Research Article

Research on the Intrusion Detection Model of Underwater Sensor Networks

Haijie Huang, Na Liu, Dandan Chen, Qiuling Yang, and Xiangdang Huang

School of Computer Science and Technology, Hainan University, Haikou 570000, China

Correspondence should be addressed to Xiangdang Huang; 990623@hainanu.edu.cn

Received 28 September 2021; Accepted 8 March 2022; Published 29 May 2022

Academic Editor: Yuan Li

Copyright © 2022 Haijie Huang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The characteristics of underwater nodes, the harshness of the underwater environment, and the openness of the deployment area make underwater sensor networks vulnerable to various network attacks; in order to defend against potential threats and attacks, intrusion detection, as an active defense technology, can detect different attacks before they occur; currently, the technology is less used in underwater sensor networks and has a low detection rate for some types of attacks and cannot effectively detect the problem of multiple attack types. To address this problem, this paper proposes an intrusion detection model for underwater sensor networks for multiple types of attacks. Firstly, cluster head nodes use neighborhood rough sets for feature extraction, and the reduced dimensional data is transmitted to sink nodes to reduce the node computation. Further, the synthetic minority oversampling technique (SMOTE) is used to balance the data set, increase the number of minority class samples, and improve the detection rate of minority class attacks. Finally, determine whether a node is trusted according to the trust value of the cluster head node, and train the classifier using the random forest algorithm to detect the type of attack; it suffers to achieve intrusion detection of multiple types of attacks. Simulation results show that the model can not only improve the performance of intrusion detection of multitype attacks but also achieve an accuracy of over 99% for the detection of imbalance classes.

1. Introduction

Underwater wireless sensor networks (UWSNs) are an extension of land-based wireless sensor networks to the underwater or ocean environment, in which underwater sensor nodes are manually deployed to a defined region and can float on the surface, sink into the water, or be attached to the seafloor [1]. As seen in Figure 1, they sense the surrounding surroundings and gather data and then transmit the gained data through wireless acoustic communication to the water surface convergence node via a single hop or several hops to reach the user. Underwater sensor networks have the potential to improve the technological equipment and information platforms available for marine environmental monitoring, marine resource protection, marine catastrophe monitoring, marine exploration, marine production operations, and maritime military actions.

Underwater sensor nodes face several resource restrictions, including limited energy, processing capacity, and communication range, which expose the network to a variety of network assaults [2]. Because of the harsh and unpredictable underwater environment, as well as the openness and dynamic nature of the deployment area, it is difficult to physically resist potential attackers; these shortcomings make it easier for attackers to intervene in the network and launch various network attacks. As a result, the security of underwater sensor networks has become a priority.

Because of the wide application of underwater sensor networks, different application scenarios often have different security requirements. In areas with low security requirements such as marine environment monitoring, marine exploration, etc. only the integrity and freshness of the data is required, but for areas with high security requirements such as target monitoring, tracking, etc. security techniques such as authentication, encryption techniques, intrusion detection, etc. are required [3]. As in the marine environmental monitoring, marine exploration, and other areas of the security requirements of low only need to ensure
The remainder of this essay is structured as follows. Section 2 is devoted to related work, Section 3 to the model’s development, and Section 4 to the experimental findings and analysis. Finally, Section 5 provides a synopsis of the article.

The abbreviations in this paper are expressed as shown in Table 1.
2. Related Work

Recently, many scholars have applied different machine learning methods to intrusion detection, such as support vector machine, deep neural network, random forest, and artificial neural network.

Wang et al. [8] developed an ensemble learning-based intrusion detection model, utilizing Bayesian networks and random trees as fundamental classifiers, meta-learning algorithms for random submission and voting, and presenting the KDD-99 data set for model performance evaluation. However, this model’s detection accuracy for U2R assaults with a small sample size is worse than that for other attacks. Ashfaq et al. [9] presented a fuzziness-based semisupervised learning technique for intrusion detection systems. The algorithm for unlabeled sample-aided supervised learning is used to enhance the performance of the classifier in an intelligent decision support system. The neural network with random weights (NNRw) is used as the fundamental classifier to categorize unlabeled data with fuzzy quantity. However, this technique is limited to detecting normal and abnormal tasks and does not identify numerous attacks. Han et al. [10] proposed an intrusion detection model based on game theory and autoregressive models. The autoregressive theory model is improved to a noncooperative complete information static game model to predict the attack pattern. The model considers the energy consumption in the intrusion detection process and obtains the optimal defense strategy by analyzing the hybrid Nash equilibrium solution of the model, which balances the detection efficiency and the energy consumption of the system. Raja and Rabbani [11] presented an intrusion detection system based on support vector machines (SVM) and principal component analysis (PCA). Dimensionality reduction is accomplished through the use of principal component analysis. The standard data KDD-99 is used to evaluate the role of different cores in an intrusion detection system using support vector machine and principal component analysis, which significantly reduces data analysis time and improves intrusion detection performance, but the system is unable to distinguish between different types of attacks. Mohapatra et al. [12] introduced several security threats that arise in WSNs and a malicious node detection method based on the base station machine learning (ML) algorithm (BS). BS identifies harmful behaviors in the network and sends notifications to neighboring nodes to prevent attackers from attacking. The model trains an ML algorithm to classify attacker nodes efficiently and correctly within the BS. Gao et al. [13] suggested a self-coding network-based support vector machine intrusion detection model. The Boltzmann machine is used to lower the dimension of data, the self-coding network is utilized to acquire the ideal low-dimensional data, and SVM is used as a classifier for intrusion recognition. However, the classifier performs poorly against R2L and U2R assaults. Sun et al. [14] proposed an intrusion detection model for wireless sensor networks based on an improved v-detector algorithm. They modified the v-detector algorithm by modifying the detector generation rules and optimizing the detector and then used principal component analysis to reduce the detection characteristics. However, the detection effect of new attacks is insufficient. Gite et al. [15] proposed a MITM-intrusion detection system (MITM-IDS) model for a variety of attacks such as man-in-the-middle (MITM) and black holes due to the decentralized structure of wireless sensor networks. The model is an attack-tolerant intrusion detection system by training nodes that are likely to be attacked, enabling more accurate detection, isolation of attacks, and reconfiguration of attacked nodes. Al-Qatf et al. [16] presented an effective deep learning approach based on a framework for autonomous learning. This approach, which employs a sparse automated encoder mechanism for feature learning and dimension reduction, significantly lowers training and testing time and significantly improves the prediction accuracy of SVMs subjected to assault. However, this technique focuses on deep learning’s capability for feature reduction. It primarily uses deep learning for pretraining and classifies using the conventional supervision mode. This technique performs better with two classes than with several classes. Idhammad et al. [17] presented a technique for detecting denial-of-service attacks using an artificial neural network. The approach optimizes a feed forward neural network (FNN) to identify denial-of-service threats with high accuracy. Simultaneously, the fewest resources are consumed. The experimental performance
is evaluated using two data sets, UNSW-NB15 and NSL-KDD. However, this approach has been tested exclusively against denial-of-service attacks.

