Hindawi
Scientific Programming
Volume 2023, Article ID 2989533, 18 pages
https://doi.org/10.1155/2023/2989533

Research Article

Network Traffic Anomaly Detection Model Based on Feature Reduction and Bidirectional LSTM Neural Network Optimization

Hanqing Jiang,[1,2] **Shaopei Ji**,[2] **Guanghui He**,[3] **and Xiaohu Li**[4]

[1]*China Satellite Network Application Co., Ltd., Beijing 100001, China*
[2]*The 30th Research Institute of China Electronics Technology Group Corporation, Chengdu, Sichuan 610041, China*
[3]*Jiangnan Institute of Computing Technology, Wuxi, Jiangsu 214000, China*
[4]*Institute of Data Communication Science and Technology, Beijing 100001, China*

Correspondence should be addressed to Shaopei Ji; 435323646@qq.com

Received 30 January 2023; Revised 6 September 2023; Accepted 5 October 2023; Published 3 November 2023

Academic Editor: Dongpo Xu

Aiming at the problems of large data dimension, more redundant data, and low accuracy in network traffic anomaly detection, a network traffic anomaly detection model (FR-APPSO BiLSTM) based on feature reduction and bidirectional long short-term memory (LSTM) neural network optimization is proposed. First, the feature dimensions are divided by hierarchical clustering according to the similarity distance between data features, and the features with high correlation are divided into the same feature subset. Second, an automatic encoder is used to reduce each feature subset, eliminating redundant information, and reducing the computational complexity of the detection data. Then, a particle swarm optimization algorithm based on adaptive updating of variables and dynamic adjustment of parameters (APPSO) is proposed, which is used to optimize the parameters of the bidirectional LSTM neural network (BiLSTM). Finally, the optimized BiLSTM is used as a classifier to model network traffic anomaly detection using the reduced feature data. Experiments based on NSL-KDD, UNSW-NB15, and CICIDS-2017 datasets show that the proposed FR-APPSO-BiLSTM model can effectively reduce data features, improve the accuracy of detection, and the performance of network traffic anomaly detection.

1. Introduction

The development and breakthrough of network communication technology bring the convenience of the big data era and interconnection of 5G technology. The scale of traffic data in network communication is larger and larger. While enjoying the convenience of network communication, network attacks in network communication security are becoming more and more serious. Finding abnormal traffic through network traffic anomaly detection has become an urgent problem to be solved. Abnormal network traffic refers to the trend that current network traffic behavior deviates from normal network traffic. Network traffic anomalies are mainly caused by malicious network attacks, such as denial of service attacks, port scanning, password blowout, remote control, and so forth, as well as network misconfiguration and other exceptions [1].

After decades of development, the current network traffic anomaly detection methods can be divided into feature-based detection methods and anomaly based detection methods based on different detection methods [2]. The method based on feature detection [3] first analyzes various attack modes, extracts attack features, and then adds them to the feature library for detecting new attacks. When the detection sample matches the information in the feature library, it can be determined as an attack behavior. This type of method has a low false alarm rate, but it has a certain lag and can only detect existing attack patterns in the feature library, resulting in a low detection rate for new types of attacks. The method based on anomaly detection [4] establishes a probability statistical model to generalize the patterns of normal information flow samples and generate a benchmark model. When the patterns of the detected samples do not match the benchmark model, it can be considered an anomaly attack. This type of method has a low false alarm rate and certain

detection ability for new attacks, but it also has the problem of insufficient detection accuracy.

In response to the current problems faced in anomaly detection, many researchers have proposed various technologies to improve the accuracy and stability of detection methods, most of which improve detection accuracy by reducing false positives and detecting unknown attacks [5]. With the emergence and development of machine learning and deep learning, the task of manually defining features can be replaced by a trainable multilayer network, which can achieve higher accuracy and lower false alarm rate in intrusion detection tasks than traditional machine learning. Therefore, various methods are widely used in the field of network traffic anomaly detection methods.

1.1. Anomaly Detection Based on Machine Learning Methods. Blanco et al. [6] presented a hybrid dimensionality reduction technique combining information gain and principal component analysis techniques. The experimental results showed that the hybrid dimensionality reduction method is better than the single algorithm and can effectively identify abnormal traffic. Phanindra et al. [7] used recursive feature elimination techniques to rank features according to their importance and used stochastic forest algorithms to classify attacks. Su et al. [8] studied the characteristics of abnormal behavior of network traffic, used hierarchical clustering method to sample traffic data, and then used support vector machine (SVM) to detect anomalies. Andresini et al. [9] combined an automatic encoder with a triplet network to solve the problem of convergence during triplet learning. Onah et al. [10] introduced a genetic algorithm-based wrapper and Bayesian anomaly detection model (GANBADM) for fog environments, which eliminated extraneous attributes to reduce time complexity and enable more accurate detection. Zhang et al. [11] proposed an anomaly detection model (MFFSEM) based on multidimensional feature fusion and stack integration mechanism, and established multiple basic feature datasets considering time, space, load, and other aspects of traffic information. And established multiple comprehensive feature datasets considering the association and relevance among the basic feature datasets to meet the requirements of real-world anomaly detection. Liu et al. [12] combining the ELM with a selective ensemble algorithm based on edge distance minimization has shown good detection performance and reduced false positives in the KDD99 dataset. Alzahrani and Alenazi [13] combining data reduction with stochastic forest, adaboost, and other machine learning algorithms proved the effectiveness of data reduction in anomaly flow detection, and improved detection efficiency with less data and computation. Aiming at the problems of weak generalization ability and poor learning ability of ELM model with single kernel function, Jinghao et al. [14] constructed a hybrid kernel ELM model (HKELM) and optimized the parameters of HKELM model by combining GSA and DE, so as to improve its global and local optimization ability in anomaly detection.

The common machine learning algorithms include logistic regression, *K*-means, SVMs, naive Bayes, *K*-nearest neighbors, random forests, and so forth. Although these machine learning algorithms have made great progress in network traffic anomaly detection, there are still some problems such as difficult detection and long detection time. At the same time, the traditional feature selection method can keep the effective information of the original data and reduce the computational complexity of the classifier [15].

1.2. Abnormality Detection Based on Deep Learning. Lin et al. [5] proposed a method for network intrusion detection based on a convolutional neural network. The accuracy of this method increases with the number of training samples. Almiani et al. [16] proposed an automatic intrusion detection system based on multilayer recurrent neural networks to resist network attacks faced by fog computing. Zhang et al. [17] proposed an intrusion detection method based on a deep convolutional neural network. The biggest feature of this method is that it converts 1D intrusion data into 2D "image data" for training networks, effectively improving the detection accuracy of the intrusion detection system. Although accuracy and false alarm rate have always been the focus of NIDS research, real-time performance, and detection efficiency are also important indicators. In deep learning, compared with other neural networks, autoencoder (AE) can effectively reduce the feature dimension and is easy to combine with other models. It can greatly reduce the training time of the model while improving detection accuracy. As an unsupervised learning artificial neural network, the AE consists of an encoder function that maps the input to a hidden layer and a decoder function. The learned reconstruction input is generated by minimizing the loss function. Alqatf et al. [18] proposed a deep learning framework based on a stacked AE for feature learning of normal samples and then used a SVM classifier to improve the accuracy of the method. Andresen et al. [19] used deep neural networks to train the characteristics of input data, and then used AE as an anomaly detector to refine classification, thus improving the classification accuracy for unforeseen attacks. Mirza and Cosan [20] proposed a deep learning network intrusion detection method based on a mixture of sparse AE and long short-term memory (LSTM) networks. This method uses AE to reduce the dimension of data and extract features and then uses LSTM networks to deal with the sequential nature of computer network data so that it can effectively deal with unpredictable and unpredictable network attacks.

Although the above-depth methods can achieve certain results, these methods inevitably lose information when using AE to compress the original data, and these studies only train a single AE from normal samples, without considering attack samples, so the detection rate of new attack samples is low.

To solve the above problems, this paper proposes a network abnormal traffic detection method based on feature reduction (FR) and bidirectional LSTM (BiLSTM) neural network. Feature selection and fusion are carried out through hierarchical clustering and automatic encoder to achieve the goal of feature dimensionality reduction; then an improved particle swarm optimization (PSO) algorithm is proposed to optimize the parameters of the BiLSTM neural network;

finally, anomaly detection is performed based on the optimized BiLSTM.

The main contributions of this paper are as follows:

(1) A FR algorithm based on hierarchical clustering and automatic encoder is proposed.
(2) A PSO algorithm based on adaptive updating of variables and dynamic adjustment of parameters is proposed.
(3) A network traffic anomaly detection model based on FR and BiLSTM neural network optimization is proposed.

The rest of the paper is organized as follows: Section 2 presents the correlation theory on abnormal detection. Section 3 presents a PSO algorithm based on variable adaptive update and parameter dynamic adjustment. Section 4 presents the design and development phases of the proposed model. Section 5 presents the experimental results and discussion. The last section concludes the paper with future direction.

