency, and generalization ability of the model.

To solve the above problems, we propose a lightweight gradient boosting method, called LGBM, to improve the recognition accuracy, training efficiency, and generalization ability of the model. The main contributions are as follows:

(1) For the problem of unbalanced training data distribution, we develop a Gradient Borderline-synthetic Minority Oversampling Technique (GSMOTE) for expanding small samples of data (data classes with small sample size). The algorithm first updates the data samples based on the gradient value of each sample in the dataset by the unilateral gradient sampling algorithm and then uses the synthetic minority oversampling algorithm to expand the updated dataset with small samples, thus ensuring the balance among the data samples.

(2) For the problem of large dimensionality of data features, we design an Exclusive Features Binding-Hierarchy Cross-Validation algorithm (EFB-HCV) to reduce the feature dimensionality of the data. The algorithm first performs feature combinations based on the graph coloring idea and binds the mutually exclusive features existing in the data set to reduce the number of features.

(3) For the problem of difficult parameter search during model training, we propose a Bayesian Optimization algorithm (BO) to improve the optimization efficiency of model parameters. The algorithm adds a step limit to the parameter search process and regulates the search range of the parameters according to the step size, which can effectively avoid the traversal operation of all parameters.

(4) To verify the effectiveness of LGBM, we compare LGBM, Random Forest, Adaboost, Decision Tree, and GBDT with the help of four metrics: precision, recall, F-measure, and Roc curve. Detailed experimental results show that the new scheme improves the recognition rate of the model for a few attack types, the efficiency of the model parameter search, the learning efficiency, and the generalization ability.

2. Basic Knowledge

This section introduces two aspects of gradient boosting Decision Tree and basic optimization solution.

2.1. Gradient Boosting Decision Tree. GBDT is an efficient regression problem-solving method based on the boosting algorithm, which uses the regression tree as the basic classifier and a gradient boosting learning algorithm to iteratively generate a Decision Tree. Boosting algorithm is a weighted linear combination of multiple weak learners, i.e., $f(\vec{x}) = f_M(\vec{x}) = \sum_{m=1}^{M} h_m(\vec{x}, \Theta_m)$, $\vec{x}$ is the input of model, $h_m(\vec{x}, \Theta_m)$ denotes the mth model, Θ_m are the parameters of the mth model, M is the number of base models. The CART tree applied by the boosting algorithm is a binary decision tree. The CART tree generates a classification decision tree, if the data to be predicted is discrete, and a regression decision tree if the data to be predicted is continuous. As an improved algorithm of GBDT, the LGBM also uses the CART regression tree as the base learner to find the best division point containing all features. The CART regression tree uses the squared error as the discriminant of the best division point, and the regression boosting tree process is described as follows:

For the training set $\mathbb{N} = \{(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \ldots, (\vec{x}_N, \vec{y}_N)\}$, the final output regression boosting tree model $f_M(\vec{x})$.

(1) Initialize regression lift tree $f_0(\vec{x}) = 0$;

(2) For the number of training sessions $m = 1, 2, \ldots, M$ there are:

(a) calculation of residuals $r_{mi} = -[\partial L(\vec{y}_i, f(\vec{x}_i))/\partial f(\vec{x}_i)]_{f(\vec{x})=f_{m-1}(\vec{x})}$;
(b) construction of training residual sets $R_m = \{(\vec{x}_1, r_{m,1}), (\vec{x}_2, r_{m,2}), \ldots, (\vec{x}_N, r_{m,N})\}$;
(c) the residuals were fitted by learning regression boosting tree r_m to obtain $h_m(\vec{x}, \Theta_m)$;
(d) update $f_m(\vec{x}) = f_{m-1}(\vec{x}) + h_m(\vec{x}, \Theta_m)$;

(3) Obtain the final regression lift tree model $f_M(\vec{x})$, $f_M(\vec{x}) = \sum_{m=1}^{M} h_m(\vec{x}, \Theta_m)$.

2.2. Basic Optimization Solution. GBDT needs to traverse all features of all samples when constructing a Decision Tree to obtain effective splitting nodes to obtain the maximum information gain points. However, when the number of samples is large and the dimensionality of sample features is too high, its training efficiency will be significantly reduced. The new algorithm LGBM uses one-sided gradient sampling algorithm and mutually exclusive feature binding algorithm to reduce the number of training samples and the number of sample features in the training process to improve the training speed of the model.

2.2.1. One-Sided Gradient Sampling. One-sided gradient sampling is a common processing algorithm when the dataset contains a large amount of sample data. Instead of using weight values to measure the importance of the samples in GBDT, the negative gradient of the loss function is fitted. The larger the sample prediction error, the larger the absolute value of the gradient, and the worse the learning of the sample. And, the smaller the sample prediction error, the smaller the absolute value of the gradient, and the better the learning of the sample.

The one-sided gradient sampling algorithm measures the importance of the sample by the gradient of the sample, i.e., the higher the absolute value of the gradient of the sample, the higher the importance of the sample. One-sided gradient sampling keeps all samples with larger gradient values, while random sampling is performed among samples with smaller gradient values. The specific process is that firstly, the samples are arranged in descending order according to their absolute values of gradient, and then the top a% of them are selected as the large gradient sample point set A, and the remaining sample set is randomly selected b% as the small gradient sample set B. Finally, the two sets are combined and the model is trained under the updated data set.

