Research Article

Cyber Intrusion Detection Based on a Mutative Scale Chaotic Bat Algorithm with Backpropagation Neural Network

Xuebin Xu, Hu Qin, and Jie Zhou

College of Information Science and Technology, Shihezi University, Shihezi 832000, China

Correspondence should be addressed to Jie Zhou; jiezhou@shzu.edu.cn

Received 14 June 2022; Revised 6 August 2022; Accepted 10 August 2022; Published 10 September 2022

Academic Editor: Jawad Ahmad

Copyright © 2022 Xuebin Xu 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.

In the current complex cyber environment, malicious attacks take various forms and constantly change. To overcome the problems of low detection efficiency and high false alarm rate in traditional intrusion detection techniques, a new method of network intrusion detection combining the variable scale chaos bat (MSCB) algorithm and BP neural network (BPNN) called MSCB-BPNN is proposed in this paper. In the proposed approach, a new MSCB algorithm is proposed to improve the BPNN by optimizing its thresholds and weights so as to prevent it from falling into local optimum. To verify the practical classification performance of the proposed method for intrusion detection, some experiments are carried out. In these experiments, the proposed approach is compared with BPNN, SA-BPNN, and GA-BPNN on the benchmark intrusion detection datasets KDD cup 99 and UNSW-NB15. The experimental results show that the accuracy of the proposed method can reach 99.4% and 99.8% accuracy on the KDD cup 99 training and test sets, respectively, while 89.0% and 93.9% accuracy on the UNSW-NB15 training and test sets, respectively, which are higher than BPNN, SA-BPNN, and GA-BPNN. Furthermore, the precision and recall ratio of MSCB-BPNN are also superior to other approaches. Thus, the proposed model has significant advantages over BPNN, SA-BPNN, and GA-BPNN methods.

1. Introduction

The security of computer networks has received an increasing amount of focus as a natural consequence of the ongoing development of both computer technology and networks. However, there are a large number of complex attacks in the network, and these attacks have become the main threats to network and information security [1]. As the second line of defense after the firewall, intrusion detection has proactive and dynamic defense capabilities, effectively making up for the deficiencies of traditional security technologies [2–4]. Simultaneously, intrusion detection technology may assist system administrators with security audits, monitoring, attack detection, and response and increase their security management skills.

The commonly used intrusion detection techniques can be divided into two types: anomaly detection and misuse detection [5]. Misuse detection is to match the existing attack data information features with the data in the host or network, and then judge whether the host or network is in an abnormal state through the matching result. However, this detection method can only detect the types of network attacks that already exist in the sample database. The anomaly detection method is to first establish basic features with normal behavior trajectories in the sample database, and then treat all behaviors that deviate from or do not conform to the normal trajectory as intrusion behaviors [6].

Although intrusion detection, as an active and effective defense method, can predict intrusion behaviors in the time before they occur, there are still problems in how to improve the prediction accuracy. In recent years, with the rapid development of machine learning technology, it has been widely used in the fields of medicine, autonomous driving, image recognition, and natural language processing [7]. Because artificial neural networks in machine learning technology have good adaptive learning ability, multi-data parallel computing, strong anti-interference performance, and high-speed optimization ability, and can handle the
characteristics of distortion and incomplete data information, it is ideal for identifying intrusion data from complex, high-speed, and large network data. Therefore, improving the effectiveness and accuracy of intrusion detection using machine learning methods has become a research hotspot in the field of network security [8].

Network intrusion detection first needs to collect network status data of the target network, including connection duration, network layer protocol type, application layer service (protocol) type, etc., and then use probability and statistical analysis or machine learning technology to conduct network behavior. By using these technologies, the types of abnormal network behaviors can be accurately analyzed, so that network managers can take the corresponding security measures according to the detection results. The network state is affected by a variety of factors, and has the characteristics of randomness and time-varying. Therefore, network intrusion detection is essentially a nonlinear classification problem, and it is difficult for traditional linear classification algorithms to establish an optimal classifier. Due to the nonlinear and intelligent learning ability of machine learning algorithms, network intrusion detection classifiers based on neural networks and support vector machines have emerged. Among them, back-propagation neural network (BPNN) is a multi-layer feed-forward neural network with working signal forward propagation and error backpropagation. It has strong self-learning ability, generalization ability, and strong nonlinear mapping ability, and hence has become a most widely used network intrusion detection method [9–11].

Although BPNN can achieve better results than traditional intrusion detection, when using BPNN for network intrusion detection, if the BPNN parameters are not properly selected, it is difficult to accurately and comprehensively reflect network trends and intrusion dynamics, which may lead to low correct network intrusion detection rate and a rather high rate of false positives and missed positives of network intrusion [12, 13]. To solve this problem, this paper proposes a new intrusion detection approach named MSCB-BPNN. The approach utilizes the strong optimization ability of the Mutative Scale Chaotic Bat (MSCB) algorithm to help the BPNN select the optimal thresholds and weights to prevent it from being trapped in the local extreme values. Therefore, BPNN can obtain higher classification accuracy. In MSCB, the chaos theory is introduced, and the chaotic map is used to initialize the population to improve the randomness of individuals in the population. Moreover, a nonlinear mutative scale dynamic inertia weight mechanism is designed for the algorithm, which greatly improves the exploration and development capabilities of the algorithm. And the improved algorithm is then used to optimize the BPNN’s initial thresholds and weights. In this paper, the benchmark intrusion detection datasets KDD Cup 99 and UNSW-NB15 are used to verify the practical performance of the proposed MSCB-BP approach in intrusion detection. Compared with BPNN, SA-BPNN, and GA-BPNN in simulation experiments, it can be seen from the experiment results that MSCB-BP has tremendous advantages in detection accuracy, precision, and recall ratio. The major contributions of this paper are as follows:

1. A new MSCB algorithm is proposed to optimize the initial weight and threshold of BPNN, which is of great help to the training process of BPNN. By optimizing the initial thresholds and weights, the BPNN is effectively trained, and the classification accuracy is satisfactory.