Through the study of the above linked research, the present intrusion detection methods have the drawbacks of high data dimension, feature redundancy, and low detection rate of some assaults and cannot successfully detect a range of attack types. In the intrusion detection model proposed in this paper, the features of high-dimensional data are extracted through neighborhood rough set, the SMOTE algorithm performs sample balance and improves the detection rate of a few types, and the random forest realizes attack classification to effectively detect multiple types of attacks.

3. Intrusion Detection Model

The method for developing an intrusion detection model for multitype assaults presented in this article is depicted in Figure 2, and it comprises five components: data preparation, feature selection, intrusion detection (sample balance and attack categorization), and model assessment.

3.1. Network Model. We propose an intrusion detection model based on a hierarchical UWSN structure. As shown in Figure 3, nodes are self-grouped into clusters using a predefined cluster formation algorithm [18], each cluster contains a cluster head node and several common sensor nodes, the common sensor nodes within the cluster collect data and send it to the cluster head node of the cluster they are in through a single hop or multihops, and the cluster head node uploads the data to the aggregation node.

The cluster head node manages and directs the computational work of the whole cluster on a global scale. After collecting data from conventional sensors, a heuristic reduction technique based on attribute importance is chosen through a neighborhood rough set, the SMOTE algorithm performs sample balance and improves the detection rate of a few types, and the random forest realizes attack classification to effectively detect multiple types of attacks.

3.2. Data Preprocessing. The collection of attack data is the responsibility of standard sensor nodes. The data is made of multidimensional features, and the values of various feature attributes correspond to distinct data categories. In a black hole attack, the attacker acts as the cluster head node and continuously discards packets sent by the cluster’s member nodes, and the amount of packets forwarded to the sink node has an attribute value of zero. In a flooding attack, the attacker broadcasts a large amount of high-transmission power information at the cluster head, which consumes the cluster’s energy [19, 20].

We often associate different data characteristics with distinct dimensions and kinds. Preprocessing the data is necessary to remove the impact of attribute types and dimensions on the data findings and to improve the performance of the intrusion detection model. Preprocessing is primarily concerned with digitization and normalization. Digitization is converting character attribute characteristics in the data to numerical data; normalization is mapping the encoded data to the same interval, so that the features have the same measurement scale. We performed normalization using the min-max method:

\[ f(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \]

where \( x_{\max} \) is the maximum value of the sample data and \( x_{\min} \) is the minimum value of the sample data.

3.3. Feature Selection. The cluster head and sensor nodes collaborate to process the data, with the cluster head utilizing the neighborhood rough set for feature extraction. Due to the fact that the data collected by ordinary sensor nodes is high-dimensional mixed data, in order to avoid the issue of excessive load caused by too many data dimensions, redundant features are removed via dimensionality reduction in the cluster head using powerful computing power, and the
smallest feature subset with the same classification ability as the original data set is obtained.

Pawlak et al. [21] introduced rough set theory as an efficient approach for dealing with imprecise, ambiguous, and incomplete symbolic data and primarily discrete data. A neighborhood rough set is a rough set that has been extended through neighborhood relationships in order to handle continuous data [22].

The calculation process of attribute reduction based on the neighborhood rough set is as follows:

The collected information is regarded as the information system

\[ IS = \{ U, A, V, f \} \]

where \( U \) is the universe, \( A = C \cup D \) is the attribute set, \( C \) is the conditional attribute, \( D \) is the decision attribute, \( V \) is the attribute value set, \( f \) is the mapping of \( U \times A \rightarrow V \), and \( B \subseteq C \) is the attribute subset.

3.3.1. Neighborhood Radius. For each attribute in \( a \), calculate the neighborhood radius \( r \) according to the standard deviation formula:

\[ r = \frac{s}{\lambda}, \quad \lambda = 2 \sim 5. \]  

The standard deviation formula is

\[ s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}, \]

where \( n \) is the number of samples in each attribute and is the average value of \( x \).