The variance gain in the one-sided gradient sampling algorithm is defined as:

$$\widetilde{V}_{j|T}(d) = \frac{1}{n_T}\left[\frac{\left(\sum_{\vec{x}_i\in\alpha_l} g_i + 1 - a/b\sum_{\vec{x}_i\in\beta_l} g_i\right)^2}{n_{l|T}(d)} + \frac{\left(\sum_{\vec{x}_i\in\alpha_r} g_i + 1 - a/b\sum_{\vec{x}_i\in\beta_r} g_i\right)^2}{n_{l|T}(d)}\right]. \tag{1}$$

n_T is the number of all retained samples, $n_{l|T}(d)$ is the number of samples in the left subtree, and $n_{r|T}(d)$ is the number of samples in the right subtree. α_l and α_r are the set of samples with larger retention gradient values in the left subtree and right subtree. β_l and β_r are the set of samples with smaller retention gradient values in the left subtree and right subtree. The algorithm defines the approximation error as: $\varphi(d) = |\widetilde{V}_{j|T}(d) - V_{j|T}(d)|$, the gradient values are $\overline{g}_l(d) = \sum_{\vec{x}_i\in L} g_i/n_{l|T}(d)$, $\overline{g}_r(d) = \sum_{\vec{x}_i\in R} g_i/n_{r|T}(d)$. The approximation error satisfies $\varphi(d) \le \mathbb{C}_{a,b}^2\lambda * \max\left\{1/n_{j|T}^j(d), 1/n_{r|T}^j(d)\right\} + 2\,D * \mathbb{C}_{a,b}\sqrt{\lambda/n}$. $\mathbb{C}_{a,b} = 1 - a/\sqrt{b}\max_{x_i\in B}|g_i|$ is the maximum gradient weighted value in B. $D = \max(\overline{g}_l(d), \overline{g}_r(d))$ is selected as the maximum of the mean gradient values in the left and right subtrees. The one-sided gradient sampling algorithm increases the diversity of the base model, which helps to improve the generalization ability of the integrated model and also improves the ability of the model to monitor the attack behavior.

2.2.2. Mutually Exclusive Feature Binding. For the sample $\vec{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})^T$ in the dataset $\mathbb{R} = \left\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\right\}$, if for each sample $i = 1, 2, \ldots, N$, there will be no $x_{i,j} \ne 0$, $x_{i,k} \ne 0$, then the sample features j and k are mutually exclusive features. The mutually exclusive feature finding process is described as follows (Algorithm 1):

Benefiting from the histogram algorithm, LGBM first groups consecutive features into n integers (n integers represent n histograms), then iterates through the sample features, merges the features belonging to a certain integer range into the histogram represented by that integer, and finally merges each feature into each histogram, thus finding the best segmentation point based on the discrete value of the histogram.

3. Lightweight Gradient Boosting Method

This section focuses on three aspects: unbalanced data processing, data characterization, and parameter optimization.

3.1. Unbalanced Data Processing. Through the analysis of the collected intrusion detection dataset, we found that the data distribution of each category in the original dataset is extremely unbalanced, and the number of DOS attack types in the KDD dataset is about 400,000, accounting for about 80% of the total data, while the number of U2L attack types is about 60, accounting for less than 1%. This problem is likely to cause the learner to overfit large samples (data categories with large sample data) and underfit small samples (data categories with small sample data), leading to a decrease in the accuracy of model recognition. To solve this problem, we propose the Gradient Borderline-synthetic Minority Oversampling Technique (GSMOTE) to expand the small sample data.

The GSMOTE algorithm manually synthesizes new sample data for the few classes of samples that are at the boundary, which effectively solves the problem that the boundary instances are prone to misclassification in the SMOTE. Using the gradient sampling algorithm decreases the amount of sample data and reduces the amount of learning required by the model. According to the fitting principle of GBDT to the objective function, the gradient is crucial information to measure the fitting effect of samples. Therefore, we update the dataset with the gradient value of samples. If the gradient value of a sample is small and its learning error is small, it indicates that the sample has been well trained and the instances with small gradient values can be appropriately deleted in the dataset. If the gradient value of a sample is large, it indicates that the sample has not been fully learned and the instances with large gradient values in the dataset are retained.

We first update the dataset using the one-sided gradient sampling algorithm to eliminate sample points with small gradient values, and then perform data balancing on the updated dataset using the GSMOTE Algorithm 2, the pseudo-code for the GSMOTE is as follows:

Where T is the number of minority samples, k is the number of nearest neighbors, and N is the sampling rate. *MinoritySam*[][] is an array for original minority samples, *Newindex*[][] is the number of synthetic samples generated, *SyntheticSam*[][] is the array for synthetic samples, and *numattrs* is the number of attributes.

3.2. Data Feature Dimensionality Reduction. The recursive feature elimination algorithm, which first assigns weights to each feature of the sample, is trained on the specified dataset using the base model. Then the feature weights of the trained model are extracted and the sample features with the smallest weight are removed by sorting them from largest to smallest according to their absolute values. Finally, the process is recursively repeated until the desired number of features is reached. In the sparse feature space, many features are mutually exclusive, i.e., they never obtain nonzero values at the same moment, and mutually exclusive features can be bundled into one feature. The algorithm reduces the data feature dimensionality to some extent and retains the valid feature information by cross-validation methods. However, this algorithm is costly in the process of dimensionality reduction and requires iterative training of the base model to traverse all sample features before the final sample dataset can be obtained. In addition, the sample features eliminated by recursion also contain information useful for the classification samples, and the dataset after multiple recursions will lose some effective features. Therefore, the performance of the model trained with this dataset is reduced.

The number of samples in the original dataset is large, and the feature dimension is large. According to the analysis of the original dataset, it is known that the high-dimensional data feature space has multiple features whose values will not be nonzero at the same time, i.e., the high-dimensional data feature space is sparse, which we call mutually exclusive features. Therefore, the mutually exclusive features can be used to merge multiple features in the dataset, thus reducing the number of features and the dimensionality of the features. According to the idea of the graph coloring problem, the mutually exclusive feature merging algorithm uses graph vertices to represent sample features, and there is no connection between mutually exclusive features, so when the graph is colored with K colors, there are K groups of mutually exclusive features in the graph. The pseudo-code of the Exclusive Features Binding Algorithm 3 is as follows:

The mutually exclusive feature merging algorithm performs feature combination based on the graph coloring problem idea, where features are used as vertices of a graph, edges connect two nonmutually exclusive features, and the weights of the edges indicate the total conflict values of the two features, and feature points of the same color in the graph are mutually exclusive features. For incomplete mutually exclusive features, the algorithm allows lower conflicts, so the features can be further combined to reduce the number of features and improve computational efficiency. To ensure that the original features are successfully separated after each feature combination is merged, i.e., the original feature values can be identified in the merged feature combination, the algorithm sets offsets for the corresponding feature values, appropriately changes the range of feature values, and assigns different feature values to different bins in the feature combination, thus avoiding feature value confusion after feature fusion.