2. Feature Reduction Algorithm Based on Hierarchical Clustering and Automatic Encoder

As an unsupervised clustering algorithm, hierarchical clustering is used to classify network features at different levels to form tree-like clustering results [21]. The clustering algorithm can perform hierarchical relationship mining of data without predetermining the number of clusters. The AE is a kind of artificial neural network model that can reduce the feature dimension and extract the nonlinear information in the data by training the neural network for input reconstruction [22].

In order to reduce the impact of data size on the detection model while retaining the effective information of network traffic data, this section proposes a FR algorithm based on hierarchical clustering and AE. The algorithm implementation process is as follows.

(1) Calculate the similarity distance between the feature vectors and construct the feature similarity matrix R.
(2) Taking the n features as n classes, and the classes are unrelated.
(3) Concatenate the two closest classes as a new class.
(4) Calculate the similarity distance between the new class and the current classes. If the number of clusters is equal to n at this time, go to Step 5, otherwise continue to Step 3.
(5) Output the clustering results to the AE.
(6) Construct k classes according to the similarity of traffic features. The feature subclass is $V = \{V_1, V_2, ..., V_K\}$, and the AE is $L = \{L_1, L_2, ..., L_K\}$. Create an AE L_i for each feature subclass V_i to learn the normal and abnormal behavior of the corresponding feature subclass.
(7) The weights of the AE are initialized according to a uniform distribution $\chi\left(\frac{-1}{\dim(V_i)}, \frac{-1}{\dim(V_i)}\right)$.
(8) Input the data x into the nonlinear activation function f in the AE V_i to encode, and get the encoded data y of the hidden layer.

$$y = f(W_1 x + b_1). \tag{1}$$

(9) The data y is used as the input of the nonlinear activation function g in the decoder, and the decoded reconstructed data $\overline{x}$ is obtained.

$$\overline{x} = g(W_2 y + b_2). \tag{2}$$

(10) In the training stage, the parameters θ are optimized by random gradient descent algorithm so that the error of decoded reconstruction data is reduced as much as possible. At the running time, skip Step 10 and go to Step 11.

$$L_{\text{loss}} = \|x - \overline{x}\|^2. \tag{3}$$

(11) Using AE L_i to calculate the anomaly scores S_i of the corresponding feature subclasses V_i.
(12) Output the anomaly score S, $S = \{S_1, S_2, ..., S_K\}$.

The anomaly score S will be used as the input data of the classifier for anomaly detection.

3. Particle Swarm Optimization Algorithm Based on Variable Adaptive Update and Parameter Dynamic Adjustment

For large-scale high-dimensional optimization problems, the mechanism of the original particle swarm algorithm and other swarm intelligence algorithms makes the algorithm itself unable to expand the search space. Once it falls into a local optimum, it is difficult to jump out of the high-dimensional local optimum, so the convergence is early and the convergence accuracy is extremely high. Moreover, the current improved particle swarm algorithm generally strengthens one of the local search abilities or global search abilities and even abandons the ability of global search or local development to pursue convergence speed or convergence accuracy [23]. Therefore, some scholars start from the balance to take into account the global search and local development capabilities of the PSO algorithm, which not only expands the particle search space, but does not reduce the accuracy of the algorithm convergence, but from the current results, it is considered to strengthen the local and global search capabilities. It is also easy to lose control of the balanced searchability, and eventually the algorithm still only strengthens the part of the ability in essence, and it is also easy to fall into local optimum in solving high-dimensional

problems, failing to give full play to the characteristics of PSO.

In order to solve these problems, APPSO is proposed, which is based on variable adaptive update and parameter dynamic adjustment. In APPSO, the adaptively updated variables include particle velocity and position, and the dynamically adjusted parameters include inertial weights and learning factors.

3.1. Adaptive Update of Particle Velocity. The APPSO algorithm uses the adaptive velocity update Equation (4) to control the influence of the individual optimal position and the global optimal position on the particle velocity. To achieve the effect of joint optimization of global search and local development, the influence of the individual optimal position on the particle velocity is greater in the early stage, and the influence of the global optimal position on the particle velocity in the later stage gradually increases.

$$V_{id}^{t+1} = \omega V_{id}^{t} + c_1(t) r_1 \left(p\text{best}_{id}^{t} - \text{rand}\left(\frac{1}{2} - \frac{t}{T}\right) x_{id}^{t} \right) + c_2(t) r_2 \left(g\text{best}_{d}^{t} - \text{rand}\left(\frac{t}{T} - \frac{1}{2}\right) x_{id}^{t} \right), \tag{4}$$

where V_{id}^{t+1} is the particle velocity after the update, w is the inertia weight, $p\text{best}_{id}^{t}$ is the individual optimal position of particle i, $g\text{best}_{d}^{t}$ is the global optimal position of the population P, x_{id}^{t} is the particle position, t is the number of current iterations, and T is the maximum number of iterations.

The essence of Equation (4) is to control the degree of influence of individual learning ability and social learning ability on particle velocity by the ratio of the current iteration number and the maximum iteration number. In the early stage of the algorithm, when the number of iterations t/N is less than 1/2, the influence of the particle's individual learning ability on the particle speed is smaller than the particle's social learning ability. When the current iteration number t/N is greater than 1/2, the opposite is true. Equation (13) adaptively controls the influence of individual optimal and global optimal on particle velocity update.

3.2. Adaptive Update of Particle Position. The position adaptive update in the APPSO algorithm mainly adopts two update methods: Cauchy update and normal update, which evolved from the Cauchy distribution and the normal distribution. The distribution diagrams of the Cauchy distribution and the normal distribution are shown in Figure 1 [24, 25].

As can be seen from Figure 1, the Cauchy distribution has a wider range of values on the x-axis than the normal distribution, which means that the solution space range of the particle search updated based on the Cauchy distribution is larger. It can be seen that more and better feasible solutions can be searched based on the Cauchy update particle, which is very suitable for the early global search work.

The normal update has a deeper value range on the y-axis, which means that the particles updated based on the normal distribution have a deeper optimal solution value range, and can improve the convergence accuracy of the algorithm. It can be seen that the normal update is very suitable for the later local development work. The Cauchy distribution and the normal distribution also have a significant effect on dealing with high-dimensional problems.

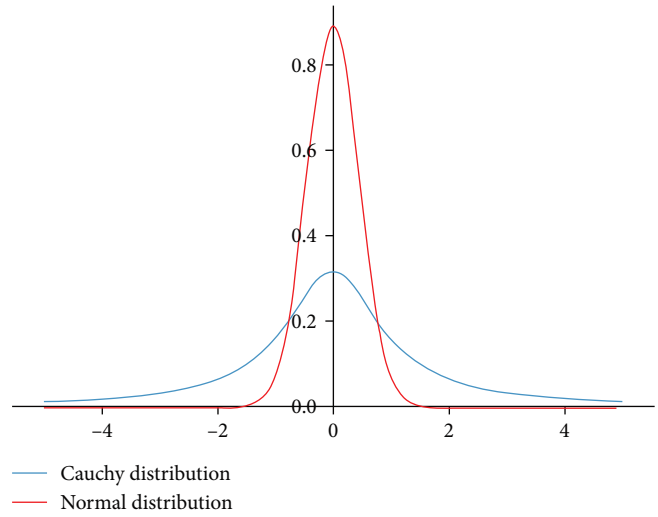

Figure 1: Cauchy distribution and normal distribution.

When the particle is updated based on the f Equation (4), the particle is already in the state of adaptive speed update, and then the particle position is updated adaptively. Equations (5) and (6) are the particle position update equals based on the Cauchy distribution and normal distribution.

$$X_{id}^{t+1} = X_{id}^{t} + \text{Cauchy} * \left(p\text{best}_{id}^{t} - x_{id}^{t} \right), \tag{5}$$

$$X_{id}^{t+1} = X_{id}^{t} + \text{Normal} * \left(p\text{best}_{id}^{t} - x_{id}^{t} \right), \tag{6}$$

where X_{id}^{t} is the position of the particle i in the tth iteration, X_{id}^{t+1} is the position of the particle i in the $t+1$th iteration, $p\text{best}_{id}^{t}$ is the local optimal position of the particle i in the tth iteration, Cauchy is an update operator based on the Cauchy distribution, and Normal is an update operator based on the normal distribution.

The update method based on Cauchy distribution is suitable for early global search, and the update method based on normal distribution is suitable for later local development. According to the number of algorithm iterations, the particle position is adaptively updated. In Equations (5) and (6), the particle uses the Cauchy distribution and normal distribution to update and optimize its own position information to find the most suitable way. This optimization method can combine the local and global optimum influence degree in Equation (4), and make the global optimum influence degree maximized in the early stage, so that the algorithm can expand the search scope well in the early stage, find more and better feasible solutions, and maximize the individual optimum influence degree in the later stage.