3.3.2. Neighborhood and Neighborhood Relationship. For \( x_i \in U \), the neighborhood \( \delta_B(x_i) \) of \( x_i \) in attribute subset \( B \) is

\[ \delta_B(x_i) = \{ x_j \mid x_j \in U, \Delta_B(x_i, x_j) \leq r \}. \]

The neighborhood relationship \( N_B(X) \) is the lower approximation of the sample subset \( X \) with respect to \( B \), and it is also the positive domain POS\(_B\)(\( X \)) of \( X \) with respect to \( B \):

\[ \text{POS}_B(X) = N_B(X) = \{ X \mid \delta_B(X) \subseteq X, X \in U \}. \]  

3.3.3. Neighborhood Dependence and Importance. Dependency of decision attribute \( D \) on attribute subset \( B \) is

\[ \gamma_B(D) = \frac{\text{card}(\text{POS}_B(D))}{\text{card}(U)}, \]

where \( \text{card}() \) represents the number of elements in the set.

Attribute importance is the influence of a condition attribute on decision attributes. There are two calculation methods:

1. Calculation method for deleting attributes: if \( a \in B \), the importance of a certain attribute \( a \) in \( B \) with respect to the decision attribute \( D \) is

\[ \text{SIG}(a, B, D) = \gamma_B(D) - \gamma_{B-\{a\}}(D). \]

2. Calculation method of added attributes: if \( a \in C - B \), the importance of attribute \( a \) relative to \( B \) on decision attribute \( D \) is

\[ \text{SIG}'(a, B, D) = \gamma_{B+\{a\}}(D) - \gamma_B(D). \]

If the importance is zero, the attribute is redundant.

3. Reduced attribute subset

While

\[ \begin{align*}
\forall a \in B, & \, \gamma_{B-\{a\}}(D) < \gamma_B(D), \\
\gamma_B(D) = \gamma_C(D),
\end{align*} \]

then the attribute subset \( B \) is a reduction of \( C \).
Algorithm 1: Attribute reduction process of neighborhood rough set.

Algorithm 1 summarizes the major stages involved in feature extraction from a neighborhood rough set. To begin, the neighborhood dependence and importance of data feature attributes are calculated to determine the core attribute; the suboptimal attribute is then chosen to determine the optimal reduction subset; and finally, the optimal reduction subset is transmitted to the sink node via the cluster head node.

3.4. Intrusion Detection. Intrusion detection may be viewed as a classification issue, with the data set being partitioned into normal and attack segments. This section is comprised of two sections. SMOTE is used to oversample the data set, increase the amount of data for a few classes, reconstruct the data set, and resolve the class imbalance problem; the reconstructed training set is then trained using the random forest algorithm to generate a classifier capable of detecting multiple types of attacks.

3.4.1. Sample Balance. Unbalanced data is a relatively regular occurrence. There are 995 R2L samples in the NSL-KDD data set [23], but only 52 U2R samples. Direct categorization of imbalanced samples is useless, even if the classification accuracy is good. Sampling is used to address the issue of unequal distribution in the data layer and to enhance the method for balancing the data categories, therefore enhancing the classifier’s classification impact.

SMOTE is a heuristic oversampling approach introduced by Chawla et al. to address the issue of class imbalance [24]. The SMOTE method is used in this article to insert randomly produced additional samples between minority samples and their neighbors, therefore increasing the number of minority samples and improving the class’s imbalance [25–29]. In Figure 4, the circle symbolizes minority groups, whereas the square represents majority groups. The triangle displays the data interpolated and synthesized between the minority sample and its nearest neighbors for each minority sample.

The basic principle of the SMOTE algorithm is as follows: calculate the nearest neighbor set \( \hat{P} \) of the minority sample \( x_i \) in the training sample \( S \), randomly select a sample \( x_q \) in \( P \), and the difference on the corresponding attribute \( q \) of \( x_i \) and \( x_q \) is \( \text{diff}(q) = x_{iq} - x_{iq} \); then, the mathematical expression of the newly synthesized minority sample \( f_{iq} \) is

\[
\begin{equation}
    f_{iq} = x_i + (x_{aq} - x_{iq}) \times \text{rand}(0, 1),
\end{equation}
\]

where \( \text{rand}(0, 1) \) represents the random number in the (0, 1) interval.

Algorithm 2 describes the specific implementation process of the SMOTE algorithm through pseudo-code.

The newly synthesized minority samples are added to the initial training samples to increase the number of minority samples and reduce the degree of data imbalance, and then, use the classifier to classify the new training samples. Algorithm 3 lists the sampling process of the SMOTE algorithm.

3.4.2. Attack Classification. In wireless sensor networks, attackers compromise sensor nodes, eavesdrop on information, alter the integrity of data, and consume the energy of nodes; therefore, creating effective intrusion detection methods to detect known and unknown attacks is very important to maintain the security of wireless sensor networks. DOS attacks are considered to be the most common and dangerous attacks, and we have identified a variety of different DOS attacks, as described below: blackhole, gray-hole, and flooding are three DOS attack modes [30].

(i) Blackhole attack

In blackhole attack, the attacker starts the attack by disguising himself as a CH (cluster head) node. In the attacked cluster, the cluster member node CM (cluster member) sends packets to the CH node; in order to forward the packets to the BS (base station), the attacker pretends to be the CH node, receives the packets from the CM node, and then drops all of them without forwarding them to the BS. Algorithm 3 shows the steps of the algorithm of blackhole attack.
Grayhole attack is like blackhole attack in that the attacker receives packets from the CM nodes in the cluster by faking that he is the CH. Unlike blackhole attack, the attacker of grayhole attack discards the packets randomly or selectively after receiving them and sends the remaining data to the BS. Table 2 shows the algorithm steps of grayhole attack.