To avoid the uneven distribution of data categories by the K-fold cross-validation method, we propose to use the Hierarchy Cross-Validation Algorithm (HCV), which treats the sample data of each attack category in a balanced way and uses a hierarchical data extraction method to ensure the equal proportional division of attack categories in the training and test sets. The pseudo-code of the Exclusive Features Binding-Hierarchy Cross-Validation Algorithm 4 (EFB-HCV) is as follows:

The EFB-HCV algorithm first optimizes the data feature reduction scheme of RFE-HCV to avoid recursively manipulating the dataset, and then uses the same hierarchical cross-validation method in data slicing to ensure equal proportional distribution of attack categories in the training and test sets. The specific details are described as follows:

(1) In terms of feature optimization, the EFB-HCV algorithm uses the mutually exclusive feature binding technique to feature the dataset and merges the same color feature points, i.e., mutually exclusive features, and sets an offset for incomplete mutually exclusive feature values to further reduce the number of features. The algorithm does not need to assign a weight value to each feature, which avoids the iterative training of the model and reduces the model training cost.

(2) In terms of data assignment, the data features are processed by the mutually exclusive feature binding algorithm. Then the hierarchical cross-validation algorithm divides the data proportionally. In other words, the data in each training set belong to different attack categories, and the proportion of attack categories in each training set and test set is the same as the original training set.

3.3. Parameter Optimization. The grid search algorithm first combines the values of multiple parameters and grids them, then uses each set of parameter combinations for base model training, and finally selects the best parameter combination based on the performance of the model. The updated grid search algorithm improves the efficiency of parameter search to some extent by stepping updating strategy, but both algorithms simply search for parameter combinations without making full use of the information of search points, which reduces the quality and efficiency of parameter search. Bayesian Optimization Algorithm (BO) in the case of a large number of parameter combinations can be more efficient than grid search by establishing the proxy function through finite iteration, making full use of the searched parameter information, and determining the optimal parameter combination directly based on the maximum value of the proxy function. The pseudo-code of BO is as follows (Algorithm 5):

The base model root node is initialized, the constant values are predicted, and the parameters such as model n_estimator are estimated. Where *f* is the black box function being optimized, *X* is the parameter search space, *S* is the collection function, and *M* is the agent model. According to the Bayesian optimization idea, firstly initialize the data set *Data* which contains *n* candidate solutions. Second, the *n* candidate solutions found by this point set are used to build a Gaussian regression model for making the posterior probabilities of other candidate points. Then, the collection function is constructed based on the posterior probabilities to find the next point that may produce the extreme value. Finally, the point that makes the function reach its maximum value is selected as the parameter of the training model.

Bayesian modeling of the function values of the black-box function using a Gaussian process gives the probability distribution of each function value, lets the function value at each point be a random variable, and multiple random variables form a random vector obeying a normal distribution. For the function *f*(*x*), there are *n* sampling points $(x_1, x_2, \cdots, x_n)$, the corresponding function values $f(x) = [f(x_1), f(x_2), \ldots f(x_n)]$ of which form a vector, which obey a normal distribution in the Gaussian regression process:

$$f(x) \sim N\left(\mu(x_{1:n}), \sum(x_{1:n}, x_{1:n})\right). \tag{2}$$

$\mu(x_{1:n})$ is the mean vector of the Gaussian distribution. $\sum(x_{1:n}, x_{1:n})$ denotes the covariance matrix. The covariance matrix is usually implemented using a kernel function, which is defined in the Gaussian regression process as:

$$k(x_1, x_2) = \partial \exp\left(-\frac{1}{2\varepsilon^2}\|x_1 - x_2\|^2\right). \tag{3}$$

∂, ε is the parameter of the kernel function, and the mean vector is calculated from the mean function $\mu(x)$. According to the multidimensional normal distribution from the covariance matrix and the mean the vector,we can predict the probability distribution of the function value of the pointx_{n+1}, after adding the point the function value vector distribution is $f(x_{1:n+1})$ and obeys the $n+1$ dimensional normal distribution.

$$\begin{bmatrix} f(x_{1:n}) \\ f(x_{n+1}) \end{bmatrix} \sim N\left(\begin{bmatrix} \mu(x_{1:n}) \\ \mu(x_{n+1}) \end{bmatrix}, \begin{bmatrix} K & k \\ k^T & k(x_{n+1}, x_{n+1}) \end{bmatrix}\right), \tag{4}$$

where $f(x_{1:n})$ obeys the *n*-dimensional normal distribution, the mean vector is $\mu(x_{1:n})$, k is denoted as $k = [k(x_{n+1}, x_1), \quad k(x_{n+1}, x_2), \ldots k(x_{n+1}, x_n)]$, calculated from the kernel function. The covariance matrix *K* can be calculated based on the mean function and the covariance function. The mean and variance expressions of the conditional distribution obeyed can be introduced according to the rules for calculating the $f(x_{n+1})$ multidimensional normal distribution as:

$$\begin{aligned} \mu &= k^T K^{-1}(f(x_{1:n}) - \mu(x_{1:n})), \\ \sigma^2 &= k(x_{n+1}, x_{n+1}) - k^T K^{-1} k. \end{aligned} \tag{5}$$

Suppose the mapping relationship between the parameters to the objective function is *f*(*x*), *f*(*x*) is uncertain, and the acquisition function constructed by the mathematical expectation of *f*(*x*) does not satisfy the conditions of the corresponding function, i.e., the value of the acquisition function is small at the existing adopted points, and the value of the acquisition function is large at the points within the confidence interval and the mean value of the function is larger.We improve theexpectation acquisition function as follows: let *n* candidate solutions have been searched and the function is maximal:

$$\widehat{f}_n = \max(f(x_1), f(x_2), \ldots, f(x_n)). \tag{6}$$

Calculate the function value for the next candidate point x_{n+1} as $f(x_{n+1})$, if $f(x_{n+1}) \geq \widehat{f}_n$, then the extreme value of the function at *n*+1 is $f(x_{n+1})$, and vice versa $\widehat{f}_n$. After adding new candidate points, the improvement value of the function is $[f(x_{n+1}) - \widehat{f}_n]^*$, and the optimization goal is to find the candidate point *x* that makes the maximum improvement value.