3.3. Dynamic Adjustment of the Parameters

3.3.1. Dynamic Adjustment of the Inertia Weight. The diversity of the population is usually considered an important reason for premature convergence of swarm intelligence optimization algorithms [26]. It is well known that inertia weights play a vital role in balancing global exploration and local mining capability in PSO. Therefore, how to introduce the diversity of population into inertia weights and enhance the balance of global exploration and local mining ability of PSO is a meaningful thing. High diversity means that the population searches in a larger space, whereas low diversity means that the population enters a smaller search space.

This section introduces an inertial weight based on population diversity, as defined below:

$$\omega(t) = e^{-S(t)} \times \left(1 - \frac{t}{T_{\max}}\right), \tag{7}$$

where $T_{\max}$ is the maximum number of iterations and $S(t)$ is the population diversity. The definition of $S(t)$ is as follows:

$$S(t) = \frac{t}{M} \times \sum_{i=1}^{M} \sqrt{\sum_{s=1}^{\text{Sim}} (x_i^s(t) - \overline{x}^s(t))^2}, \tag{8}$$

where the size of the population, Sim represents the dimension of the problem to be optimized, $x_{id}(t)$ is the position of the particle i, and $\overline{x}^s(t)$ is the average position of the entire population. The definition of $\overline{x}^s(t)$ is as follows:

$$\begin{aligned} \overline{x}(t) &= \text{mean}\{x_{1d}(t), x_{2d}(t), \ldots, x_{Md}(t)\} \\ &= \left(\frac{1}{M}\sum_{i=1}^{M} x_{id}^1(t), \frac{1}{M}\sum_{i=1}^{M} x_{id}^2(t), \ldots, \frac{1}{M}\sum_{i=1}^{M} x_{id}^{\text{sim}}(t)\right). \end{aligned} \tag{9}$$

It can be seen from Equation (8) that the value of population diversity $S(t)$ is determined by the distance between each particle in the population and the average position of the entire population. In the initial stage of the algorithm operation, the value of $S(t)$ has a great influence on the inertia weight $\omega(t)$. When the algorithm converges towards a certain point, the population diversity increases rapidly, which makes the entire population approach the optimal point in a relatively stable form. Such search results will reduce the diversity of the population in subsequent iterations, which causes the value of $\omega(t)$ to drop, and leads the entire population to finer mining in a smaller flying space.

3.3.2. Dynamic Adjustment of the Learning Factors. In the APPSO, the self-cognitive learning factor C_1 and the social learning factor C_2 are used to control the magnitude of each related learning item. Generally, the value of both is 2.5. In order to better control the amplitude, this paper adopts a time-varying strategy to dynamically adjust the learning factor, in which the self-cognitive learning factor C_1 decreases from 2.5 to close to 0.5, and the social learning factor c_2 increases from 0.5 to close to 2.5. The self-cognitive learning factor c_1 and the social learning factor c_2 are calculated as follows:

$$c_1(t) = 2.5 - 2 \times \ln\left(e^{\frac{t-1}{T_{\max}}} - 1\right), \tag{10}$$

$$c_2(t) = 0.5 + 2 \times \ln\left(e^{\frac{t-1}{T_{\max}}} - 1\right). \tag{11}$$

3.4. Implementation Process of the APPSO. The main idea of the APPSO is to divide the entire algorithm optimization process into two stages: the early stage and the later stage. In the early stage, in order to expand the global search ability, the Cauchy update was performed during the iterative update of particles to increase the search diversity and expand the global search range. In the later stage, in order to expand the local development capability, the normal update was performed during the iterative update of particles to improve the convergence accuracy of the algorithm. During the entire search process, the algorithm dynamically adjusts the inertia weight and learning factor to balance the global exploration and local mining capabilities according to the diversity of the population.

The implementation process of the APPSO algorithm is shown in Table 1.

4. Network Traffic Anomaly Detection Model Based on Feature Reduction and Bidirectional LSTM Neural Network Optimization

In order to obtain better detection accuracy and improve detection speed, this section proposes a network abnormal traffic detection model based on FR and BiLSTM neural network optimization. First, the hierarchical clustering algorithm and AE are used to select and reduce data features, which allows to retain valid information while reducing the amount of data. Then, the reduced data is sent to the classifier for training. Finally, an anomaly detection model is constructed to identify whether the newly incoming traffic data is abnormal data.

4.1. Data Preprocessing. There are three types of data in a dataset. The preprocessing of data is to transform the character data in the dataset into unique code before the model training, unify the data type into the recognized type of the model, and normalize the data to the number between 0 and 1. The calculation equal of data preprocessing is as follows:

$$y = \frac{x - \text{MIN}}{\text{MAX} - \text{MIN}}, \tag{12}$$

where y represents the preprocessed data value, x represents the original data value, MAX represents the maximum value in the dataset, and MIN represents the minimum value in the dataset.

Table 1: The APPSO algorithm.

Initialize the population parameters: population size: I, maximum number of iterations: T, particle position domain $[X_{\min}, X_{\max}]$, particle velocity domain $[V_{\min}, V_{\max}]$, particle dimension: $d = 1, 2, 3\ldots, D$.

Initialize x: $X_i^1 = X_{\min} + \text{rand} \cdot (X_{\max} - X_{\min})$
Initialize v: $V_i^1 = V_{\min} + \text{rand} \cdot (V_{\max} - V_{\min})$
for $i = 1, 2, 3\ldots, T$ do
  for $i = 1, 2, 3\ldots, I$ do
    Calculate the fitness: fitness($p$best$_i^t$), fitness(gbest^t)
    update ω with Equation (16)
    update c_1 and c_2 with Equations (19) and (20)
    if $\frac{t}{T} \leq \frac{1}{2}$ then
      update V with Equation (13)
      update X with Equation (14)
    if $\frac{t}{T} > \frac{1}{2}$ then
      update V with Equation (13)
      update X with Equation (15)
    Calculate the fitness: fitness(X_i^t)
    if fitness$_i^t$ < fitness($p$best$_i^t$) then
      $p$best$_i^t = X_i^t$
    fitness($p$best$_i^t$) = fitness(X_i^t)
    end if
    if fitness($p$best$_i^t$) < fitness($g$best^t) then
      $p$best$_i^t = g$best$^t = X_i^t$
    fitness($g$best^t) = fitness($p$best$_i^t$)
    end if
  end for
end for

4.2. The Process of Optimizing BiLSTM Based on APPSO. Jian et al. [27] put forward the variant recurrent neural network short-term memory network, which introduced a gating mechanism to simply and effectively solve the problem of gradient explosion or disappearance of the traditional recurrent neural network. The LSTM controls the information transmission between each cell through the gating mechanism. The LSTM is a one-way extraction of sequence information, but for the network traffic anomaly detection problem, the current network situation is not only related to the previous situation, but also may be related to the subsequent situation. In order to improve the prediction effect, BiLSTM is introduced to detect network traffic anomalies.

The BiLSTM model used for the experiments in this paper is shown in Figure 2.

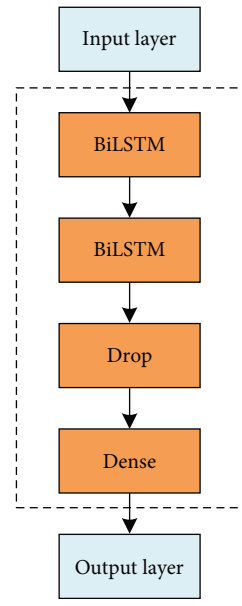

Figure 2: BiLSTM model.

(1) BiLSTM layers: with two BiLSTM layers, their combined before-and-after capabilities can be fully exploited to enhance model learning.
(2) Dropout layer: avoid overfitting of the model and improve generalization.
(3) Dense layer: set the last layer as Dense, transform the output dimension, and get the prediction result.

For the BiLSTM network, the selection of parameters in its structure is critical to the effect of the model, such as the number of hidden layers, weight, number of hidden layer units, and learning rate. Many researchers determine these parameters based on experience or trial and error method, which makes the robustness and accuracy of the model unreliable. Therefore, this paper selects the PSO algorithm which is simple in principle, low in complexity, fast in convergence, and suitable for dealing with real value problems to optimize the structural parameters of the BiLSTM network.