2. To solve the problem of network intrusion detection based on network information, we introduced MSCB in BPNN. Using the newly proposed MSCB-BPNN model to accurately determine and discover suspicious attacks, the corresponding defense measures can be taken in time, allowing the network to avoid being attacked, thereby effectively reducing economic losses.

The rest of this paper is as follows: Section 2 describes some research work on intrusion detection technology and BPNN optimization. In Section 3, the MSCB-BP approach is proposed. To overcome the shortcomings of the BPNN, we introduced the MSCB algorithm and detailed the process of optimizing the BPNN in the MSCB algorithm. Section 4 applies the proposed model and the other three models to the KDD cup 99 and UNSW-NB15 intrusion detection datasets. By conducting comparative experiments, the experimental results are analyzed in Section 5. Section 6 summarizes the entire research work and briefly introduces the direction of future research.

2. Related Work

At present, intrusion detection technology has gradually become an important research direction in network security. Since it has the characteristics of real-time detection, dynamic response, and smart monitoring, it is an essential supplement to traditional security products. Early intrusion detection mainly depends on statistical probability analysis and the expert system [14]. The author of the paper [15] built the hidden Markov model in the operation status of computer systems. Paper [16] proposes an intrusion detection method based on a state conversion analysis tool (NetStat). Paper [17] examines how an expert system identifies intrusions using a rule-based pattern recognition engine. Paper [18] presents Kitsune: a plug and play NIDS that can learn to detect attacks on the local network without supervision, and in an efficient online manner. In paper [19], the researchers propose a two-stage layered network intrusion detection method called H2ID, which can help address new vulnerabilities and cyberattacks in the Internet of Things.

Compared with the traditional intrusion detection methods, using a neural network for intrusion detection has more advantages. Because the neural networks have the capability for self-learning, self-organization, and self-adaptation, they are able to start training without knowing the background in detail; intrusion detection technology based on the neural network has gradually become a hot spot of research. In paper [20], the Wireless Equivalent Privacy
(WEP) attack data are analyzed. However, no intrusion detection experiment is conducted. Paper [21] provides an intrusion detection system for DDOS that may quickly and effectively detect DDOS attacks. However, that system only targets particular sorts of attacks and has significant limits. In paper [22], Principal Component Analysis (PCA) was used to detect cyber intrusions. However, that algorithm's complexity is too high. To reduce the complexity of the algorithm, an intrusion detection approach based on DBN and probabilistic neural networks was proposed in paper [23]. Nevertheless, that method is only applicable to wireless networks with fewer levels.

Paper [24] proposes a dual support vector machine model, which is challenging to choose the correct parameters quickly, although the classification accuracy is improved. Paper [25] employs a deep self-coding network as a feature extractor in conjunction with a Softmax classifier to detect intrusions; however, the actual results are affected by artificial factors. Paper [26] proposes an improved method for intrusion detection in Bayesian networks based on deep learning and sliding windows to address attribute redundancy in the training sets. Nevertheless, working with data from a multi-dimensional network is difficult. Paper [27] proposes a least squares vector machine intrusion detection approach based on deep belief networks, which effectively solves the classification problem of massive intrusion data. Nevertheless, its detecting performance must be enhanced.

BPNN is an artificial feedforward neural network capable of handling complex nonlinear problems. It is composed of an input layer, a hidden layer, and an output layer. It has been widely used in many other fields due to the advantages of simple BPNN structure, adjustable parameters, multiple training algorithms, and good operability. However, BPNN has some inherent drawbacks, including that the performance is highly rest with the topology, initial connection weights and thresholds of connected nodes, and is prone to trapping into local optimum solutions [28]. Thus, to solve this challenge, some scholars have made use of the proposed improvement of BPNN parameters using swarm intelligence learning algorithms. Paper [29] uses genetic algorithm to improve BPNN and established the GA-BPNN model. Based on GA-BP, the paper [30] used the Bayesian regularization (BR) algorithm to train the GA-BPNN further to improve the accuracy of the GA-BPNN agent model. Although the research has enhanced the agent model’s fitting precision and computing efficiency to some degree, GA-based BPNN is still plagued by issues like early maturation and delayed convergence.

Paper [31] used an improved harmonic search (HS) algorithm to optimize the thresholds and weights of BPNN. Although the HS-BP approach provides superior generalization capability and prediction accuracy, it requires constant updating of the HS repository, which leads to inefficiency. In the paper [32], a new BPNN model based on particle swarm optimization algorithm and artificial bee colony algorithm (PSO-ABC) is proposed to perform network traffic prediction. When optimizing the thresholds and weights of the BPNN using PSO-ABC, PSO-ABC has good global search capability from the beginning to the end of the iterative process. However, the search performance is not robust against its neighborhood in the late iteration of the algorithm, and it cannot perform an effective local search, so a more accurate solution cannot be obtained.