$$EI_n(x) = E_n\left[\left[f(x_{n+1}) - \widehat{f}_n\right]^*\right]. \tag{7}$$

$E_n[*] = E[* \,|x_{1,n}, y_{1,n}]$ denotes the expected value calculated from the first *n* sampling points and their function values. Because the Gaussian process *f*(*x*) obeys a normal distribution, located at point *x* the mean value is $\varphi = \varphi(x)$, and the variance is $\sigma^2 = \sigma^2(x)$, such that $\lambda = f(x)$, is introduced:

$$\begin{aligned} EI_n(x) &= \int_{-\infty}^{+\infty} \left[\lambda - \widehat{f}_n\right]^+ \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\lambda - \varphi)^2}{2\sigma^2}\right) dz \\ &= \int_{\widehat{f}_n}^{+\infty} \left(\lambda - \widehat{f}_n\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\lambda - \varphi)^2}{2\sigma^2}\right) dz. \end{aligned} \tag{8}$$

Based on the points for dollars we get,

$$EI_n(x) = \int_{\hat{f}_n}^{+\infty} \left(\lambda - \hat{f}_n\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\lambda-\varphi)^2}{2\sigma^2}\right) dz$$
$$= \left(\varphi - \hat{f}_n\right)\left(1 - \vartheta\left(\frac{\left(\hat{f}_n - \varphi\right)}{\sigma}\right)\right) + \sigma\tau\left(\frac{\left(\hat{f}_n - \varphi\right)}{\sigma}\right). \quad (9)$$

$\tau(x)$ is the probability density function of the standard normal distribution. $\vartheta(x)$ is the distribution function of the normal distribution. Let $\Delta(x) = \varphi(x) - \hat{f}_n$ then,

$$EI_n(x) = [\Delta(x)]^+ + \sigma(x)\tau\left(\frac{\Delta(x)}{\sigma(x)}\right) - |\Delta(x)|\vartheta\left(\frac{\Delta(x)}{\sigma(x)}\right). \quad (10)$$

The expectation improvement function defines the expected value at each point as a function of that point, and eventually, the next candidate point is obtained based on the extreme value of the expected improvement function:

$$x_{n+1} = \arg\max EI_n(x). \quad (11)$$

4. Experimental Section

We validate the model optimization scheme for LGBM by using the Anaconda integrated development tool. First, we introduce the NSL-KDD. Secondly, we compare LGBM with GBDT, Adaboost, Decision Tree, and Random Forest under three metrics of precision, recall, F-measure, and Roc curve to elaborate advantages of LGBM. Then we analyze the effectiveness of LGBM for identifying different attack types using Roc curves. Finally, the optimization process of Bayesian and grid search for hyperparameters is compared and analyzed to verify that the Bayesian optimization algorithm has a better optimization effect on hyperparameters while ensuring optimization efficiency.

4.1. NSL-KDD Dataset. To address the problems of redundant records and unbalanced attack categories in the KDD, the NSL-KDD removes duplicate records from the training and test sets to ensure that the classifier does not bias towards a larger number of attack types, which in turn improves the detection accuracy of the classifier. Setting the number of records in the training and test sets can reduce the running cost of the experiment and eliminate the need to randomly select some data. As shown in Table 1, NSL-KDD contains four attack types (Dos, Probe, U2R, and R2L) and 21 specific attack instances, which are more abundant compared with the KDD, and the test set contains new samples of attack instances to better evaluate the classification performance of the learner. The distribution of the NSL-KDD data set is shown in Table 2.

5. Comparative Analysis of Model Recognition Performance

To test the recognition performance of LBGM to identify different classes of attacks, we compare the LGBM model with four models, GBDT, Adaboost, Decision Tree, and Random Forest under the three metrics of precision, recall, and F-measure. GBDT (Gradient Boosting Decision Tree) [27] is an iterative decision tree algorithm, which constructs a set of weak learners and accumulates the results of multiple decision trees as the final prediction output. Random forest [28] is a commonly used machine learning algorithm, which combines the output of multiple decision trees to reach a single result. The five models are trained and learned under the NSL-KDD training set, and the models are validated by the test set to derive the recognition performance of each model for the attack types under different metrics.

Figure 1(a) indicates the precision of the five models for the attack instances. From this table, it can be seen that the LGBM has a high precision for each attack type and most of the models have good precision for two common attack types, Probe and DOS. LGBM has the highest accuracy of precision compared to the other two options, which is only slightly weaker in Probe. The number of weak classifiers in the Adaboost is hard to set, resulting in a lower precision for the Probe attack type. For the two minority attack types U2R and R2L, the LGBM and the GBDT have high precision for both attacks, which is because both models use the boosting mechanism to integrate multiple weak learners to obtain strong learners, thus optimizing the overall performance of the model. Also, the LGBM uses the gradient synthesis minority class over adoption algorithm to process the data set and reasonably increases the number of samples of both U2R and R2L attacks, making the model equally good at identifying minority attack types. Decision Tree and Random Forest do not sample the dataset, resulting in a lower precision for U2R attack samples than GBDT and LGBM, but the Decision Tree is more sensitive to anomalous samples, so the Decision Tree has better precision for R2L.

Figure 1(b) presents the recall of the five models for different attack types. Compared to the precision rate, the recall of all models for U2R and R2L attack samples decrease to some extent, but the LGBM model still maintains relatively high recall for both attack samples, indicating that the model throws to maintain a good recall of positive example samples while ensuring the precision rate. For common attack types such as Probe and Dos, the number of sample features is large and the model can maintain a stable recall through training, so most of the models still present high recall for common attack types.