The traditional cross-validation method is a commonly used parameter optimization method, but it takes a long time and the parameter selection is blind. At present, many parameter optimization methods have been studied and

applied [28]. PSO has been widely used in optimization, evolutionary computing, and other fields due to its outstanding algorithm performance. Combining the APPSO proposed in this paper, it optimizes the parameters of BiLSTM to give better performance. The specific implementation steps for optimizing BiLSTM parameters based on APPSO are as follows:

(1) According to the size of the sliding window, the training set samples, and test set samples are constructed.
(2) Initializing the relevant parameters in APPSO, which include the maximum and minimum values of the search dimension D, the number of particles P_N, the acceleration factors c_1 and c_2, the maximum number of iterations max_iter, the initial position of the particles x_i^0 and the initial velocity v_i^0, the inertia weight factor ω, and the learning factors r_1 and r_2.
(3) Setting the range of values for each dimension in the particles to be optimized. The particle dimensions α, iterator, n_1, n_2, and s represent the learning rate, the number of model iterations, the number of cells in the first hidden layer, the number of cells in the second hidden layer of the LSTM, and the random seeds in the BiLSTM model, respectively.
(4) The model sets the fitness function of the particle swarm algorithm, randomly generates the initial positions of the particle swarm, calculates the initial fitness value of each particle, and obtains the individual optimal solution *p*best and the global optimal solution *g*best at the beginning.
(5) The model calculates the fitness value of each particle, update the individual optimal solution *p*best and the global optimal solution *g*best, and update the velocity and position of the particle.
(6) If the maximum number of iterations is reached, proceed to Step 6. Otherwise return to Step 5 and continue iterating.
(7) The optimal parameters obtained are assigned to the BiLSTM model to obtain the situation prediction results.

The process of optimizing BiLSTM based on APPSO is shown in Figure 3.

4.3. Implementation Process of the FR-APPSO-BiLSTM Model. The FR-APPSO-BiLSTM model combines the advantages of hierarchical clustering algorithm, adaptive encoder, APPSO, and BiLSTM, which can effectively overcome the problems of low precision and insufficient feature extraction ability of existing intrusion detection technology. The implementation process of the FR-APPSO-BILSTM model is as follows:

(1) Preprocess the detection dataset, and divide the training dataset and the test dataset.
(2) Using the hierarchical clustering algorithm and adaptive encoder to extract features of the training dataset.

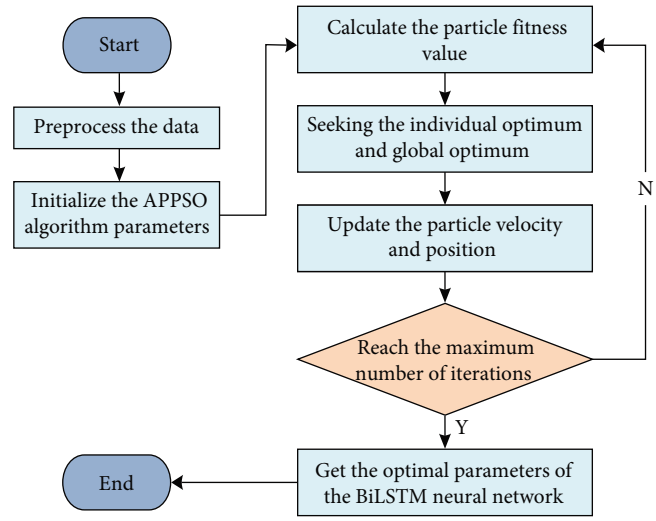

Figure 3: The process of optimizing BiLSTM based on APPSO.

(3) Using the APPSO algorithm to optimize the parameters of BiSLTM.
(4) Using the hierarchical clustering algorithm and adaptive encoder to extract features of the test dataset.
(5) The parameters obtained in Step 3 are brought into BiLSTM, and the test dataset features obtained in Step 4 are detected.
(6) Saving the obtained detection results, and the model stops running.

The flowchart of the FR-APPSO-BiLSTM model is shown in Figure 4.

5. Simulation Results Analysis

The experimental environment of this paper is Windows 10 operating system, and Keras deep learning framework is used for model training and testing in Python 3.8 environment. The hardware configuration is the 64-bit operating system, and the processor is Inter (R) Core (TM) i7 CPU 2.9 GHz.

5.1. Particle Swarm Optimization Comparison Experiment. In order to verify the performance of APPSO, this section selects PSOBSA, FOPSO, HPSO, ASPSO, OLPSO, GEPSO, and HMaPSO as the comparison algorithms for comparative analysis [29–35]. In the experiment, eight benchmark functions are used to test the performance of eight PSO algorithms. The specific information of the functions is shown in Table 2. The functions F_1–F_3 are unimodal functions and F_4–F_8 are multimodal functions. For all benchmark functions, the search dimensions are set to 5, 50, and 100, respectively. To fairly compare the performance of the eight algorithms, 50 independent runs of each test function were performed with each algorithm. The maximum number of iterations per run is set to 2,000.

5.1.1. Comparison of Optimization Results of Benchmark Functions. In this section, this paper compares the optimization of eight algorithms in eight benchmark functions. When

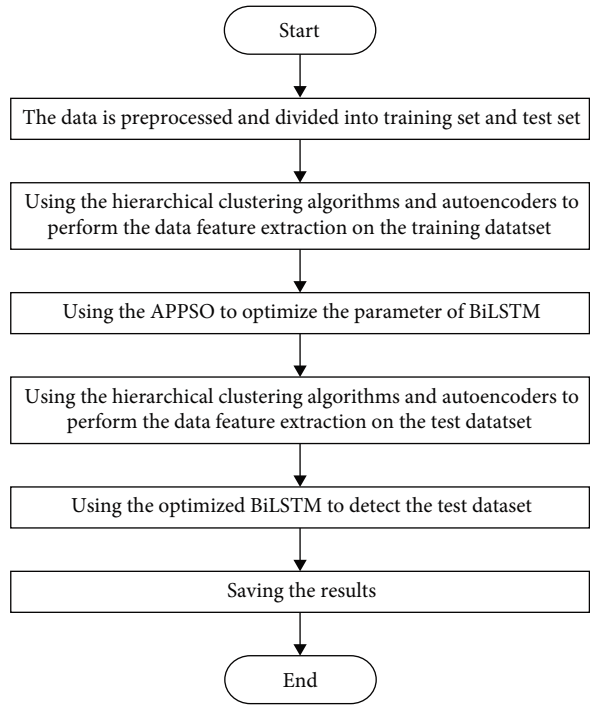

Figure 4: The flowchart of the FR-APPSO-BiLSTM model.

Table 2: Benchmark test functions.

Function name	Formulation	Range
Sphere	$F_1(X) = \sum_{i=1}^{n} X_i^2$	$[-5.12, 5.12]$
Dixon and price	$F_2(X) = (X_1 - 1)^2 + \sum_{i=2}^{n} i(2X_i^2 - X_{i-1})^2$	$[-10, 10]$
Zakharov	$F_3(X) = \sum_{i=1}^{n} X_i^2 + (\sum_{i=1}^{n} 0.5iX_i)^2 + (\sum_{i=1}^{n} 0.5iX_i)^4$	$[-5, 10]$
Rastrigin	$F_4(X) = 10n + \sum_{i=1}^{n} (X_i^2 - 1 - \cos(2\pi X_i))$	$[-5.12, 5.12]$
Levy	$F_5(X) = \sin^2(\pi(4X_i - 3)) + \sum_{i=1}^{n-1} (X_i - 1)^2[1 + \sin^2(\pi X_i + 1)] + (X_i - 1)^2[1 + \sin^2(2\pi X_i)]$	$[-3, 8.25]$
Griewank	$F_6(X) = \sum_{i=1}^{n} \frac{X_i^2}{4,000} - \prod_{i=1}^{n} \cos\left(\frac{X_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$
Rosenbrock	$F_7(X) = \sum_{i=1}^{n-1} [100(X_i^2 - X_{i+1})^2 + (X_i - 1)^2]$	$[-5, 10]$
Ackley	$F_8(X) = 20 + e - 20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} X_i^2)}) - \exp\left(\sqrt{(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi X_i))}\right)$	$[-16, 32]$

the dimensions are 5, 50, and 100, the mean of the optimization results of each algorithm is shown in Tables 3–5.

As can be seen from Tables 3 to 5, when the search space dimension is 5, the APPSO obtains the optimal solution on 6 of the 8 functions. On the functions F_1 and F_5, due to their special shapes, the APPSO is easy to fall into the local optimum. The APPSO jumps out of the local optimum by performing Cauchy update and normal update in the early and later stages according to the number of iterations, but still falls into the local optimum due to the complexity of these two functions, and the performance is not brought into play. When the search space dimension is 50, the APPSO obtains the optimal solution on 5 of the 8 functions. Since the functions F_4, F_5, and F_7 are high-dimensional complex functions, almost all algorithms fall into local optimum. However, the APPSO benefits from the adaptive update method of speed and position, and the results of these three functions are still second only to OLPSO, PSOBSA, and ASPSO, respectively. When the search space dimension is 100, the APPSO obtains the optimal solution in all eight functions. This is because, compared with other algorithms, the adaptive update of APPSO can expand the global search range in the early stage, and also has better local development in the later stage.