(iii) Flooding attack

In flooding attack, the attacker sends many broadcast messages to other sensor nodes by faking to be a CH node and sending them at high frequency, which may send 10 messages, or 20, 30, or 50, and when an ordinary sensor node receives many broadcast messages, it consumes a lot of energy and spend more time to choose which cluster to join, especially for the nodes that are far away; their energy consumption is the greatest. Table 3 shows the algorithm steps of flooding attack.

In order to identify whether the network is secure more precisely, this paper introduces trust assessment of nodes before classification, by calculating the trust value of nodes to determine in advance whether the nodes have suffered from attacks and become untrustworthy nodes. The trust value calculation of nodes mainly considers communication trust, including interactive trust and honesty trust and calculates the total trust value of nodes based on the resulting trust value and finally makes a preliminary judgment of node security by comparing with the trust threshold.

3.4.3. Interactive Trust. Interactive trust refers to the trust value calculated from the number of interactions between nodes. Interaction refers to all communication behaviors from one node to another node, including the sending and receiving of requests or data packets. The more the number of interactions between two nodes, the higher their trust value, but when the number of interactions exceeds the threshold, the trust value decreases because the nodes may encounter malicious interactions, such as in a flooding attack, where the attacker sends many requests to exhaust the node’s energy.

In this paper, the interaction trust value of CH\(_i\) nodes is calculated based on the number of interactions between node \(j\) and CH\(_i\) nodes in the time region, and the interaction between CH nodes and CM nodes can be abstracted as an undirected weighted graph, whose weights represent the number of interactions between nodes and whose trust values are

\[
\text{CIT}_{ij}(\Delta t) = \begin{cases} 
10 \times \frac{\omega_{ij}}{\max (\omega_{ij})}, & j \in G, \omega_{ij} \leq \lambda \mu, \\
10 \times \exp \left(\frac{-|\omega_{ij} - \mu|}{\theta}\right), & j \in G, \omega_{ij} > \lambda \mu,
\end{cases}
\]

where \([x]\) denotes the largest integer less than or equal to \(x\), \(\mu\) is the average number of nodes interacting with each other, \(\lambda\) is the parameter used to determine the upper limit of the number of interactions, \(\lambda \mu\) denotes the threshold of the interaction range, and \(\theta\) is a coefficient determined by \(\omega_{ij}\) with values of 1, 10, and 100 when \(\omega_{ij}\) is a single-digit, decimal, and 100-digit number, respectively.

3.4.4. Honesty Trust. We calculated honesty trust based on the successful and unsuccessful interactions between two nodes. We evaluate the honesty trust value of CH by its interaction with BS. If CH does not forward the collected packets to BS or does not forward all of them, it is unsuccessful interaction, such as when it encounters blackhole attack or grayhole attack. The higher the ratio of the number of successful interactions between nodes to the number of all interactions, the higher the node trust value.

The number of successful and unsuccessful interactions between CH\(_i\) node and BS node \(b_k\) during the time window is \(s\) and \(f\), respectively, and the trust value of CH node is

\[
\text{SHT}_{ik}(\Delta t) = \begin{cases} 
10 \times \frac{s + 1}{s + f + 2}, & f = 0, \\
10 \times \frac{s + 1}{s + f + 2} \times f^{-1/2}, & f \neq 0.
\end{cases}
\]

When there is an unsuccessful interaction, we perform \(f^{-1/2}\) to penalize it to lower the trust value.

3.4.5. Total Trust Value. The total trust value, which includes both interactive trust and honest trust, is calculated from the total trust value of BS and CM for CH nodes as follows.

\[
\text{COT}_i = [\alpha \text{CIT}_{ij} + \beta \text{SHT}_{ik}]
\]

where the parameters \(\alpha, \beta \in [0, 1]\) are the weights of each subtrust value. We consider each subtrust as an equally important trust and therefore the parameter \(\alpha = \beta = 1/2\).
3.4.6. Trust Threshold. By calculating the trust value of each CH, we select a trust threshold as the detection criterion, and its threshold is calculated as follows.

\[
TV = \frac{\text{avg} \in \text{CHS and } \text{COT}_i \geq 5}{\exists i, s.t. \text{COT}_i \geq 5, 5, \text{ others,}}
\]  

where CHS is the set of CHs and avg is the average function that compares the trust value with the threshold value, and we consider the node with trust value less than the threshold value as the attacked node.

We mark nodes as untrustworthy nodes when their trust value is reduced after being attacked. In order to more accurately determine whether an untrustworthy node is under attack and to detect the attack category, this paper uses the random forest algorithm to implement the attack classification. Random forest is an integrated learning model with the decision tree as the basic classifier, containing several decision trees trained by the bagging method [31], and the classification result is determined by the output vote of a single decision tree. Random forest overcomes the overfitting problem of decision trees and has good scalability and parallelism for high-dimensional data classification problems [32].
In this paper, we use the random forest algorithm to implement intrusion detection at the sink node and distinguish between normal data and multiple attack data in the new sample data, and the detailed classification steps are as follows.