Figure 1(c) shows the F-measure of the five models for different attack types. A comparison of the F-measure of the five models for different attack types shows that the overall performance of the Adaboost model is poor, which is because that the model fails to effectively process the data samples and cannot effectively identify unknown or uncommon attack types. Compared with Adaboost, the LGBM has better overall performance, especially for the detection and identification of two rare attack types, U2R and R2L. That is because the sample processing of the dataset removes the sample data with small gradient values and expands the small samples (a small number of attack samples) so that the model can fully learn from them. Random Forest outperforms Adaboost in overall performance but is less effective in

```
        Input: $\mathbb{R} = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$, the conflict threshold $K$;
        Output: the set $\mathbf{F}$ of feature grouping $bin$;
Step 1: Initialize $FN$ as an array consisting of the number of nonzero eigenvalues;
Step 2: Iterate over all features $j$ of all samples and obtain the nonzero value $FN_j$ of $j$, sort the number of vertex features in descending
        order according to the array $FN$, and initialize the set $\mathbf{F}$;
Step 3: Assume that the current vertex is $j$, traverse the feature grouping $bin$, calculate the conflict value $con$ between $j$ and the feature
        points in the $bin$, $con$ is less than $K$, then it indicates that vertex $j$ and the feature points in the $bin$ do not conflict, add $j$ to the
        feature grouping $bin$;
Step 4: If vertex $j$ conflicts with the features in $bin$ and is not added to the feature grouping $bin$, create a new feature grouping for that
        vertex and add it to the set $\mathbf{F}$.
```

Algorithm 1: Mutually exclusive feature search algorithm.

```
(01) Initialize $T$, $k$, $N$;
(02) If $N < 100$ then
(03)     Randomize the $T$ minority class samples;
(04)     $T = (N/100) * T$;
(05)     $N = 100$;
(06) End if
(07) For $i = 1$ to $T$:
(08)     Compute $k$ nearest neighbors for $i$ and save the indices in the $mArray$;
(09)     while $N! = 0$ do
(10)         Choose a random number $s$ between 1 and $k$;
(11)         for $a = 1$ to $numattrs$:
(12)             $dif = MinoritySam[mArray[s]][a] - MinoritySam[i][a]$;
(13)             $gap$ = random a number between 0 and 1;
(14)         end for
(15)         $Newindex{+}{+}$;
(16)         $N = N - 1$;
(17)     End while
(18) End for
```

Algorithm 2: GSMOTE algorithm.

identifying and detecting uncommon attack types than LGBM models. That is because the trained models are slightly less targeted due to the lack of expansion of the small sample data. To sum up, the overall performance of the LGBM model is better than the other models.

5.1. Comparative Analysis of Model Roc Curves. Figure 2 depicts the Roc curves of the five models for different attack types. According to the definition of Roc curves, the special points in the figure are first analyzed. The point (0, 1), i.e., TPR = 1 and FPR = 0, indicates that the classifier classifies all samples correctly. The point (1, 0), i.e., TPR = 0 and FPR = 1, indicates that the classifier misclassifies all samples and has the worst performance. The two points (0, 0) (1, 1) indicate that the classifier predicts negative samples and positive samples. The ability of the model to detect and identify each attack type is known from the Roc curve of each model.

Figures 2(a) and 2(c) show that the Roc curves of Decision Tree and Random Forest have similar recognition effects on attack samples such as U2R and R2L. Also, the curves are close to $y = x$ indicating that the model classifies the samples randomly and does not effectively detect the sample data. That is because the Decision Tree and Random Forest models tend to select different attributes when classifying the category data with a large difference in the number of samples, resulting in a poor recognition rate due to insufficient training for sample data with few attributes. It can conclude that the two models have a higher recognition accuracy for a larger number of Probe, DOS, and normal data samples. Figure 2(d) shows the Roc curves of the Adaboost model for different attack types. The model appears to misclassify R2L samples and has lower recognition accuracy for other types of sample data, as the model is sensitive to the distribution of sample data and does not balance the dataset leading to a decrease in the classification accuracy of the model. The Roc curves of the GBDT and LGBM models are shown in Figures 2(b) and 2(e), and it can be seen that both of them have better recognition ability for each type of data sample. Since the GBDT model can effectively deal with anomalous data and can handle both continuous and discrete values, the model has better recognition accuracy for R2L attack types. In summary, the overall recognition accuracy of the LGBM model for different attack types is better than other models.