5.1.2. Comparison of the Convergence Speed. This section compares the convergence speed of the APPSO and other algorithms. The search space dimension is set to 50. The results are shown in Figure 5. It can be seen from that, compared with other algorithms, the APPSO can make the particle population have a very reliable performance in global search ability due to its adaptive update mechanism. It has faster convergence speed and better optimization effect

Table 3: The optimization results when the dimension is 5.

	PSOBSA	FOPSO	HPSO	ASPSO	OLPSO	GEPSO	HMaPSO	APPSO
F_1	**5.82E-14**	8.16E-05	5.34E-05	4.03E-07	1.35E-06	2.31E-04	9.82E-11	2.88E-13
F_2	2.16E+01	9.99E-02	1.41E-02	1.63E-02	1.24E+00	3.82E-03	2.20E-03	**4.91E-12**
F_3	1.53E-12	3.39E-02	2.63E-04	1.88E-06	5.27E-01	5.79E-04	2.32E-02	**2.94E-13**
F_4	1.63E-02	2.23E + 00	2.49E-02	1.96E + 00	2.07E + 00	3.99E-01	1.13E + 00	**9.76E-06**
F_5	5.51E-13	2.32E-04	7.40E-05	2.46E-06	3.22E-02	1.07E-04	**1.33E-13**	1.04E-10
F_6	0.0927	0.3492	0.1189	0.0300	7.4448	0.6866	0.0136	**0.0043**
F_7	3.04E-01	3.92E + 00	1.28E + 00	2.37E + 00	6.67E + 00	7.16E-01	8.64E-02	**6.38E-02**
F_8	3.88E-06	5.01E-02	7.30E-02	5.10E-03	1.07E-01	3.16E-02	2.38E-04	**5.38E-07**

Bold values signify the result is the optimal value for the corresponding function.

Table 4: The optimization results when the dimension is 50.

	PSOBSA	FOPSO	HPSO	ASPSO	OLPSO	GEPSO	HMaPSO	APPSO
F_1	2.97E + 01	2.83E + 01	1.64E + 02	3.45E + 01	6.27E + 00	4.95E + 01	7.28E + 01	**1.00E + 01**
F_2	6.92E-01	4.64E + 02	3.47E + 02	1.04E + 00	4.01E + 00	1.57E + 00	4.41E + 01	**6.17E-01**
F_3	5.79E-04	3.41E + 02	6.26E + 01	1.12E + 00	6.73E + 01	1.78E-01	2.81E + 02	**3.47E-04**
F_4	2.57E + 01	2.44E + 01	1.42E + 02	2.98E + 01	**8.64E + 00**	4.27E + 01	6.28E + 01	5.42E + 00
F_5	**3.10E-03**	4.69E-01	2.97E-01	2.36E + 00	1.85E + 00	1.15E + 00	7.01E-01	1.99E-01
F_6	1.56E-02	6.96E + 00	9.21E + 00	9.61E-03	3.53E + 02	1.84E + 01	2.22E + 00	**3.18E-04**
F_7	6.10E + 01	1.68E + 03	8.43E + 02	**6.74E + 00**	1.64E + 03	5.93E + 01	3.29E + 02	6.81E + 01
F_8	3.22E-01	2.19E + 00	5.61E + 00	6.76E + 00	1.60E + 01	2.49E-01	9.31E + 00	**1.63E-02**

Bold values signify the result is the optimal value for the corresponding function.

Table 5: The optimization results when the dimension is 100.

	PSOBSA	FOPSO	HPSO	ASPSO	OLPSO	GEPSO	HMaPSO	APPSO
F_1	6.25E-04	8.81E + 01	3.41E + 01	4.26E + 01	7.78E-02	4.28E-01	2.67E + 02	**6.99E-05**
F_2	8.42E-01	1.10E + 01	1.78E + 03	1.08E + 03	8.01E + 00	3.59E + 01	5.68E + 02	**7.72E-01**
F_3	4.74E + 01	3.12E + 03	4.78E + 02	8.62E + 01	1.98E + 02	1.78E + 02	1.48E + 03	**2.90E + 00**
F_4	9.12E + 01	4.81E + 02	8.90E + 02	2.48E + 02	3.46E + 01	1.91E + 02	8.72E + 02	**6.42E + 00**
F_5	2.62E + 00	1.02E + 02	4.62E + 01	1.89E-01	4.54E + 00	1.80E + 01	2.55E + 02	**1.13E-01**
F_6	3.02E-01	3.29E-01	3.14E + 02	1.21E + 02	1.31E + 00	9.67E + 01	9.47E + 02	**1.76E-02**
F_7	2.43E + 02	3.70E + 05	3.03E + 04	1.13E + 02	1.08E + 02	2.39E + 02	1.60E + 06	**1.01E + 01**
F_8	2.61E-01	5.49E + 00	1.55E + 01	1.24E + 01	1.32E + 01	9.28E + 00	2.04E + 01	**3.18E-02**

Bold values signify the result is the optimal value for the corresponding function.

when performing unimodal function optimization or multimodal function optimization.

5.1.3. Comparison of the Running Time. Due to the running time of the algorithm is very important in many practical engineering applications, a set of experiments are conducted in this section to compare the running time of all the algorithms. The experimental results are shown in Table 6. The values in Table 6 represent the average usage time of the algorithm running 50 times independently on a function.

As can be seen from Table 6, the APPSO exhibits the best experimental results on all functions, which exhibit very competitive properties in terms of convergence speed. Although APPSO is relatively time-consuming to dynamically adjust particle positions, it is still ahead of other algorithms in running time, which is mainly because of the APPSO can achieve an appropriate balance between the accuracy of optimization and running time.

5.2. Network Traffic Anomaly Detection Comparison Experiment

5.2.1. Experimental Environment and Evaluation Metrics. To evaluate the performance of the FR-APPSO-BiLSTM model, this section adopts Accuracy, Precision, Recall, and *F*-score as evaluation metrics. These metrics are calculated as follows:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})}, \tag{13}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{14}$$

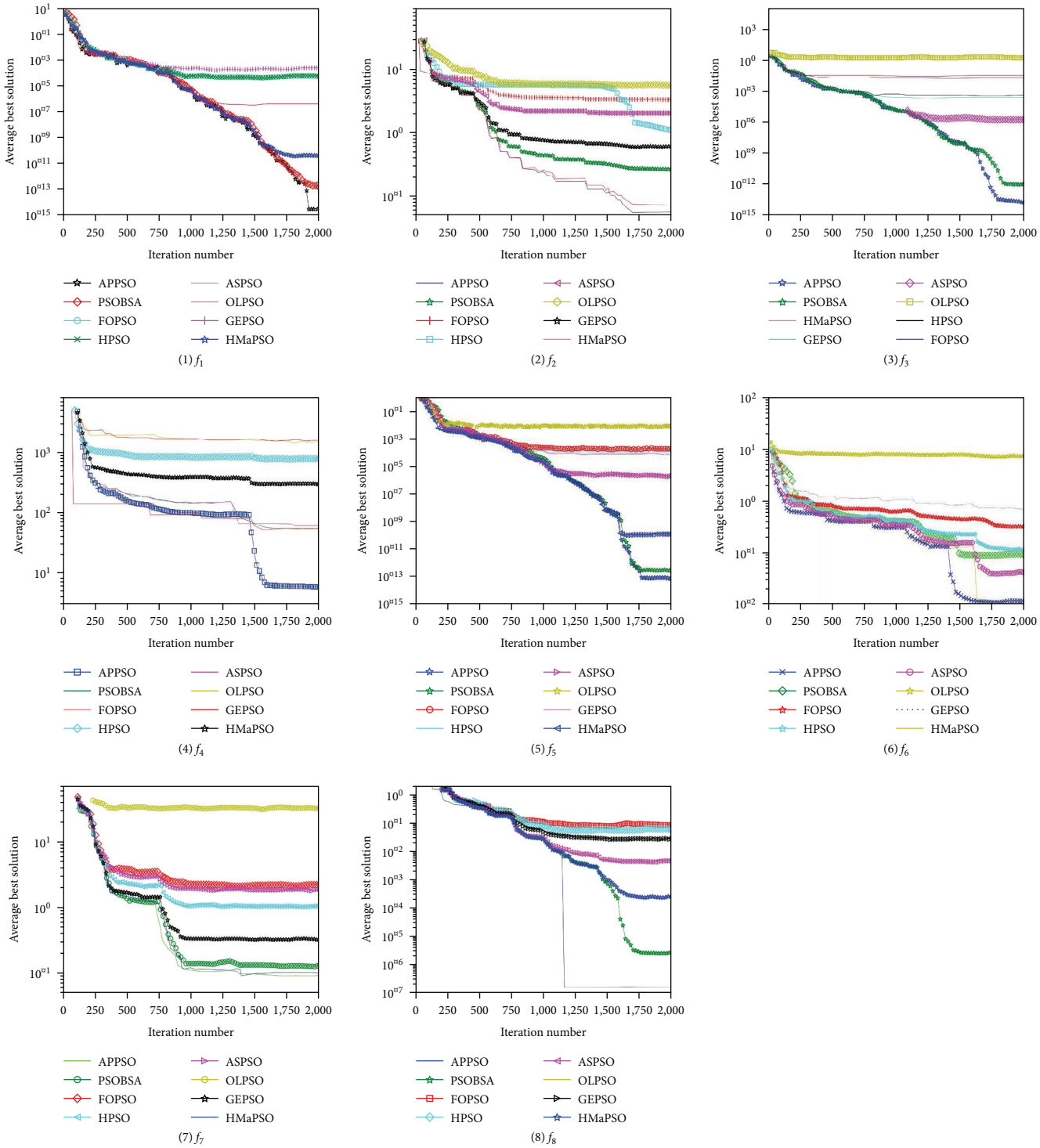

Figure 5: Convergence curve of the function f_1–f_8.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{15}$$

$$F\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{16}$$

where TP and TN mean true positive and true negative, respectively, indicating the attack and normal samples are correctly classified. A false negative (FN) refers to an attack sample that is wrongly classified as a normal one, and a false positive (FP) denotes a normal sample that is falsely considered as an attack.