**Step 1.** Intrusion data is divided into normal data and attack data. Assuming that the sample space of attack data is \( A \), \( A \) is composed of \( m \) types of attack samples, \( X_i \) represents the \( i \)-th attack sample, \( A = \{ X_1, X_2, \ldots, X_m \} \), each sample is composed of \( f \) features after feature extraction, and \( X_i = \{ F_{i1}, F_{i2}, \ldots, F_{id} \} \).

**Step 2.** For the minority sample \( X_j \) in the attack data, calculate its \( K \) nearest neighbor set \( P \) and select \( X_j \)'s neighbor \( X_j \) from the \( P \) set, where \( j = 1, 2, \ldots, k \).

**Step 3.** According to equation (10), the minority sample \( X_j \) and the randomly selected neighbor sample \( X_j \) synthesize a new sample on the corresponding attribute \( f \), which is represented by \( X_{new} \).

\[
X_{new} = X_j + (X_j - X_d) \times \text{rand}(0, 1).
\]

**Step 4.** Add the sampled new sample \( X_{new} \) to the original sample to construct a new data sample space \( R \).

**Step 5.** Resample the new sample \( R \) using the bootstrap method, and randomly generate \( n \) training sets \( D_1, D_2, \ldots, D_n \).

**Step 6.** Before selecting attributes for each nonleaf node, randomly select \( y \) (\( y \leq f \)) attributes from the \( f \) attributes as the split attribute set of the current node and split the node in the best split way; thus, make each training set produce its corresponding \( n \) decision trees.

**Step 7.** Use each decision tree to test the test samples to obtain corresponding \( n \) classification results.

**Step 8.** Using the voting method, the category with the most output in the classification results is the category of the test sample, and the results are used to determine whether an abnormal attack occurs in the network.

Random forest is composed of several decision trees, and there are several implementation algorithms for decision trees, among which the CART algorithm is used to create decision trees by expressing the model purity using Gini coefficient. The CART algorithm classifies the features in the samples according to them using a dichotomous method, and the training needs to make judgments about each feature of each data at each node. Therefore, its time complexity is approximated as

\[
O(m \times n \times D),
\]

where \( m \) is the sample size, \( n \) is the number of features in the data sample, and \( D \) is the depth of the tree.

For a uniformly bifurcated tree, there are roughly \( 2^D \) nodes at level \( D \). Let \( 2^D = m \), which yields \( D = \log_2 m \), so the complexity is approximated as

\[
O(m \times n \log_2 m).
\]

Table 3 illustrates the method of achieving attack categorization through the use of random forests. SMOTE creates new attack data, reconstructs the data set, and resolves the problem of data imbalance for a few forms of attack data. This approach enables intrusion detection to get more accurate classification findings and hence increase detection accuracy.

The number of decision trees in a random forest has an effect on how well it generalizes. It is more critical to choose an adequate number of decision trees when taking into account both the number of decision trees and the modeling speed. The number of decision trees picked by this model will be demonstrated in the experiment's subsequent section.

### Table 3: NSL-KDD experimental data identification type.

| Marking type | Meaning                  | Specific categories                      |
|--------------|--------------------------|------------------------------------------|
| Normal       | Normal type              | back, land, neptune, pod, smurf, teardrop |
| DOS          | Denial of service        | ipsweep, nmap, portsweep, satan          |
| Probe        | Surveillance and probing | ftp_write, guess_passwd, imap, phf, multihop, spy, warezclient, warezmaster |
| R2L          | Remote to local          | buffer_overflow, loadmodule, perl, rootkit |
| U2R          | User-to root             |                                          |

3.5. Model Evaluation. The evaluation of the intrusion detection model proposed in this paper is measured by accuracy (ACC), precision (P), detection rate (TPR), and false alarm rate (FPR) [34]. The combination of true and predicted values of the samples can be classified into four cases: true cases (TP), false positive cases (FP), true negative cases (TN), and false negative cases (FN). TP indicates the number of cases in which the model predicts a positive result and the actual value is also positive; FP indicates the number of cases in which the model predicts a negative result and the actual value is also negative. FN indicates the number of cases where the model predicts negative cases yet the actual value is positive [35, 36]. It is represented by the confusion matrix shown in Table 2.
ACC is the ratio of correctly classified samples to the total number of samples indicating the accuracy of the prediction results.

\[
ACC = \frac{TP + TN}{TP + FN + FP + TN}. \tag{18}
\]

\(P\) is the ratio of samples that are predicted to be positive.

\[P = \frac{TP}{TP + FP}. \tag{19}\]

TPR is the probability that a positive sample will be paired in a sample with a positive true value.

\[TPR = \frac{TP}{TP + FN}. \tag{20}\]

FPR is the probability that a sample with a negative true value is incorrectly classified as a positive sample.

\[FPR = \frac{FP}{TN + FP}. \tag{21}\]

4. Experimental Results and Analysis

In this section, we evaluate the effectiveness of the algorithm proposed in this paper by four metrics: accuracy rate (ACC), precision rate \((P)\), detection rate (TPR), and false alarm rate (FPR). First, we verified the classification performance of feature extraction, comparing the correct rate that can verify the classification effect and the modeling time that affects energy consumption. Second, we examined the balancing effect of the data, comparing the detection accuracy before and after sampling for each attack type. Finally, we evaluated the performance of random forest intrusion detection, comparing it with four classifiers, naive Bayes, decision tree, ELM, and bagging, in terms of total correctness with four evaluation metrics.