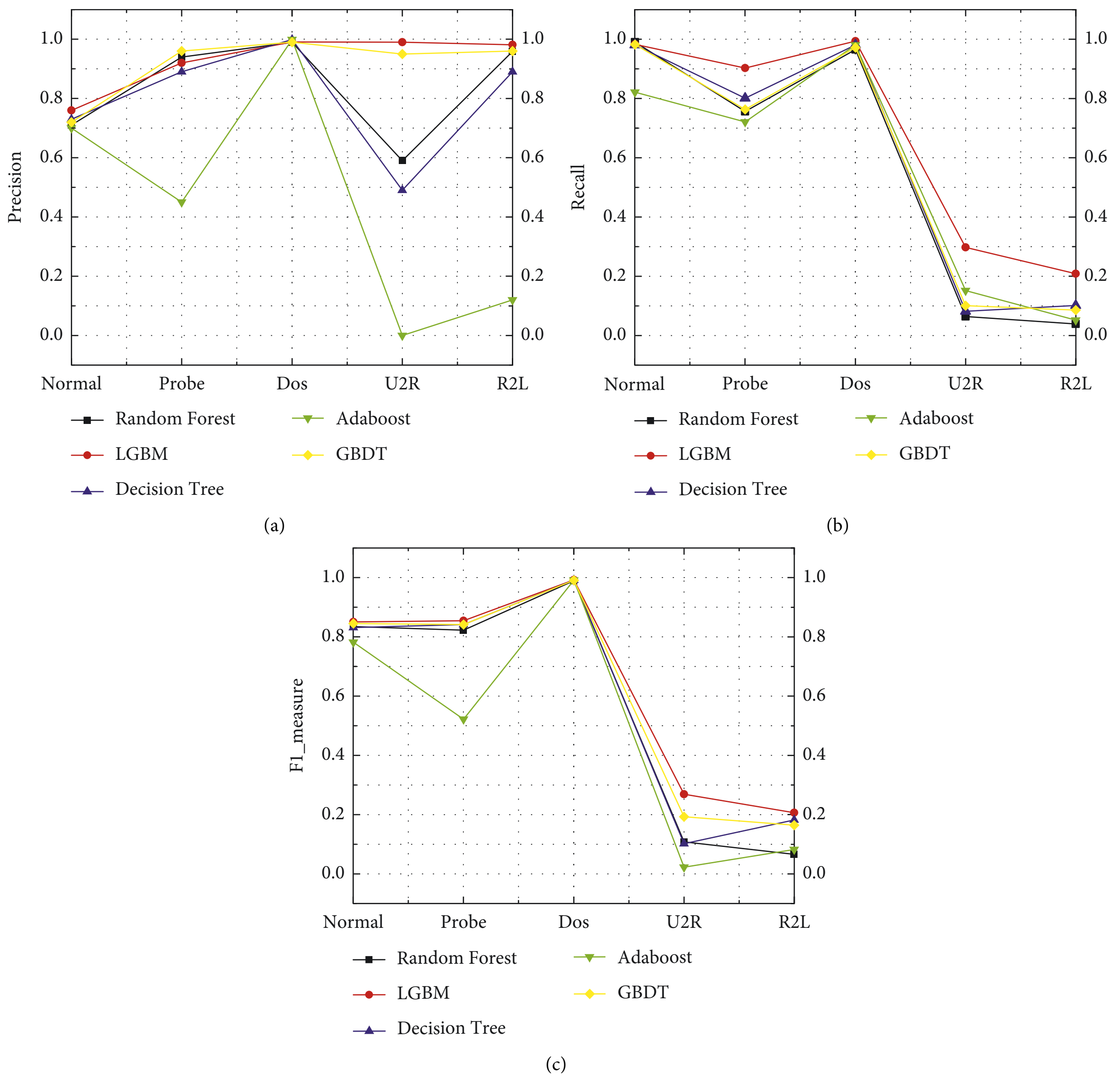

Figure 1: Accuracy of the analysis of model recognition performance. (a) Precision. (b) Recall. (c) F-measure.

5.2. Comparative Analysis of Model Parameter Optimization. Hyperparameters are set before the training of a machine learning model and directly affect the learning effect of the model. A set of optimal hyperparameters can improve the learning ability and effectiveness of the model. The grid search algorithm can iterate through all parameter combinations to find the optimal combination of parameters, but this method is less efficient. If the number of model parameters is too large, the grid search algorithm will increase the training cost of the model and reduce the efficiency of parameter optimization. Moreover, the algorithm performs an iterative search so that the model training is not targeted. Therefore, we use the Bayesian optimization algorithm to optimize the parameters, and the experimental results are shown in Tables 3 and 4. We select three important parameters, max_depth, n_estimator, and num_leaves, to measure the amount of data contained in the NSL-KDD and train them under the LGBM model with tuning parameters.

Tables 3 and 4 show the overall performance of optimization methods with different combinations of parameters. The model achieves the highest performance value of 0.9528 in the test set, the grid search algorithm, while the performance value of the model optimized with Bayesian parameters is 0.9906, which indicates that the Bayesian-optimized parameters can enhance the training learning ability of the model and obtain a stronger classifier. This is because the random combination of different parameter values by the grid search algorithm tends to lead to excessive differences between different parameter values, making it difficult for the model to reach the optimal value. The Bayesian optimization algorithm establishes a functional relationship between the hyperparameters and the model