5.2.2. Experimental Dataset. This section verifies the effectiveness of the FR-APPSO-BiLSTM model based on the datasets NSL-KDD, UNSW-NB15, and CICIDS-2017 [36–38]. The details of the three datasets are shown below.

Table 6: Average running time of each algorithm.

	PSOBSA	FOPSO	HPSO	ASPSO	OLPSO	GEPSO	HMaPSO	APPSO
F_1	39.01	27.65	27.79	28.29	32.18	26.27	25.72	24.61
F_2	55.44	40.95	42.38	40.13	48.14	38.09	37.29	35.68
F_3	46.23	28.63	32.12	29.62	37.13	27.21	26.63	25.48
F_4	61.59	46.42	47.37	45.82	53.03	43.52	42.61	40.78
F_5	37.17	22.87	23.16	21.04	29.68	19.99	19.57	18.73
F_6	43.00	28.19	30.42	27.79	37.54	26.40	25.84	24.73
F_7	44.95	30.84	32.38	30.36	39.79	28.84	28.23	27.02
F_8	47.72	31.59	31.61	29.51	40.34	28.04	27.45	26.27

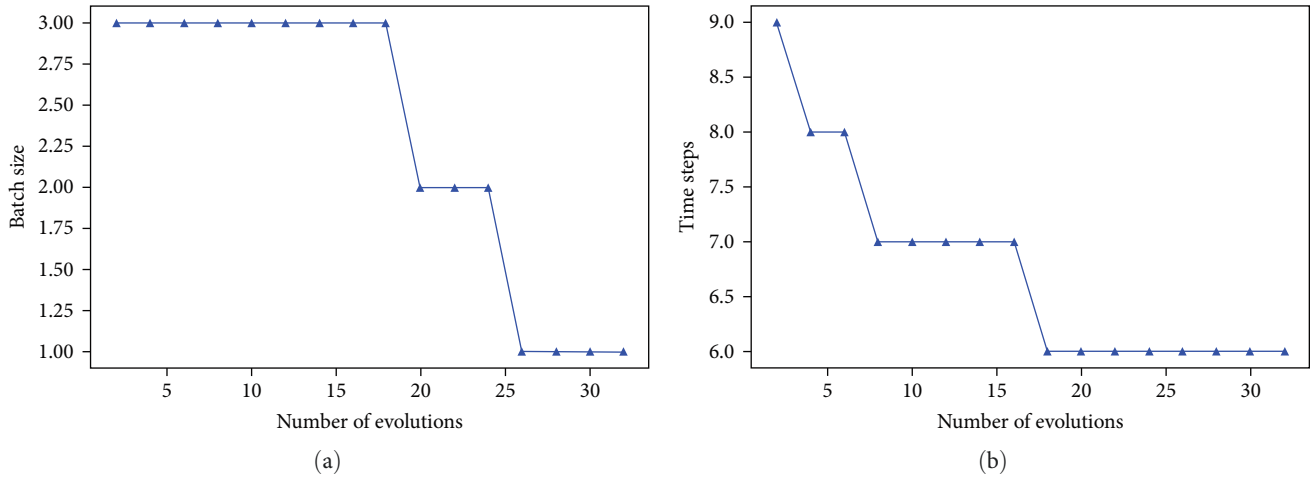

Figure 6: APPSO optimizes the parameters of BiLSTM.

(1) The NSL-KDD dataset consists of a training set and a test set, in which the training set has 125,973 records and the test set has 22,543 records. The dataset contains four attack types (Dos, Probe, R2L, and U2R), in which the amount of attack type data is much lower than that of normal types.

(2) The UNSW-NB15 dataset nine attack types collected by the Cyber Range Lab of the Australian Cyber Security Center, namely Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Normal. The dataset consists of a training set and a test set, with 175,341 records in the training set and 82,332 records in the test set.

(3) The CICIDS-2017 dataset contains network traffic based on packet and bidirectional flow formats, and each record contains 82 network flow features. Compared with NSL-KDD and UNSW-NB15, the CICIDS-2017 dataset includes a wider range of attack types, such as brute force attack, DoS, Heartbleed, Web penetration, and DDoS.

5.2.3. Parameter Selection of the BiLSTM. Figure 6 shows the training results of the APPSO algorithm optimizing the BiLSTM neural network. The time step size and batch size gradually converge to the optimal value with the update of the algorithm. As can be seen from Figure 6, the batch size of model training data is 1, and the optimal time step is 6. So far, the best superparameters are obtained to modify the model structure of the BiLSTM neural network and obtain the best parameter combination.

Table 7: Parameters setup of the model.

Parameter	Value
Maximum number of iterations	2,000
Number of the population particles	25
Dimension	6
Time steps	6
Batch size	1

5.2.4. Parameter Settings of the FR-APPSO-BiLSTM Model. In the FR-APPSO-BiLSTM model, the settings of related parameters are shown in Table 7.

After FR using hierarchical clustering and AEs, the feature subsets of the three datasets are shown in Table 8.

5.2.5. Analysis of Simulation Results. In this section, the FR-APPSO-BiLSTM model is compared with FR-APPSO-LSTM, FR-BiLSTM, and APPSO-BiLSTM to verify the effectiveness of FR based on hierarchical clustering and AE, and the effectiveness of the BiLSTM parameter optimization

Table 8: Feature selection results.

Dataset	Feature	Number
NSL-KDD	protocol_type, flag, count, serror_rate, rerror_rate, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate	13
UNSW-NB15	Proto, dttl, dloss, sinpkt, swin, stcpb, dmean, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, is_sm_ips_ports	11
CICIDS-2017	Destination Port, Flow Duration, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std, Flow Packetss, Flow IAT Mean, Flow IAT Std, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Bwd Bwd IAT Max, Fwd PSH Flags, Min Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance, SYN Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count, Down/Up Ratio, Average Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Init_Win_bytes_backward, Idle Mean, Idle Std, Idle Max, Idle Min	34

Table 9: Detection results of the four models.

Dataset	Model	Evaluation metrics			
		Accuracy (%)	Precision (%)	Recall (%)	*F*-score (%)
NSL-KDD	FR-APPSO-LSTM	90.02	83.33	95.26	88.89
	FR-BiLSTM	90.81	84.46	97.54	90.53
	APPSO-BiLSTM	90.40	83.86	95.97	89.61
	FR-APPSO-BiLSTM	91.76	85.37	98.50	91.46
UNSW-NB15	FR-APPSO-LSTM	85.19	92.97	92.26	92.61
	FR-BiLSTM	89.84	97.01	97.23	97.12
	APPSO-BiLSTM	91.41	96.92	97.00	97.06
	FR-APPSO-BiLSTM	92.08	97.88	98.32	98.10
CICIDS-2017	FR-APPSO-LSTM	92.29	86.15	87.58	86.86
	FR-BiLSTM	93.95	98.25	97.79	98.02
	APPSO-BiLSTM	94.20	98.02	97.73	98.12
	FR-APPSO-BiLSTM	95.44	98.58	98.40	98.49

based on the APPSO. The FR-APPSO-BiLSTM model is compared with FR-ASPSO-BiLSTM, FR-QPSO-BiLSTM, and FR-HPSO-BiLSTM to verify the effectiveness of the BiLSTM optimized based on APPSO compared to the BiLSTM optimized by other PSO algorithms. Finally, the FR-APPSO-BiLSTM model is compared with other existing detection models to verify its detection effect.

(*1*) *Comparison of FR-APPSO-BiLSTM and FR-APPSO-LSTM*, *FR-BiLSTM*, *APPSO-BiLSTM*. This section compares the FR-APPSO-BiLSTM model with FR-APPSO-LSTM, FR-BiLSTM, and APPSO-BiLSTM on three datasets. The experimental results are shown in Table 9 and Figure 7.