By summarizing the above-related work, we present the research and problems on optimizing BPNN using optimization algorithms in Table 1. According to the above-mentioned scholars’ research, this paper proposes a new intrusion detection approach MSCB-BPNN to solve the shortcomings of the current BPNN-based intrusion detection system with low detection accuracy mentioned in Section 1, which takes the number of errors in the training data as the objective function and reduces the number of misclassifications and improves the classification accuracy through continuous iterative optimization. The proposed MSCB method uses the number of errors in the training data as the objective function to reduce the number of misclassification and improve the classification accuracy through continuous iterative optimization. The proposed MSCB method introduces chaos theory based on the bat algorithm. It uses chaos mapping to initialize the population to improve the randomness of individuals in the population. More importantly, the proposed algorithm also designs nonlinear convergence factors and dynamic weights, which gives the algorithm advantages in assisting the BPNN in determining the optimal thresholds and weights.

3. Cyber Intrusion Detection Based on MSCB-BPNN

This section describes the principle of BPNN, the detailed optimization process of the proposed MSCB algorithm for the initial weights of BPNN, and the application of the new MSCB-BPNN method to network intrusion detection. Table 2 lists all the notations used in this paper.

3.1. Principle of BPNN. The BPNN used in this paper is a three-layer feedforward neural network with error backpropagation. Each layer contains a certain number of neurons, and its model topology is shown in Figure 1. The learning process of a neural network includes forward propagation of signals and backpropagation of the output errors in some form through the hidden layer to the input layer; the error is distributed to each layer after the second input is performed. Based on the most rapid descent theory, the connection thresholds and weights of each layer are continuously adjusted so that the mean square error sum of the network output is minimized.

3.2. Parameter Optimization Based on MSCB. Since BPNN continuously corrects the weights and thresholds by backpropagating the error signal to minimize the error value between the actual output and the desired output, the classification accuracy of BPNN is affected by the weights and thresholds used to train the network. The weights and thresholds of traditional BPNN use random operators, which can lead to poor accuracy when the amount of original data is small. Therefore, to solve the above problem,
Table 1: Related work summary.

| Method                              | Problem                                                                 |
|-------------------------------------|-------------------------------------------------------------------------|
| GA-BPNN adopted in paper [29] and  | GA-based BPNN is still plagued by issues like early maturation and delayed convergence |
| paper [30]                          |                                                                         |
| HS-BP adopted in paper [31]         | Harmonic search (HS) algorithm requires constant updating of the HS repository, which leads to inefficiency |
| PSO-ABC adopted in paper [32]       | The search performance of PSO-ABC is not robust against its neighborhood in the late iteration of the algorithm, and it cannot perform an effective local search, so a more accurate solution cannot be obtained |

Table 2: Notations used in this paper.

| Notations                  | Description                                                                 |
|----------------------------|-----------------------------------------------------------------------------|
| Npop                       | Bat population size                                                         |
| G                          | Maximum quantity of epochs                                                  |
| \( A^0, A_i^t, A_{avg} \)  | Initial loudness, the loudness of the \( i \)th individual at time \( t \), the average loudness of all individual bats at time \( t \) |
| \( r_i^0, r_i^t \)         | The initial pulse emissivity of the individual, the pulse emission rate of the \( i \)th individual at time \( t \) |
| \( \psi \)                 | Pulse emission rate increase coefficient                                     |
| fit                        | Fitness function                                                            |
| \( x_i^t, x_0, x_{best}, x_{old} \) | The code of the \( i \)th individual at time \( t \), initial individual code, current global optimal individual code (solution), and current individual optimal coding (solution) |
| \( N_\text{input} \)       | Quantity of particles in the chaotic sequence                                |
| \( N_\text{input}, N_\text{hidden}, N_\text{output} \) | Number of neurons in the input layer, hidden layer, and output layer of BPNN |
| L                          | Individual bat code length                                                  |
| \( D \)                    | Dimensionality of individual bat coding                                      |
| \( d_{\text{min}}, d_{\text{max}} \) | The range of values for the weight and threshold |
| \( w_i \)                  | The nonlinear decreasing inertia weight at time \( t \)                      |
| \( w_0, w_1, \mu \)        | All are constants                                                           |
| \( \delta \)               | Stagnation algebraic threshold                                              |
| \( \beta, \epsilon \)      | Random number                                                               |
| \( \alpha \)               | Attenuation coefficient of the volume                                       |
| \( f_i^t, f_{\text{min}}, f_{\text{max}} \) | The value of the \( i \)th individual bat at moment \( t \), the minimum and maximum values of frequency taken |
| \( P \)                    | Bat individual replacement ratio                                            |

Figure 1: Model topology of the BPNN model.
this paper proposes a chaotic variable scale bat (MSCB) algorithm. In MSCB, a nonlinear convergence factor and a dynamic weight mechanism are designed, which dramatically improves the exploration and development capabilities of the algorithm. Moreover, chaotic perturbation is introduced in the iterative process, and the range of the perturbations decreases with the increase in the number of iteration steps, such that the classification accuracy of the algorithm can be accelerated at a later stage. In addition, MSCB uses random chaotic particles to replace some of the population particles to increase the particle diversity and thus avoid the algorithm falling into a local optimum. By using the proposed MSCB algorithm to optimize the BPNN, the thresholds and weights of the network are fixed and optimized, which can significantly increase the BPNN’s classification accuracy. As shown in Figure 2, the specific algorithm flow of MSCB is as follows:

Step 1. Configure the initial parameters of the MSCB algorithm. Parameters include bat population size, initial flight speed, loudness, pulse rate, frequency, volume decay coefficient, search frequency enhancement coefficient, and the maximum quantity of epochs.