4.1. Data Preprocessing. Since attack detection is rarely used underwater and there is currently no relevant attack detection data set, this paper uses the KDD-99 [37] data set, which is widely recognized in the field of intrusion detection. KDD-99 is collected from a simulated US Air Force LAN. Nine weeks of network connectivity data, split into labeled training data and unlabeled test data. The test data and training data have different probability distributions. The test data contains some attack types that do not appear in the training data, which makes intrusion detection more realistic. NSL-KDD eliminates redundancy and repetition in KDD-99 recorded data. NSL-KDD contains 1 normal behavior identifier and four attack types DOS, Probe, R2L, and U2R. The four attack types consist of 22 training attack data, as shown in Table 3.

In this paper, 20% of the data in NSL-KDD are used as experimental data, and the specific distribution of the number and percentage of each data type is shown in Table 4.

There are four types of assaults in this data set: Dos, Probe, U2R, and R2L. Selective forwarding and resource consumption can be classed as DoS attacks in wireless sensor network assaults; a probe attack requires scanning network nodes before it can be characterized as a probe attack. Flooding is an internal assault that falls under the U2R classification; attacks that use network vulnerabilities go under the R2L classification [14].

Each record in the NSL-KDD data set has 41-dimensional features, and the 41-dimensional attributes comprise several class attribute features, as indicated in Table 5, including protocol type, service, and flag, which are encoded as numerical data. Protocol type contains three distinct values: tcp, udp, and icmp, which are denoted by the sequence numbers 1, 2, and 3, respectively, and service and flag are treated similarly.

Normalize the encoded data and map the feature values to the 0-1 interval to ensure that the values of distinct characteristics are similar. Features with a big distribution difference have the same weight influence on the model as those with a small distribution difference. In this work, we utilize equation (1) to normalize the data per column, with the maximum and lowest values of each column used as \(x_{\text{max}}\) and \(x_{\text{min}}\) in equation (1), and we generate the normalized training samples by calculating each column of data.

4.2. Experimental Analysis. The description and analysis of the experiments in this section comprises two main parts.
First, we set different parameters to derive the data after feature extraction and validate the classification performance of the data. Second, we verify the effectiveness of the model before and after sampling, and verify the effectiveness of the random forest by comparing the correct rate with other classifiers.

### 4.2.1. Feature Extraction Classification Performance

The energy and resources of sensor nodes are limited. First, the cluster head node uses neighborhood rough set to extract features. The parameters of the neighborhood rough set mainly include the neighborhood radius and the importance matrix. The neighborhood radius is calculated by dividing the standard deviation by $\lambda$, which can take a value of 2 to 5, and the lower limit of the importance is set to control the selected characteristic parameters. After setting different values of $\lambda$ and the lower limit of importance, it is concluded that when $\lambda$ is 5 and the lower limit of importance is 0.00001, extract \{1,2,3,4,8,10,14,23,30,31,32,33,35,36,37,40,42\} a total of 17-dimensional attribute features; its classification accuracy is the best.

We compared the correct rates and modeling times of four classifiers, naive Bayes, tree, bagging, and random forest, to verify the classification performance of 17-dimensional data (Figure 6). Figure 6(a) shows that the overall correct rates before and after feature extraction are basically the same, showing that the data after feature extraction can achieve the classification effect before feature extraction. Figure 6(b) shows that the modeling time of 17-dimensional features on all four classifiers is less than half of the modeling time of 42-dimensional features, which can effectively reduce the computational energy consumption. It is concluded that the feature extraction method based on the neighborhood rough set proposed in this paper can not only maintain the classification performance of the data but also reduce the modeling time.

### 4.2.2. Attack Detection

In order to verify the impact of SMOTE oversampling on random forest attack detection, we use the SMOTE algorithm at the sink node to add samples to the imbalance samples U2R and R2L, extract some data from the majority class attack in the data set, set the label attribute of the minority class U2R and R2L to 1, and set the label attribute of the rest of the data to 0. The new training samples are got by the SMOTE linear interpolation method, and the trust values of the cluster head nodes in the samples are calculated, as well as random forest training new data to complete the intrusion detection.

We calculate the trust values of cluster head nodes based on equations (11) to (13) mentioned in the text for the sample-balanced data and initially determine whether the cluster head nodes are trustworthy nodes. Figure 7 intercepts the trust value of some cluster head nodes within a certain time window and calculates the trust threshold of nodes within this time period based on the trust value.

In Figure 7, id denotes the number of the cluster head node, Trust_value denotes the node trust value, and we filter the nodes with trust value greater than 5. We calculate the threshold value of nodes with trust value greater than 5 as 7. From this, we know that when the trust value is less than or equal to 7, it is an untrustworthy node, and when the trust value is greater than 7, the node is trusted.

The trust value can initially determine the state of the cluster head node, and for the untrustworthy nodes, we use random forest to identify which node is under attack. Figure 8 shows the results of the influence of the number of decision trees on the correct classification rate of random forest. When the number of decision trees is 50, the classification rate reaches 99.287%, and when the number of decision trees is 150, the classification rate is 99.272%. After considering the number of decision trees included in the random forest and the modeling speed, we choose 50 decision trees to achieve the best result of random forest classification.

We split the balanced assault data into a training and test set and send them to a random forest classifier for training simulation, comparing the classification results before and after sampling. As seen in Figure 9, the accuracy of minority R2L prior to SMOTE sampling is only 78 percent, whereas U2R is just 20%, indicating a poor detection impact. As a result, the classifier sampled by the SMOTE algorithm has a significant improvement in detecting imbalanced data.