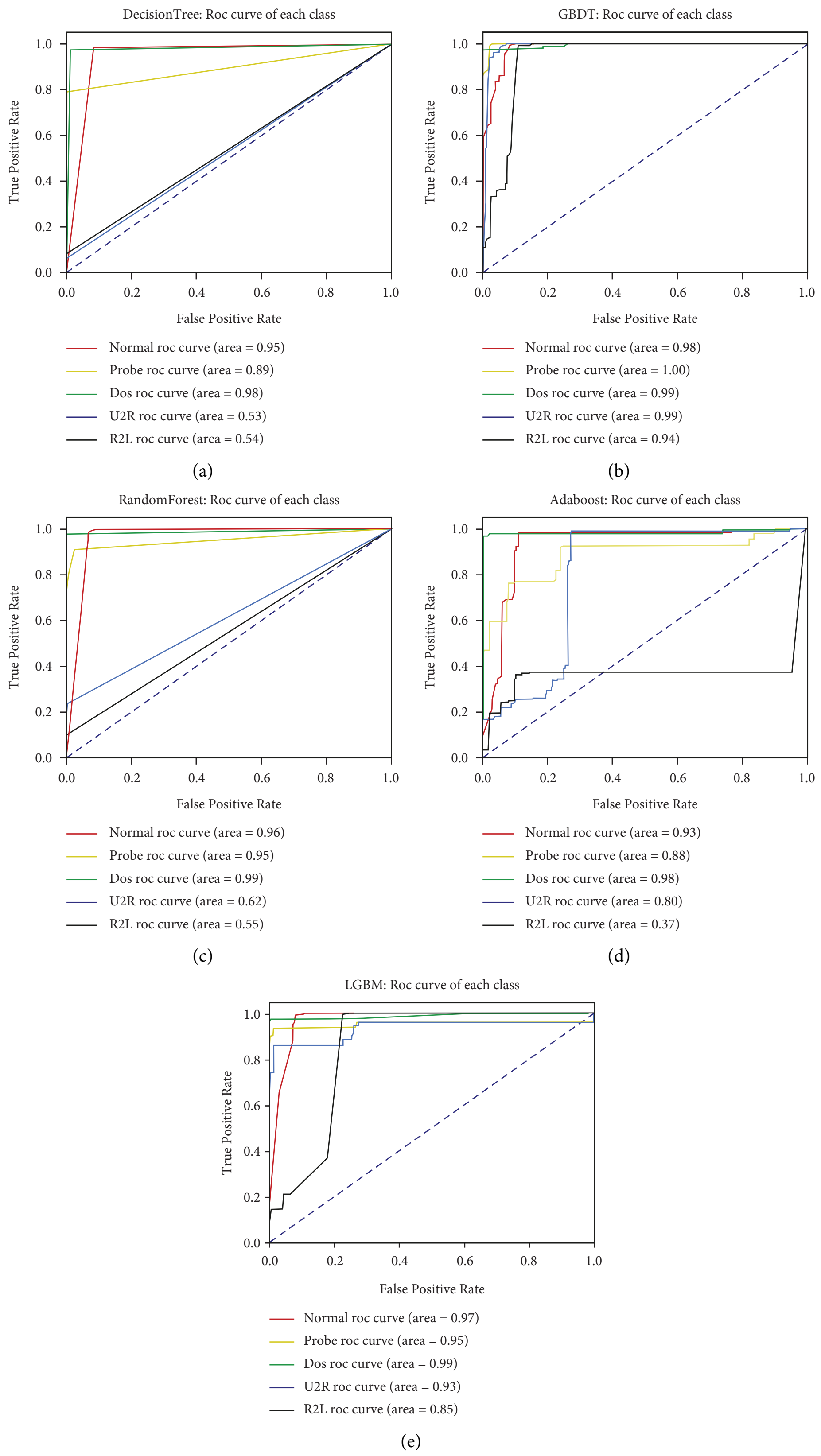

Figure 2: Roc curve of model. (a) Decision tree. (b) GBDT. (c) Random forest. (d) Adaboost. (e) LGBM.

```
(01) **Input**: $F$: features, $MC$: max conflict count, $G$: construct graph, $NumData$: number of data, $B$: One bundle of exclusive features;
(02) searchOrder = $G$.sortbyDegree();
(03) for $i$ in searchOrder do:
(04)   needNew = True;
(05)   for $j = 1$ to len ($bundles$) do:
(06)     cnt = ConflictCnt ($bundles$[j], $F$[i]);
(07)     if cnt + $bundlesConflict$ [i] $\le MC$ then
(08)       $bundles$ [j].add (F[i]), needNew = False;
(09)       break;
(10)       $binScope = 0$, $NumBin = 0$;
(11)     end if
(12)   end for
(13) end for
(14) for $i = 1$ to $NumData$ do
(15)       newBin [$i$] = 0;
(16)       for $j = 1$ to len ($B$) do
(17)         if $B[j]$.bin$[i]! = 0$ then
(18)           newBin [$i$] = $B[j]$.bin[$i$] + $binScope$ [$j$];
(19)       end if
(20)     end for
(21)   end for
(22) Output: newBin, $binScope$
```

Algorithm 3: Exclusive features binding.

```
(01) Initialize Estimator, $\sum_k (\gamma_k)$, $M_1, \cdots, M_g$;
(02) for $m = M_1$ to $M_g$ do:
(03)   exclusive features binding for $M_i, i = 1, \cdots, g$;
(04)   covariance matrix $Sc$ obtained from sample set;
(05)   fitting $F(S_c, \sum_k (\gamma_k)) \longrightarrow \hat{\gamma}_{k,c}, \sum_k (\hat{\gamma}_{k,c})$;
(06)   covariance matrix $S_V$ obtained from Validation set;
(07)   fitting $F(S_v, \sum_k (\hat{\gamma}_{k,c})) \longrightarrow \Theta$;
(08)   comparison of the cross-validation indices $\Theta$;
(09) end for
(10) Output the most stable $\Theta$.
```

Algorithm 4: EFB-HCV algorithm.

```
(01) Initialize $f_0(x) = \text{argmin}_\gamma \sum_{i=1}^N L(y_i, \gamma)$, $f$, $X$, $S$, $M$;
(02) Initialize EFB-HierarchyCV;
(03) n_estimator: Data = Samples($f, X$);
(04) for $i = av$(Data) to $T$ do:
(05)   $P(y|x, \text{Data}) = \text{Fit Model}(M, \text{Data})$;
(06)   $X_i = \arg\max S(x, P(y|x, \text{Data}))$;
(07)   $Y_i = f(X_i)$;
(08)   Data = Data + $(X_i, Y_i)$;
(09)   for $m = 1$ to $M$ do:
(10)     for $i = 1, 2, \cdots, N$ do:
(11)       compute $r_{im} = -[\partial L(y_i, f(x_i))/\partial f(x_i)]_{f=f_{m-1}}$;
(12)     end for
(13)   end for
(14)   fit a regression tree to the targets $r_{im}$ giving terminal regions $R_{jm}, j = 1, 2, \cdots, J$;
(15)   for $j = 1, 2, \cdots, J_m$ do:
(16)     compute $r_{jm} = \arg\min \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$;
(17)   end for
(18)   Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$;
(19) end for
(20) Output: $\hat{f}(x) = f_M(x)$.
```

Algorithm 5: BO algorithm.

Table 1: Types of attacks on the NSL-KDD dataset.

Type of attack	Specific attack examples
DOS	apache2; back; mailbomb; neptune; pod; land; processtable; smurf; teardrop; udpstorm
Probe	ipsweep; mscan; portsweep; saint; satan
U2R	buffer_overflow; loadmodule; perl; ps rootkit; sqlattack; xterm; httptunnel
R2L	imap; multihop; named; phf; sendmail snmpgetattack; snmpguess; worm; xlock xsnoop; spy; warezclient; warezmaster

Table 2: The distribution of the NSL-KDD dataset.

Type	Number of records
Normal	67343 (53%)
DOS	45927 (37%)
Probe	11656 (9.16%)
U2R	52 (0.04%)
R2L	995 (0.8%)

Table 3: Parameter optimization analysis: grid search optimization.

Grid search optimization			
Target	max_depth	n_estimator	num_leaves
0.9139	8.557	188.0	59.78
0.9427	6.157	140.6	59.51
0.8709	7.764	225.1	56.0
0.9188	9.0	68.68	40.0
0.9528	6.0	10.0	40.0

Table 4: Parameter optimization analysis: Bayesian optimization.

Bayesian optimization			
Target	max_depth	n_estimator	num_leaves
0.9218	8.705	114.8	40.3
0.9432	6.084	111.0	59.57
0.9906	8.705	22.86	40.03
0.9831	8.963	12.35	40.39
0.9658	6.0	250.0	40.0

objective function, and the corresponding parameter values are obtained through the optimal value of the functional relationship.