Step 2. Randomly initialize the position of bats, bring them into the BPNN, let the number of errors in the training data as fitness, and select the current optimal in conformity with the fitness value.

Step 3. Update the position and velocity of the bat.

Step 4. Generate a random number Rand1 and determine whether it is greater than the pulse emission frequency of the bat. If so, generate a new bat position near the optimal solution using the chaotic perturbation strategy; otherwise, execute the next step of the algorithm directly.

Step 5. Generate a random number Rand2; if Rand2 is less than the loudness of the bat and \( f(x_{\text{new}}) < f(x_{\text{best}}) \), then replace the original with and update the pulse emission rate and loudness of each bat.

Step 6. Determine whether the algorithm is trapped in a local optimum. If the optimal bat individual is not updated within a certain number of iteration steps, it means that the algorithm is stuck in a local optimum, and it needs to help the algorithm jump out of the local optimum by variable scale chaos substitution.

Step 7. Check if the maximum number of iterations is reached; if yes, the algorithm terminates; if not, continue to execute steps 2–6.

3.3. Initialization of Bat Populations. The initialization of the bat population needs to ensure that the individuals in the population are evenly and widely distributed in the solution space such that the optimal configuration of BPNN weights and thresholds can be found. Therefore, this paper uses a chaotic mapping approach to initialize a bat population with population size \( N_{\text{pop}} \), which effectively improves the randomness of the bat individuals in the population. The most typical chaotic system nowadays is logistic mapping.

However, the experimental analysis has shown that the mapping is not uniformly distributed in the mapping space, which affects the algorithm’s optimization speed. Therefore, this paper uses a modified Tent chaotic mapping to initialize the population. The expression of this chaotic mapping is shown in (1). Random variables \( \text{rand} \times (1/N_T) \) are added to avoid the Tent mapping from falling into small or unstable periodic points without destroying the properties of chaotic variables, where \( N_T \) is the quantity of particles in the chaotic sequence; \( \text{rand} \) is a random variable with values in the range [0, 1].

\[
z_{d+1} = \begin{cases} 2z_d + \text{rand} \cdot \frac{1}{N_T} & 0 \leq z \leq \frac{1}{2} \\ 2(1 - z_d) + \text{rand} \cdot \frac{1}{N_T} & \frac{1}{2} < z \leq 1. \end{cases} \tag{1}
\]

The encoding length must be determined before generating the initial population using the chaotic mapping described above. In the bat population, the location information of individual bat represents an initial weight threshold of the BPNN; the bat location information encoding length \( L \) can be calculated by

\[
L = N_{\text{input}} \times N_{\text{hidden}} + N_{\text{hidden}} \times N_{\text{output}} + N_{\text{hidden}} + N_{\text{output}}.
\tag{2}
\]

where \( N_{\text{input}} \) indicates the quantity of neurons in the input layer, \( N_{\text{hidden}} \) indicates the quantity of neuron nodes in the hidden layer, and \( N_{\text{output}} \) indicates the quantity of neuron nodes in the output layer.

After the encoding length \( L \) is determined, a chaotic sequence of length \( L \) can be generated by (1). Then the chaotic sequence is carried to the solution space of the solution problem by (3). Repeating the \( N_{\text{pop}} \) step in this way eventually generates a population of bats uniformly distributed in the solution space.

\[
x_0 = d_{\text{min}} + (d_{\text{max}} - d_{\text{min}}) \cdot z_d,
\tag{3}
\]

where \( d_{\text{min}} \) and \( d_{\text{max}} \) denote the minimum and maximum weight and threshold values, respectively.

3.4. Change of Bat Position. It can be seen from the velocity update formula that the exploratory and exploitation capabilities of the algorithm can be adjusted by changing the velocity term coefficients. Therefore, a nonlinear decreasing inertia weight \( w \) is used in MSCB. In the early stage of the algorithm, \( w \) is large, which gives the algorithm a solid global search capability and accelerates the convergence of the algorithm. As the iterations proceed, \( w \) decreases exponentially, which makes the algorithm more capable of local search. This decreasing inertia weight can effectively balance the global exploration and local exploitation ability of the algorithm in different periods, which improves the accuracy of the algorithm. In this paper, the expression for the inertia weight \( w \) is defined as follows.
Initialize velocity, loudness, pulse emission rate, attenuation coefficient of the volume, pulse emission rate increase coefficient

Coding and initialization of bat populations using Eqs. (1)-(3)

The termination condition is satisfied

Calculate the fitness value of individual bats and record the individual with the highest fitness value

Update the position and velocity of the bat using equations (4)-(7)

Rand>r

No

Yes

Generate a new solution in the vicinity of the individual optimal solution using equation (8)

Recalculation of fitness values for bat populations

Rand<Ai && fit(x_i) > fit(x_best)

No

Keep the current solutions

Yes

Store the new solution and update the pulse emissivity and loudness according to Equation (9)

Caught in a local optimum

Yes

get out of the local optimum by chaotic perturbation

No

Yes

End

Start

Figure 2: Algorithm flowchart.
\[ w_i = w_{i_0} + w_h \left( \frac{w_h}{w_{i_0}} \right)^{-\mu(t-1)/G-1} \]  

Thus, the definition of the bat’s velocity and position update equation in MSCB is as follows:

\[ f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \quad \beta \in [0, 1], \]  

\[ v_i = \nu_i v_i^{t-1} + (x_i^{t-1} - x_{\text{best}}) f_i, \]  

\[ x_i = x_i^{t-1} + v_i. \]  

In the above equation, \( f_{\min} \) and \( f_{\max} \) denote the minimum and maximum values of frequency, respectively; \( \beta \) is a randomly distributed random number with values in the range \([0, 1]\); \( x_{\text{best}} \) denotes the current global optimal solution; and \( i \) denotes the serial number of bat individuals.