In order to verify the effectiveness of the random forest classifier, we compared the accuracy of the naive Bayes, ELM, trees, bagging, and Random Forest classifiers. The comparison result is shown in Figure 10. It can be seen that among the five classifiers, the accuracy of random forest is as high as 99.39%, and decision trees and bagging are slightly lower. Moreover, random forest can also reduce the possibility of decision trees that are easy to overfit. Therefore, random forest has the best effect in realizing the classification of multiple types of attacks.

To further verify the effectiveness of the model proposed in this paper, we compared the accuracy of the proposed SMOTE-random forest model with decision tree and ELM on five different attack types, as shown in Figure 11. It can...
Figure 6: Comparison of classification ability and modeling time before and after feature extraction: (a) comparison of accuracy before and after feature extraction; (b) comparison of modeling time before and after feature extraction.

Figure 7: Trust value of some cluster head nodes.
Figure 8: The influence of the number of decision trees on the performance of random forest.

Figure 9: Comparison of detection accuracy before and after SMOTE sampling.
be seen that the SMOTE-random forest model represented by the blue curve has higher accuracy than decision tree and ELM on all five different attack types.

4.3. Model Evaluation. To assess the effectiveness of the intrusion detection model presented in this work, we use the evaluation index assessment model described in Section 3.3 and derive the evaluation results for each index from the confusion matrix of the model’s prediction outcomes. As shown in Table 6, the accuracy rate ACC and the accuracy rate \( P \) of this model may exceed 99 percent for the five categories of data, and the same result can be obtained for imbalanced data. The detection rate TPR is similarly close to 99 percent, and the false alarm rate FPR is also less than 0.2 percent for all four assaults, at 0.074 percent, 0.0296 percent, 0.188 percent, and 0.094 percent, respectively. As can be shown, the model we provided has a high detection rate for a variety of assault types.

Additionally, this study examines the performance of the four classifiers naive Bayes, ELM, tree, and bagging, as well as the performance of each random forest index on the five data types included in this research. Figure 12 illustrates the results. Figures 12(a) and 12(b) illustrate the results of a comparison of each classifier’s accuracy (ACC) and precision (\( P \)) values on five different types of data. As can be observed, the random forest classifier presented in this paper has the greatest ACC value across all five data sets, exceeding 99 percent. On the normal type, naive Bayes has the lowest ACC, with a value of just 77.2 percent. Both naive Bayes and ELM have lower accuracy rates. The true positive rate (TPR) and false positive rate (FPR) index values for each classifier are shown in Figures 12(c) and 12(d). As shown in the image, the random forest classifier suggested in this article has the best detection rate and the lowest false alarm.

### Table 6: Intrusion detection model evaluation index value.

| Performance | Normal | Dos | Probe | U2R | R2L |
|-------------|--------|-----|-------|-----|-----|
| ACC         | 99.53% | 99.9% | 99.85% | 99.79% | 99.71% |
| P           | 99.22% | 99.78% | 99.57% | 99.21% | 99.29% |
| TPR         | 99.52% | 99.84% | 98.11% | 99.79% | 98.26% |
| FPR         | 0.46%  | 0.074% | 0.0296% | 0.188% | 0.094% |

![Figure 10: Comparison of classification performance of different classifiers.](image-url)

![Figure 11: Comparison of detection accuracy on different types.](image-url)
rate when compared to other classifiers. As seen in Figure 12, the random forest classifier achieves the best value for each indicator and is stable, followed by tree and bagging, whereas naive Bayes and ELM perform poorly on various indicators. The experimental findings indicate that our proposed random forest classifier outperforms the other four classifiers in terms of overall performance and detection performance for different attack types.

5. Conclusions

We present a model for underwater sensor network intrusion detection that is applicable to a variety of assault types. The hierarchical design can help spread node energy usage and alleviate communication burdens. The model extracts features from the cluster head using a neighborhood rough set and then classifies attacks using SMOTE-random forest to accomplish the final intrusion detection. The feature extraction algorithm SMOTE-random forest balances minority samples and detects intrusions; the algorithm SMOTE-random forest extracts attributes of high importance to the sample, thereby reducing the computational load and speeding up classification. We demonstrate through simulation tests that this model is capable of not only identifying different forms of assaults but also of detecting attacks with a small sample size. Additionally, the classification performance of Naive Bayes, ELM, decision trees, and bagging was examined. Random forest has a better sensitivity to detection. This model is intended to balance the samples at the data layer and, in further research, to achieve sample balance at the algorithm layer.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.
Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This study was financially supported by the Key Project of Hainan Province under Grant ZDYF2020199 and the National Natural Science Foundation of China under Grant 61862020.

References

[1] Z. H. Liu and J. J. Zheng, "Ocean monitoring system based on underwater wireless sensor network," in Proceedings of 2010 Asia-Pacific Conference on Information Network and Digital Content Security (2010APCID), Beijing, China, 2010.

[2] A. Rahman, S. Roy, M. S. Kaiser, and M. S. Islam, "A lightweight multi-tier S-MQTT framework to secure communication between low-end IoT nodes," in IEEE 2018 5th International Conference on Networking, Systems and Security (NSysS), pp. 1–6, Dhaka, Bangladesh, 2018.

[3] Z. H. Zhang, Research on Key Technologies of Intrusion Detection in Wireless Sensor Networks, Beijing University of Posts and Telecommunications, 2018.

[4] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Adaptive and channel-aware detection of selective forwarding attacks in wireless sensor networks," IEEE Transactions on Wireless Communications, vol. 15, no. 5, pp. 3718–3731, 2016.