It can be seen that, compared with the FR-APPSO-LSTM, FR-BiLSTM, and APPSO-BiLSTM, the FR-APPSO-BiLSTM model has higher classification accuracy and better detection effect on the three datasets. For the four indicators on the NSL-KDD dataset, the FR-APPSO-BiLSTM model improved by 1.93%, 2.45%, 3.40%, and 2.89% compared with the FR-APPSO-LSTM model, improved by 1.05%, 1.08%, 0.98%, and 1.03% compared with the FR-BiLSTM model, improved by 1.83%, 2.00%, 1.35%, and 1.57% compared with the APPSO-BiLSTM model. For the four indicators on the UNSW-NB15 dataset, the FR-APPSO-BiLSTM model improved by 8.08%, 5.29%, 6.57%, and 5.93% compared with the FR-APPSO-LSTM model, improved by 2.49%, 0.90%, 1.12%, and 1.01% compared with the FR-BiLSTM model, improved by 1.67%, 1.16%, 0.97%, and 1.16% compared with the APPSO-BiLSTM model. For the four indicators on the CICIDS-2017 dataset, the FR-APPSO-BiLSTM model improved by 3.41%, 14.43%, 12.35%, and 13.39% compared with the FR-APPSO-LSTM model, improved by 1.59%, 0.33%, 0.62%, and 0.48% compared with the FR-BiLSTM model, improved by 0.35%, 0.37%, 0.76%, and 0.42% compared with the APPSO-BiLSTM model.

The above results show that the FR-APPSO-BiLSTM model has a higher detection effect. On the one hand, it uses hierarchical clustering and AE for FR of data, which eliminates redundant information in the data. On the other hand, due to the APPSO algorithm is used for parameter optimization of BiLSTM. The combination of these two mechanisms for anomaly detection makes the model have better detection effect and higher detection efficiency.

(*2*) *Comparison of FR-APPSO-BiLSTM and FR-ASPSO-BiLSTM*, *FR-QPSO-BiLSTM*, *FR-HPSO-BiLSTM*. This section

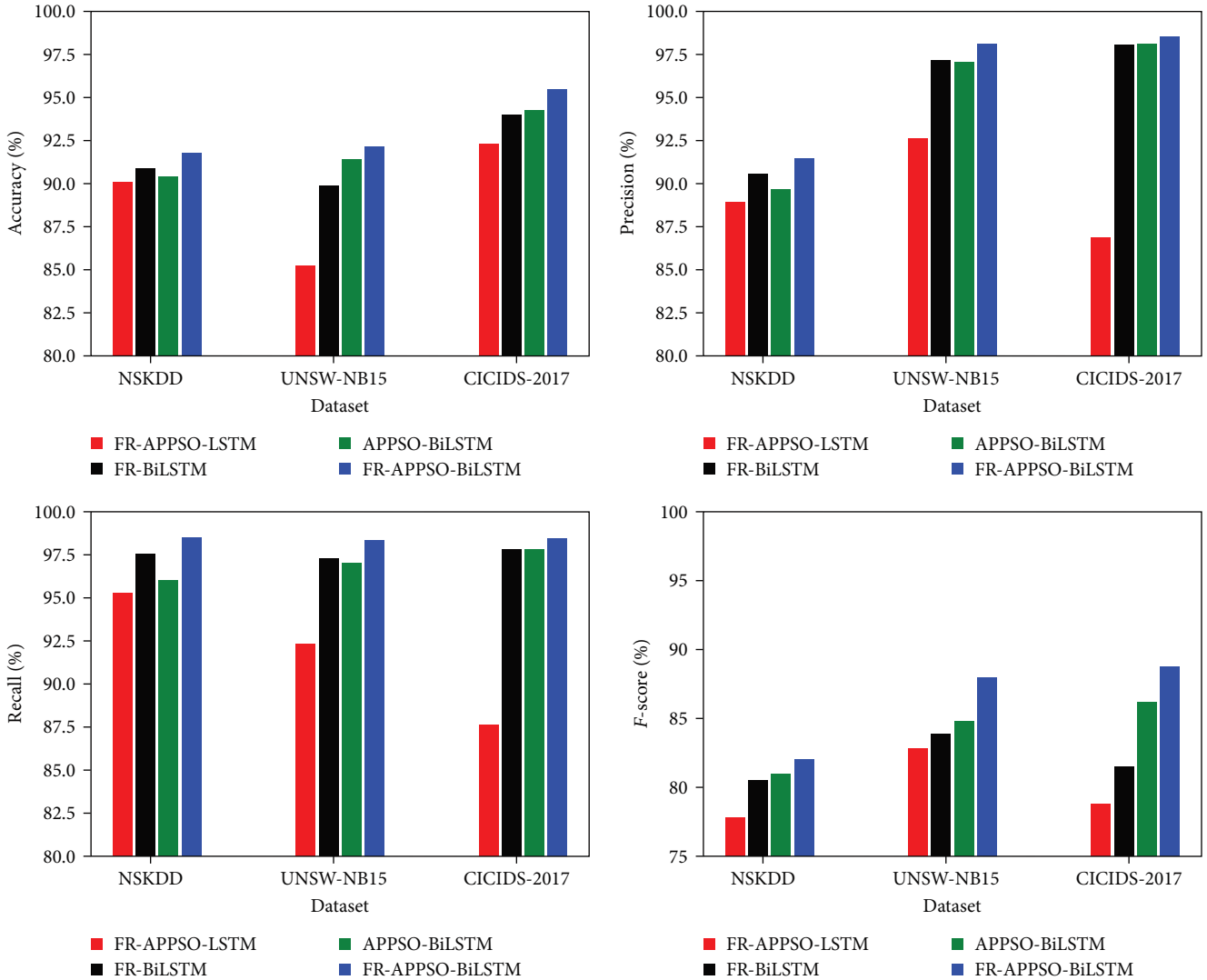

Figure 7: Detection results of the four models.

compares the FR-APPSO-BiLSTM model with FR-ASPSO-BiLSTM (optimized BiLSTM based on ASPSO [32]), FR-QPSO-BiLSTM (optimized BiLSTM based on QPSO [39]) and FR-HPSO-BiLSTM (optimized BiLSTM based on HPSO [31]) for experimental comparison on three datasets, the experimental results are shown in Table 10 and Figure 8.

It can be seen that, compared with the FR-ASPSO-BiLSTM, FR-QPSO-BiLSTM, and FR-HPSO-BiLSTM, the FR-APPSO-BiLSTM model has higher classification accuracy and better detection effect on the three datasets. For the four indicators on the NSL-KDD dataset, the FR-APPSO-BiLSTM model improved by 0.85%, 0.37%, 1.18%, and 0.74% compared with the FR-ASPSO-BiLSTM model, improved by 0.33%, 0.09%, 0.92%, and 0.48% compared with the FR-QPSO-BiLSTM model, improved by 1.10%, 0.57%, 1.33%, and 0.90% compared with the FR-HPSO-BiLSTM model. For the four indicators on the UNSW-NB15 dataset, the FR-APPSO-BiLSTM model improved by 2.49%, 0.90%, 1.12%, and 1.01% compared with the FR-ASPSO-BiLSTM model, improved by 0.85%, 0.53%, 1.04%, and 0.77% compared with the FR-QPSO-BiLSTM model, improved by 0.85%, 0.53%, 1.04%, and 0.77% compared with the FR-HPSO-BiLSTM model. For the four indicators on the CICIDS-2017 dataset, the FR-APPSO-BiLSTM model improved by 0.43%, 0.22%, 0.55%, and 0.39% compared with the FR-ASPSO-BiLSTM model, improved by 0.25%, 0.11%, 0.27%, and 0.19% compared with the FR-QPSO-BiLSTM model, improved by 0.43%, 0.30%, 0.45%, and 0.35% compared with the FR-HPSO-BiLSTM model.

The above results show that the parameter optimization of BiLSTM based on APPSO is significantly better than the BiLSTM optimized by other PSO algorithms. The BiLSTM

Table 10: Detection results of the four models.