Furthermore, after the bat individual position and velocity update, it will generate a new solution locally near the current optimal position with a certain probability using (11):

\[ x_{\text{new}} = x_{\text{old}} + \varepsilon A_{\text{avg}}^t, \quad \varepsilon \in [-1, 1], \]  

where \( \varepsilon \) is a random variable in \([-1, 1]\) range and \( A_{\text{avg}}^t \) denotes the average loudness of all individual bats in this generation.

3.5. Adjustment of Pulse Emission Rate and Loudness. It is known from the biological mechanism of bat predation that in the process of searching for prey, the bat colony initially emits sonorous and low-frequency ultrasonic waves to extend the scanning range. After the prey is found, the loudness \( A_i \) is gradually reduced. At the same time, the pulse emission rate \( r_i \) is increased to precisely grasp the location of the prey. This algorithm uses (9) to model this predation feature.

\[
\begin{align*}
A_i^t &= \alpha A_i^{t-1}, \\
r_i^t &= r_i^0 \left[ 1 - e^{-\psi t} \right], \\
\text{s.t. } A_i^t &\rightarrow 0, r_i^t \rightarrow r_i^0, \text{as} \ t \rightarrow \infty,
\end{align*}
\]

where \( A_i^t \) is the loudness emitted by \( i^{th} \) bat at moment \( t \), and \( \alpha \) denotes the attenuation coefficient of the volume. \( r_i^0 \) denotes the initial pulse rate of \( i^{th} \) bat, which is also the maximum value of the pulse rate. \( r_i \) is the pulse emission rate of \( i^{th} \) bat at time \( t \). \( \psi \) is the pulse emission rate increase coefficient, which is a constant greater than zero.

Suppose the algorithm is trapped in a local optimum. In that case, it will be challenging to get out of the local optimum by chaotic perturbation at some time. To avoid this situation, we propose to get rid of the local optimum by using chaotic substitution. The specific idea is as follows. First, determine if the location of the best individual bat has changed in consecutive \( \delta \) generations of records, and if no change occurs, we decide that the algorithm has fallen into a local optimum. Then the algorithm will use chaotic mapping to generate a random chaotic sequence of size \( P \times D \) to replace a moderate number of bat individuals in the original bat population, where \( P \) is the replacement size. Meanwhile, as the number of steps of the algorithm into the local optimum increases, the size of the chaotic substitution increases linearly until the algorithm jumps out of the local optimum.

3.6. Computational Complexity Analysis. Usually, the complexity analysis of an algorithm is analyzing the efficiency of the algorithm. The analysis process mainly consists of measuring the time efficiency as well as the space efficiency of the algorithm operation. The complexity analysis of the algorithm will reflect whether the algorithm is superior.

In this subsection, since the spatial complexity of the algorithm is not required, the analysis of the computational complexity of the proposed algorithm MSCB in this paper is equivalent to the analysis of the temporal complexity of the algorithm. In addition, to facilitate the analysis of the complexity of MSCB-BPNN approach, we only consider the generation of random numbers and multiplication operations during one round of algorithm execution.

Since MSCB is an improvement to BA, we first analyze the complexity of BA. We assume that the time required to initialize the population size \( N_{\text{pop}} \), bat individual coding dimension \( D \), and other parameters in the bat algorithm is \( T_1 \), the time required to generate uniformly distributed random numbers is \( T_2 \), and the time required to calculate the fitness is \( T_f \); thus, the time complexity of the initial stage is expressed as

\[ O\left( T_1 + N_{\text{pop}} \times \left( DT_2 + T_f \right) \right) = O(D + T_f). \]

After entering the loop, assume that the time required to generate a \( D \)-dimensional chaotic sequence is \( DT_2 \) and the time required to update the velocity and position of an individual bat according to equations (4)–(7) is \( T_3 \); the time required for the judgment operation and the loudness adjustment operation is \( T_4 \); then the time complexity of this stage is expressed as

\[ O\left( N_{\text{pop}} \times \left( DT_2 + T_3 + T_4 + T_f \right) \right) = O(D + T_f). \]

In summary, the time complexity \( T(D) \) of a single round bat algorithm on the bat individual coding dimension \( D \) is represented by

\[ T(D) = O(D + T_f) + O(D + T_f) = O(D + T_f). \]

In MSCB, we only improve the mechanism of position update in the bat algorithm and introduce the mechanism of determining the trapped local optimum. These operations do not actually introduce multiplication operations; therefore, the complexity of the proposed MSCB and BA is equal to \( O(D + T_f) \).

3.7. MSCB-BPNN for Network Intrusion Detection. As shown in Figure 3, in this paper, our proposed network intrusion detection approach comprises three main phases. The first stage is dataset preprocessing, the symbolic feature attributes
in the intrusion detection dataset need to be processed numerically, and then all data are normalized to obtain the normalized raw data. The second stage is weight optimization. The network topology of the BPNN must first be established before optimizing the initial thresholds and weights of the BPNN. Then the prediction error of the classifier is selected as the fitness function of the model. The evolutionary conditions of MSCB are used to update the thresholds and weights of the BPNN continuously, and finally, the optimized MSCB-BPNN model is obtained. The third stage is to use the obtained MSCB-BPNN model to identify intrusions, output the detection results as an obfuscation matrix, and analyze the detection performance of the model.