6. Conclusion

Intrusion detection technology is one of the most well-known security protection technologies in the traditional Internet domain. However, due to the emerging resource-constrained network entities, with limited computing power or insufficient power supply, it is difficult for mainstream intrusion detection technologies to perform as effectively as before. IDSs based on GBDT face three major challenges: unbalanced training data distribution, excessive feature dimensionality, and difficulty in finding the best model parameters that cannot be effectively applied to the security protection of end devices in IoT. To solve these problems, we propose an optimization model LGBM for GBDT. Detailed experimental results verify the effectiveness of the proposed scheme.

Data Availability

All data generated or analyzed during this study are included in this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper. This research is supported by the National Natural Science Foundation of China (NSFC) [grant nos. 62172377, 61872205], and the Shandong Provincial Natural Science Foundation [grant no. ZR2019MF018].

References

[1] S. Wijethilaka and M. Liyanage, “Survey on network slicing for Internet pf Things Realization in 5G networks,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[2] Z. Cai and X. Zheng, “A Private and efficient mechanism for data Uploading in Smart Cyber-Physical systems,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[3] Z. Cai, X. Zheng, J. Wang, and Z. He, “Private data trading towards range counting Queries in Internet of Things,” *IEEE Transactions on Mobile Computing*, p. 1, 2022.

[4] P. K. Malik, R. Sharma, R. Singh et al., “Industrial Internet of Things and its Applications in Industry 4.0: State of the Art,” *Computer Communications*, vol. 166, pp. 125–139, 2021.

[5] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, “Network intrusion detection system: a Systematic study of machine learning and Deep learning Approaches,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, pp. 1–29, 2021.

[6] Z. Cai, Z. He, X. Guan, and Y. Li, “Collective data-Sanitization for Preventing sensitive information Inference attacks in Social networks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 1–590, 2016.

[7] Z. Yang, X. Liu, T. Li et al., "A Systematic Literature Review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, pp. 102675–102720, 2022.

[8] T. M. Koo, H. C. Chang, Y. T. Hsu, and H. Y. Lin, "Malicious Website detection based on Honeypot systems," in *Proceedings of the 2nd International Conference on Advances in Computer Science and Engineering*, pp. 76–82, Atlantis Press, July 2013.

[9] J. Gu and S. Lu, "An effective intrusion detection Approach using SVM with Naïve Bayes feature Embedding," *Computers & Security*, vol. 103, no. 3, pp. 102158–102219, 2021.

[10] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for Cyber security intrusion detection: Datasets and Comparative study," *Computer Networks*, vol. 188, pp. 107840–107916, 2021.

[11] G. Andresini, A. Appice, and D. Malerba, "Autoencoder-based Deep metric learning for network intrusion detection," *Information Sciences*, vol. 569, pp. 706–727, 2021.

[12] J. Liu, Y. Gao, and F. Hu, "A Fast network intrusion detection system using Adaptive synthetic oversampling and Lightgbm," *Computers & Security*, vol. 106, pp. 102289–102316, 2021.

[13] O. Alkadi, N. Moustafa, B. Turnbull, and K. K. R. Choo, "A Deep Blockchain Framework-Enabled Collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2021.

[14] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: fully Convolutional neural network for Fast anomaly detection in Crowded Scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88–97, 2018.

[15] N. Moustaf and J. Slay, "Creating Novel features to anomaly network detection using DARPA-2009 data set," in *Proceedings of the 14th European Conference on Cyber Warfare and Security*, pp. 204–212, Academic Conferences Limited, July 2015.

[16] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using A Filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[17] N. Moustafa, B. Turnbull, and K. K. R. Choo, "An Ensemble intrusion detection technique based on proposed Statistical Flow features for protecting network Traffic of Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2019.

[18] Y. Yuan, G. Kaklamanos, and D. Hogrefe, "A Novel Semi-Supervised Adaboost technique for network anomaly detection," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 111–114, ACM, November 2016.

[19] C. A. Ronao and S. B. Cho, "Anomalous Query access detection in RBAC-Administered Databases with random forest and PCA," *Information Sciences*, vol. 369, pp. 238–250, 2016.

[20] K. Liu, X. Hu, H. Zhou, L. Tong, W. D. Widanage, and J. Marco, "Feature Analyses and modeling of Lithium-Ion Battery Manufacturing based on random forest classification," *IEEE-ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 2944–2955, 2021.

[21] V. Sharma and R. N. Mir, "An enhanced time efficient technique for image Watermarking using Ant Colony optimization and Light gradient boosting algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 615–626, 2022.

[22] J. Chen, X. Zhang, T. Wang et al., "Fidas: Fortifying the cloud via Comprehensive FPGA-based Offloading for intrusion detection: industrial Product," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ACM, June 2022.

[23] Q. M. Alzubi, M. Anbar, Y. Sanjalawe, M. A. Al-Betar, and R. Abdullah, "Intrusion detection system based on Hybridizing A Modified binary Grey Wolf optimization and Particle Swarm optimization," *Expert Systems with Applications*, vol. 204, Article ID 117597, 2022.

[24] A. Telikani, J. Yang, and P. Wang, "Industrial IoT intrusion detection via Evolutionary cost-sensitive learning and Fog computing," *IEEE Internet of Things Journal*, p. 1, 2022.

[25] D. Chou and M. Jiang, "A Survey on data-Driven network intrusion detection," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–36, 2022.

[26] A. Telikani, J. Shen, J. Yang, and P. Wang, "Deep learning for intrusion detection and security of Internet of Things (IoT): current analysis, challenges, and Possible solutions," *Security and Communication Networks*, vol. 2022, pp. 1–13, 2022.

[27] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu, "DeepGBM: a Deep learning Framework Distilled by GBDT for online prediction Tasks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 384–394, ACM, July 2019.

[28] A. Paul, D. P. Mukherjee, P. Das, A. Gangopadhyay, A. R. Chintha, and S. Kundu, "Improved random forest for classification," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4012–4024, 2018.