[5] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 266–282, 2014.

[6] W. C. Lin, S. W. Ke, and C. F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," Knowledge-Based Systems, vol. 78, pp. 13–21, 2015.

[7] A. A. Aburoma and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," Applied Soft Computing, vol. 38, pp. 360–372, 2016.

[8] Y. Wang, Y. J. Shen, and G. D. Zhang, "Research on intrusion detection model using ensemble learning methods," in 2016 7th IEEE International conference on software engineering and service science (ICSESS), pp. 422–425, Beijing, 2016.

[9] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas, and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," Information Sciences, vol. 378, pp. 484–497, 2017.

[10] L. Han, M. Zhou, W. Jia, Z. Dalil, and X. Xu, "Intrusion detection model of wireless sensor networks based on game theory and an autoregressive model," Information Sciences, vol. 476, pp. 491–504, 2019.

[11] M. C. Raja and M. M. A. Rabbani, "Combined analysis of support vector machine and principle component analysis for IDS," in 2016 International Conference on Communication and Electronics Systems (ICCES), pp. 1–5, Coimbatore, India, 2016.

[12] H. Mohapatra, S. Rath, S. Panda, and R. Kumar, "Handling of man-in-the-middle attack in WSN through intrusion detection system," International Journal, vol. 8, no. 5, pp. 1503–1510, 2020.

[13] N. Gao, L. Gao, Y. Y. He, and A. Wang, "A lightweight intrusion detection model based on self-coding network feature dimensionality reduction," Electronic Journal, vol. 45, no. 3, pp. 730–739, 2017.

[14] Z. Sun, Y. Xu, G. Liang, and Z. Zhou, “An intrusion detection model for wireless sensor networks with an improved V-detector algorithm,” IEEE Sensors Journal, vol. 18, no. 5, pp. 1971–1984, 2018.

[15] P. Gite, K. Chouhan, K. M. Krishna, C. K. Nayak, M. Soni, and A. Shrivastava, "ML based intrusion detection scheme for various types of attacks in a WSN using C4.5 and CART classifiers," Materials Today: Proceedings, 2021.

[16] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," IEEE Access, vol. 6, pp. 52843–52856, 2018.

[17] M. Idhammad, K. Afdel, and M. Belouch, "DoS detection method based on artificial neural networks," International Journal of Advanced Computer Science and Applications, vol. 8, no. 4, pp. 465–471, 2017.

[18] K. G. Omeke, M. S. Mollie, M. Ozturk et al., “DEKCS: a dynamic clustering protocol to prolong underwater sensor networks,” IEEE Sensors Journal, vol. 21, no. 7, pp. 9457–9464, 2021.

[19] Y. Yu, K. Li, W. Zhou, and P. Li, "Trust mechanisms in wireless sensor networks: attack analysis and countermeasures," Journal of Network and Computer Applications, vol. 35, no. 3, pp. 867–880, 2021.

[20] M. Mahajan, K. T. V. Reddy, and M. Rajput, "Design and simulation of a blacklisting technique for detection of hello flood attack on LEACH protocol," Procedia Computer Science, vol. 79, pp. 675–682, 2016.

[21] Z. Pawlak, L. Polkowski, and A. Skowron, "Rough set theory,"KI, vol. 15, no. 3, pp. 38–39, 2001.

[22] Y. Chen, Y. Xue, Y. Ma, and F. Xu, "Measures of uncertainty for neighborhood rough sets," Knowledge-Based Systems, vol. 126, pp. 226–235, 2017.

[23] H. Ji, D. Kim, and D. Shin, "A study on comparison of KDD Cup 99 and NSL-KDD using artificial neural network," Advances in Computer Science and Ubiquitous Computing, vol. 474, pp. 452–457, 2017.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[25] N. K. Mishra and P. K. Singh, "Feature construction and smote-based imbalance handling for multi-label learning," Information Sciences, vol. 563, pp. 342–357, 2020.

[26] J. Sun, H. Li, H. Fujita, B. Fu, and W. Ai, "Class-imbalanced dynamic financial distress prediction based on Adaboost-SVM ensemble combined with SMOTE and time weighting," Information Fusion, vol. 54, pp. 128–144, 2020.

[27] D. Elreedy and A. F. Atiya, "A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance," Information Sciences, vol. 505, pp. 32–64, 2019.

[28] S. Wang, S. Liu, J. Zhang et al., "A new method of diesel fuel brands identification: SMOTE oversampling combined with XGBoost ensemble learning," Fuel, vol. 282, article 118848, 2020.

[29] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," Information Sciences, vol. 465, pp. 1–20, 2018.
[30] I. Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "WSN-DS: a dataset for intrusion detection systems in wireless sensor networks," *Journal of Sensors*, vol. 2016, Article ID 4731953, 16 pages, 2016.

[31] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[32] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–36, 2018.

[33] S. Otoum, B. Kantarci, and H. T. Mouftah, "A novel ensemble method for advanced intrusion detection in wireless sensor networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 2020.

[34] T. Shawly, M. Khayat, A. Elghariani, and A. Ghafoor, "Evaluation of HMM-based network intrusion detection system for multiple multi-stage attacks," *IEEE Network*, vol. 34, no. 3, pp. 240–248, 2020.

[35] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: a survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, article 102767, 2020.

[36] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, "The area under the precision-recall curve as a performance metric for rare binary events," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, 2019.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, Ottawa, ON, Canada, 2009.