4. Simulation

4.1. Datasets and Performance Metrics. This paper chooses the most widely used dataset KDD Cup 99 dataset and the UNSW-NB15 dataset from intrusion detection studies as the training and test datasets for MSCB-BPNN. KDD Cup 99 is derived from the Defense Advanced Research Projects Agency (DARPA) IDS data used for evaluation, and the dataset was also used in the KDD Cup 99 contest. The complete KDD Cup 99 dataset contains nearly 5 million input patterns, each data sample representing a network connection. To verify the classification performance of MSCB-BPNN and to increase the validation efficiency of the algorithm, we randomly selected a sample of 4500 data from the 10% version of the KDD Cup 1999 dataset for the analysis of the experiment, which contains 494021 instances. Each piece of data has one label and 41 features, including basic TCP link features, content TCP link features, host-based network traffic statistics features, time-based network traffic statistics features, etc. Each row has 42 elements. The first 41 elements are feature items, while the final item indicates the sort of data anomaly. The simulated attacks in the dataset contain four categories, DoS, Probe, U2R, and R2L, and the dataset also contains normal samples. The specific percentages of data distribution are shown in Table 3.

| Attack types | percentage | Training sets | Test set |
|--------------|------------|---------------|----------|
| Normal       | 0.1969     | 0.1948        |
| DoS          | 0.7924     | 0.7390        |
| Probe        | 0.0083     | 0.0134        |
| R2L          | 0.00023    | 0.0521        |
| U2R          | 0.0001     | 0.0007        |

The UNSW-NB15 dataset was established by the Australian Cyber Security Centre in 2015 and reflects modern network traffic patterns, containing a large amount of low-occupancy intrusion and deeply structured network traffic information. The dataset contains 2540044 data instances, including normal-type data and nine types of attack-type data. The specific percentages of data distribution are shown in Table 4. In addition, there are 49 features for each record in the dataset, and these features are divided into 6 categories, including Flow Features, Basic Features, Content Features, Time Features, Additional Generated Features, and Labelled Features, where Labelled Features contain the type and label of the attack.

To understand the specific detection performance of the model during the intrusion detection research, this paper mainly uses Accuracy, Precision, and Recall as the measurement metrics. Accuracy is the ratio of correctly classified samples to the total number of pieces and is calculated by

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}
\]

Precision is the probability that a sample is positive out of all the pieces predicted to be positive, and it is defined as

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{14}
\]

Recall is a measure of coverage, which measures the ability of the model to identify positive examples. It is defined as

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{15}
\]

In equations (13)–(15), TP, TN, FP, and FN denote correctly classified positive cases, correctly classified negative cases, incorrectly classified positive cases, and incorrectly classified negative cases, respectively.
cases, misclassified negative cases, and misclassified positive cases, respectively.

4.2. Simulation Experiment and Comparative Analysis. During this session, to evaluate how well the suggested model performs in terms of intrusion detection, we carried out several simulated experiments on a device with Matlab 2018, Ryzen 5700x CPU, and RTX3060 GPU installed. Tables 5 and 6 in this study illustrate the MSCB algorithm’s unique parameter configuration and the BPNN’s parameter configuration.

Figure 4 shows the optimization results of the MSCB algorithm under different epochs and population size. The number of misclassifications is used as the objective function in this paper. When the value of the function is smaller, it means that the model’s performance is better. It can be seen from Figure 4 that MSCB-BPNN achieves good performance when the number of epochs $G$ is greater than 80, and the value of $N_{pop}$ is greater than 35.

Figure 5 shows the comparison graph of MSCB-BPNN with the other three models regarding the change of misclassifications with an increasing number of training generations. In this paper, BPNN is a three-layer structure consisting of an input layer, an implicit layer, and an output layer. These four models are identical in other parameters, such as network structure, except for the different optimization algorithms used. The four models, BPNN, SA-BPNN, GA-BPNN, and MSCB-BPNN, were trained for 100 rounds, and the ratio of the number of misclassifications to the total number of samples obtained was 8.325%, 5.238%, 3.525%, and 0.608%, respectively. It can be seen that the BPNN optimized by the optimization algorithm is better than the unoptimized BPNN, and most importantly, among the three models that all use the optimization algorithm, the BPNN optimized by MSCB is significantly better than the other two algorithms in terms of reducing the number of misclassifications.

Figure 6 demonstrates the accuracy comparison between training data and test data optimized by MSCB-BPNN, GA-BPNN, SA-BPNN, and BPNN. From the comparison results, the MSCB-BPNN approach proposed in this paper outperforms the other three approaches in terms of its intrusion classification accuracy both on the training and test sets. Table 7 shows the precision ratio of the four models for five types of network intrusions in classifying the test set. Table 7 makes it absolutely clear that the MSCB-BPNN proposed in this paper has a better detection effect for these five intrusions, unlike the other three models that have a better detection effect for one type of intrusion and a poorer detection effect for another kind of intrusion.

Table 8 shows the recall ratio of the four models for the five network intrusions during the test set classification. The recall ratio is the fraction of the relevant documents that are
successfully retrieved. In the experiments, a higher recall ratio indicates better performance of the approach. From Table 8, we can see that the recall ratio of MSCB-BPNN is higher than the other three models for four of the five network intrusion models.

In Figure 7, comparing the MSCB-BPNN with the other three techniques, the graph reveals that it has a substantially better recall ratio and average precision ratio, which means that MSCB-BPNN is more suitable for network intrusion detection than the other three methods.

In Figure 8, the ROC curves are used to reflect the classification performance of the four methods for the KDDCup99 dataset under all classification thresholds. In this paper, we make the samples with the attack category

**Table 7:** The classification precision ratio of the four methods on test sets.

| Precision ratio (%) | BA-BPNN | GA-BPNN | SA-BPNN | BPNN   |
|---------------------|---------|---------|---------|--------|
| Normal              | 99.00   | 96.45   | 95.07   | 95.98  |
| Dos                 | 100.00  | 99.78   | 100.00  | 100.00 |
| Probing             | 98.36   | 100.00  | 66.66   | 1.56   |
| R2L                 | 98.90   | 99.63   | 90.50   | 86.50  |
| U2R                 | 100.00  | 68.18   | 85.71   | 94.73  |

**Table 8:** The classification recall ratio of the four methods on test sets.

| Recall ratio (%) | BA-BPNN | GA-BPNN | SA-BPNN | BPNN   |
|-----------------|---------|---------|---------|--------|
| Normal          | 100.00  | 99.85   | 97.33   | 100.00 |
| Dos             | 99.25   | 97.50   | 97.09   | 92.26  |
| Probing         | 95.71   | 93.54   | 52.05   | 100.00 |
| R2L             | 99.63   | 98.19   | 98.88   | 95.05  |
| U2R             | 100.00  | 93.75   | 100.00  | 31.03  |
Normal as positive samples and other attack categories as negative samples. Then, ROC curves for each of the four methods are plotted. The obtained ROC curves show that the curve closer to the upper left corner has better classification performance. So, it is clear that the proposed MSCB-BPNN has better classification performance than the other three methods.

To further verify the classification performance of the proposed algorithm, we conducted another simulation experiment on the UNSW-NB15 dataset. During the validation process, we downscaled the UNSW-NB15 dataset to improve the training speed, selected 26 critical features as the input data, and sampled the training and test sets in a 25% stratified manner according to the attack types.
Figure 9 reflects the classification results of MSCB-BPNN, GA-BPNN, SA-BPNN, and BPNN on the UNSW-NB15 dataset. It can be seen that even if the dataset used for validation is changed to UNSW-NB15, the proposed MSCB-BPNN still has advantages over the other three methods in terms of accuracy, average precision rate, and average recall rate.

5. Conclusions

To solve the shortcomings of conventional intrusion detection approaches, such as low detection efficiency and a high rate of false alarms, a new intrusion detection approach called MSCB-BPNN is proposed in this paper, which uses the proposed MSCB algorithm to help BPNN for the selection of initial weights and thresholds. From the experimental results, the proposed algorithm performs better than BPNN, GA-BPNN, and SA-BPNN for network intrusion detection on the KDD cup 99 dataset and the UNSW-NB15 dataset, and MSCB-BPNN achieves 99.4% and 99.8% accuracy on the Kddcup99 training and test sets, respectively, while 89.0% and 93.9% accuracy on the UNSW-NB15 training and test sets, respectively. Evidently, the proposed method can effectively handle the network intrusion detection problem.

For future research directions, an extension we found interesting is the use of XAI tools to interpret the behavior of DL-based traffic identification methods. As far as we know, Explainable Artificial Intelligence (XAI) proposes to move towards a more transparent AI. It aims to create a set of techniques to produce more interpretable models while maintaining high-performance levels [33, 34]. So, we think much work needs to be done in that research direction.

Data Availability

The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This paper was funded by the Corps Social Science Fund Project "Research on the Accuracy of Network Ideological and Political Education in Corps Colleges and Universities", under grant no. 20YB22, the Corps Innovative Talents Plan, under grant no. 2020CB001, the project of Youth and Middle-Aged Scientific and Techno-logical Innovation Leading Talents Program of the Corps, under grant no. 2018CB006, the China Postdoctoral Science Foundation, under grant no. 220531, Funding Project for High Level Talents Research in Shihezi University, under grant no. RCZK2018C38, Project of Shihezi University, under grant
References

[1] H. A. Ahmed, A. Hameed, and N. Z. Bawany, “Network intrusion detection using oversampling technique and machine learning algorithms,” Peerf Computer Science, vol. 8, p. e820, 2022.

[2] M. Al-Qatif, 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, Article ID 52843, 2018.

[3] A. Khraisat and A. Alazab, “A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges,” Cybersecurity, vol. 4, no. 1, pp. 18–27, 2021.

[4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poonarchandran, A. Al-Nemrat, and S. Venkatraman, “Deep learning approach for intelligent intrusion detection system,” IEEE Access, vol. 7, Article ID 41525, 2019.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” Cybersecurity, vol. 2, no. 1, pp. 20–22, 2019.

[6] V. Hajisalem and S. Babaie, “A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection,” Computer Networks, vol. 136, pp. 37–50, 2018.

[7] S. Ray, “A quick review of machine learning algorithms,” in Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-ITCon), pp. 35–39, Faridabad, India, February 2019.

[8] S. Tamy, H. Belhadaoui, N. Rabbah, and M. Rifi, “Cyber security based machine learning algorithms applied to industry 4.0 application case: development of network intrusion detection system using hybrid method,” Journal of Theoretical and Applied Information Technology, vol. 98, no. 12, pp. 2078–2091, 2020.

[9] O. A. Alimi, K. Ouahada, A. M. Abu-Mahfouz, S. Rimer, and K. O. A. Alimi, “Intrusion Detection for Water Distribution Systems Based on an Hybrid Particle Swarm Optimization with Back Propagation Neural Network,” in Proceedings of the 2021 IEEE AFRICON, pp. 1–5, Arusha, Tanzania, September 2021.

[10] S. D. Tangi, P. Kumar, and M. S. Bewoor, “A novel mechanism for development of intrusion detection system with BPNN,” in Proceedings of the 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), pp. 1–7, Dehradun, India, November 2021.

[11] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, “A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding,” Computers & Security, vol. 84, pp. 376–392, 2019.

[12] N. Chen, C. Xiong, W. Du, C. Wang, X. Lin, and Z. Chen, “An improved genetic algorithm coupling a back-propagation neural network model (IGA-BPNN) for water-level predictions,” Water, vol. 11, no. 9, p. 1795, 2019.

[13] G. Qi, J. Zhou, W. Jia, M. Liu, S. Zhang, and M. Xu, “Intrusion detection for network based on elite clone artificial bee colony and back propagation neural network,” Wireless Communications and Mobile Computing, vol. 2021, Article ID 9956371, 11 pages, 2021.

[14] B. G. Atlı, Y. Miche, A. Kalliola, I. Oliver, S. Holtmanns, and A. Lendasse, “Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space,” Cogn. Comput., vol. 10, no. 5, pp. 848–863, 2018.

[15] A. Ahmadian Ramaki, A. Rasooлизadeghan, and A. Javan Jafari, “A systematic review on intrusion detection based on the Hidden Markov Model,” Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 11, no. 3, pp. 111–134, 2018.

[16] G. Vigna and R. A. Kemmerer, “NetSTAT: a network-based intrusion detection approach,” in Proceedings of the 1998 14th Annual Computer Security Applications Conference (Cat. No. 98EX217), pp. 25–34, Phoenix, AZ, USA, December 1998.

[17] Z. S. Malek, B. Trivedi, and A. Shah, “User behavior pattern-signature based intrusion detection,” in Proceedings of the 2020 4th World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 549–552, London, UK, July 2020.

[18] Y. Mirsky, T. Doithshan, Y. Elovici, and A. Shbatai, “Kitsume: An Ensemble of Autoencoders for Online Network Intrusion Detection,” 2018, https://arxiv.org/abs/1802.09089.

[19] G. Bovenzi, G. Aceto, D. Ciouzono, V. Persico, and A. Pescapè, “A hierarchical hybrid intrusion detection approach in IoT scenarios,” in Proceedings of the GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–7, Taipei, Taiwan, December 2020.

[20] K. Akomea-Agyin and M. Asante, “Analysis of security vulnerabilities in wired equivalent privacy (WEP),” Int. Res. J. Engineering Technology, vol. 6, no. 1, pp. 529–536, 2019.

[21] M. Roopak, G. Y. Tian, and J. Chambers, “An intrusion detection system against ddos attacks in iot networks,” in Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0562–0567, Las Vegas, NV, USA, January 2020.

[22] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, “Anomaly detection via online oversampling principal component analysis,” IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 7, pp. 1460–1470, 2013.

[23] Q. Tian, D. Han, K.-C. Li, X. Liu, L. Duan, and A. Castiglione, “An intrusion detection approach based on improved deep belief network,” Applied Intelligence, vol. 50, no. 10, pp. 3162–3178, 2020.

[24] D. R. Prado, J. A. Lopez-Fernandez, G. Barquero, M. Arrebola, and F. Las-Heras, “Fast and accurate modeling of dual-polarized reflectarray unit cells using support vector machines,” IEEE Transactions on Antennas and Propagation, vol. 66, no. 3, pp. 1258–1270, 2018.

[25] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, “An intrusion detection model based on feature reduction and convolutional neural networks,” IEEE Access, vol. 7, Article ID 42210, 2019.

[26] Q. Shi, J. Kang, R. Wang, H. Yi, Y. Lin, and J. Wang, “A framework of intrusion detection system based on Bayesian network in IoT,” International Journal of Performability Engineering, vol. 14, no. 10, p. 2280, 2018.

[27] X. S. Gan, J. S. Duanmu, J. F. Wang, and W. Cong, “Anomaly detection based on PLS feature extraction and core vector machine,” Knowledge-Based Systems, vol. 40, pp. 1–6, 2013.

[28] L. Wang, B. Wu, Q. Zhu, and Y.-R. Zeng, “Forecasting monthly tourism demand using enhanced backpropagation neural network,” Neurocomputing, vol. 252, no. 3, pp. 2607–2636, 2020.

[29] W. Jiao and C. Ma, “Track prediction algorithm based on GA-BPNN,” in Proceedings of the 2021 IEEE 3rd International Conference on Civil Aviation Safety and Information
[30] F. Zhang, Y. M. Fadul Mukhtar, B. Liu, and J. Li, "Application of ANN to predict the apparent viscosity of waxy crude oil," *Fuel*, vol. 254, Article ID 115669, 2019.

[31] Z. Jia, "Prediction of college students’ psychological crisis with a neural network optimized by harmony search algorithm," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 17, no. 02, pp. 59–75, 2022.

[32] Y. Zhu, G. Zhang, and J. Qiu, "Network traffic prediction based on particle swarm BP neural network," *Journal of Networks*, vol. 8, no. 11, pp. 2685–2691, 2013.

[33] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, Article ID 52138, 2018.

[34] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "XAI meets mobile traffic classification: understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.