Dataset	Model	Evaluation metrics			
		Accuracy (%)	Precision (%)	Recall (%)	*F*-score (%)
NSL-KDD	FR-ASPSO-BiLSTM	90.99	85.06	97.35	90.79
	FR-QPSO-BiLSTM	91.46	85.29	97.60	91.03
	FR-HPSO-BiLSTM	90.92	84.90	97.19	90.55
	FR-APPSO-BiLSTM	91.76	85.37	98.50	91.46
UNSW-NB15	FR-ASPSO-BiLSTM	89.84	97.01	97.23	97.12
	FR-QPSO-BiLSTM	91.15	97.53	97.45	97.49
	FR-HPSO-BiLSTM	91.14	97.54	97.75	97.77
	FR-APPSO-BiLSTM	92.08	97.88	98.32	98.10
CICIDS-2017	FR-ASPSO-BiLSTM	95.03	98.36	97.85	98.11
	FR-QPSO-BiLSTM	95.21	98.47	98.13	98.30
	FR-HPSO-BiLSTM	94.99	98.21	97.91	98.01
	FR-APPSO-BiLSTM	95.44	98.58	98.40	98.49

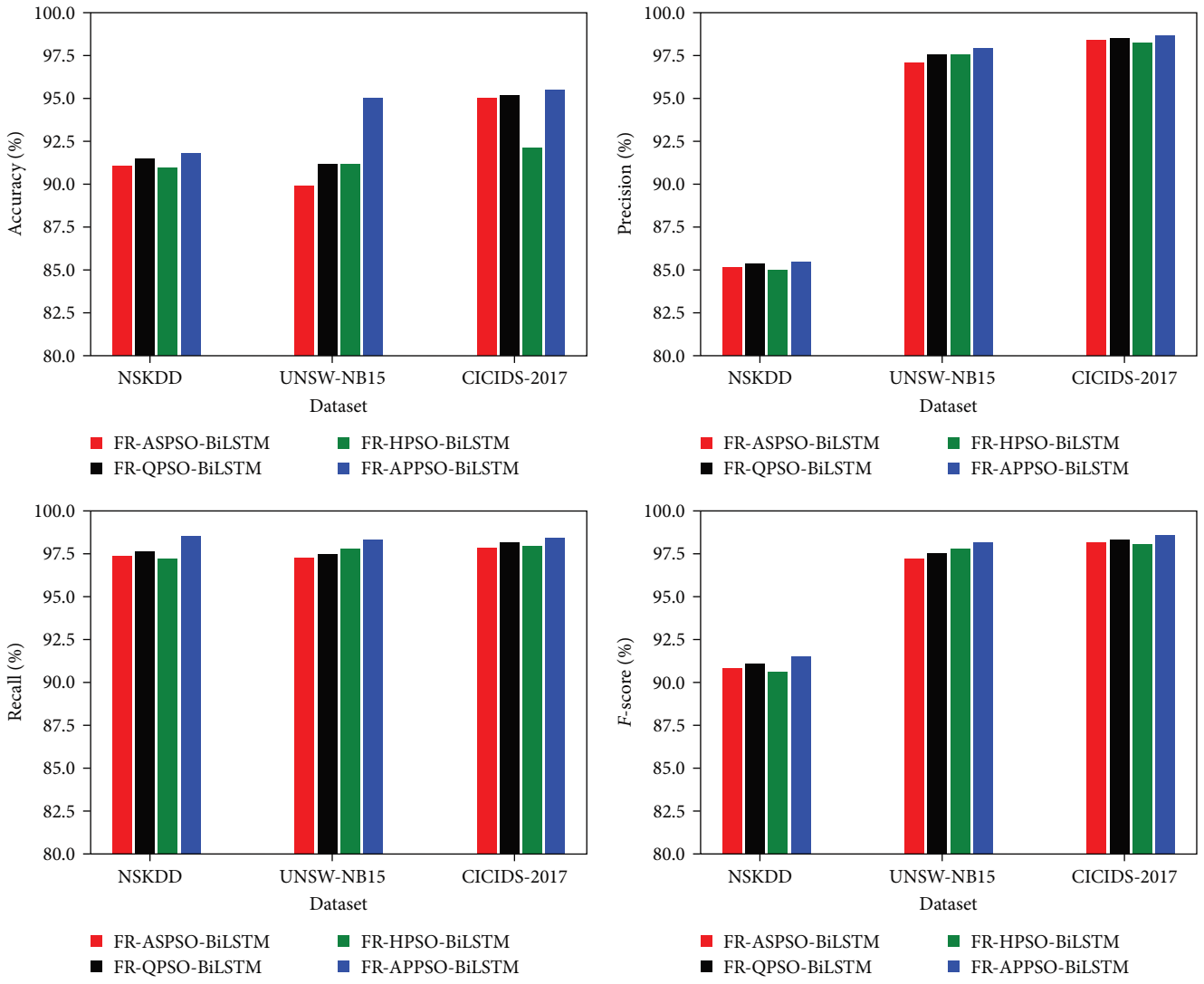

Figure 8: Detection results of the four models.

Table 11: Detection results of the six models.

Dataset	Model	Evaluation metrics			
		Accuracy (%)	Precision (%)	Recall (%)	*F*-score
NSL-KDD	HCRNNIDS	84.46	82.15	92.14	86.86
	ADASYN-LightGBM	86.25	82.97	94.05	88.16
	LNNLS-KH	89.16	83.64	96.44	89.58
	STL-HDL	91.02	82.35	94.98	88.21
	E-GraphSAGE	90.20	83.92	96.03	89.57
	FR-APPSO-BiLSTM	91.76	85.37	98.50	91.46
UNSW-NB1	HCRNNIDS	88.28	85.92	92.43	89.06
	ADASYN-LightGBM	89.36	90.27	91.44	90.85
	LNNLS-KH	90.44	94.35	91.08	92.69
	STL-HDL	90.29	92.19	96.14	94.12
	E-GraphSAGE	91.42	93.46	94.43	93.94
	FR-APPSO-BiLSTM	92.08	97.88	98.32	98.10
CICIDS-2017	HCRNNIDS	85.90	94.34	90.44	92.35
	ADASYN-LightGBM	92.00	97.30	95.44	96.36
	LNNLS-KH	91.57	89.31	90.35	89.82
	STL-HDL	90.46	95.03	93.23	94.12
	E-GraphSAGE	93.26	92.35	94.51	93.41
	FR-APPSO-BiLSTM	95.44	98.58	98.40	98.49

optimized based on APPSO significantly improves the detection ability for abnormal data, mainly due to the following reasons: (1) using hierarchical clustering and automatic encoder for FR of data, eliminating redundant information in the data; (2) the APPSO has better dynamic adaptability than ASPSO, QPSO, and HPSO. The particles in the population can automatically adjust the search direction according to the changes of the search environment, with better rapid convergence and optimization capabilities, which makes it possible to search more predictable parameters for BiLSTM.

(*3*) *Comparison with the Other Existing Detection Models*. This section compares the FR-APPSO-BiLSTM model with other existing methods in the literature (HCRNNIDS [40], ADASYN-LightGBM [41], LNNLS-KH [42], STL-HDL [43], and E-GraphSAGE [44]), the results are shown in Table 11 and Figure 9.

It can be seen that, compared with other existing methods, the FR-APPSO-BiLSTM model has higher classification accuracy on all three datasets. Compared with the HCRNNIDS model, the Accuracy values of FR-APPSO-BiLSTM on the three datasets increased by 8.65%, 4.30%, and 11.11%, the Precision values increased by 3.92%, 13.93%, and 4.49%, the Recall values increased by 6.90%, 6.36%, and 8.80%, and the *F*-score value increased by 5.30%, 10.15%, and 6.64%. Compared with the ADASYN-LightGBM model, the Accuracy values of FR-APPSO-BiLSTM on the three datasets increased by 6.39%, 3.04%, and 3.74%, the Precision values increased by 2.90%, 8.43%, and 1.31%, the Recall values increased by 4.73%, 7.52%, and 3.10%, the *F*-score value increased by 3.75%, 7.98%, and 2.20%. Compared with the LNNLS-KH model, the Accuracy values of FR-APPSO-BiLSTM on the three datasets increased by 2.92%, 1.80%, and 4.24%, the Precision values increased by 2.07%, 3.74%, and 10.38%, the Recall values increased by 2.13%, 7.94%, and 8.91%, the *F*-score value increased by 2.10%, 5.84%, and 9.64%. Compared with the STL-HDL model, the Accuracy values of FR-APPSO-BiLSTM on the three datasets increased by 0.81%, 1.98%, and 5.51%, the Precision values increased by 3.67%, 6.17%, and 3.74%, the Recall values increased by 3.70%, 2.26%, and 5.54%, the *F*-score value increased by 3.69%, 4.22%, and 4.64%. Compared with the E-GraphSAGE model, the Accuracy values of FR-APPSO-BiLSTM on the three datasets increased by 1.74%, 0.72%, and 2.35%, the Precision values increased by 1.72%, 4.74%, and 6.75%, the Recall values increased by 2.57%, 4.12%, and 4.11%, the *F*-score value increased by 2.12%, 4.43%, and 5.43%. The above results fully demonstrate that the FR-APPSO-BiLSTM model has better detection performance on all three datasets, which verifies the effectiveness of the model. The FR-APPSO-BiLSTM model has better performance compared to other existing models, mainly due to the following reasons: (1) the proposed FR algorithm based on hierarchical clustering and automatic encoder effectively reduces the dependence of the detection model on the dataset size while preserving the effective information of network traffic data; (2) the combination of FR, APPSO, and BiLSTM effectively leverages their advantages and overcomes the inherent shortcomings of existing models.

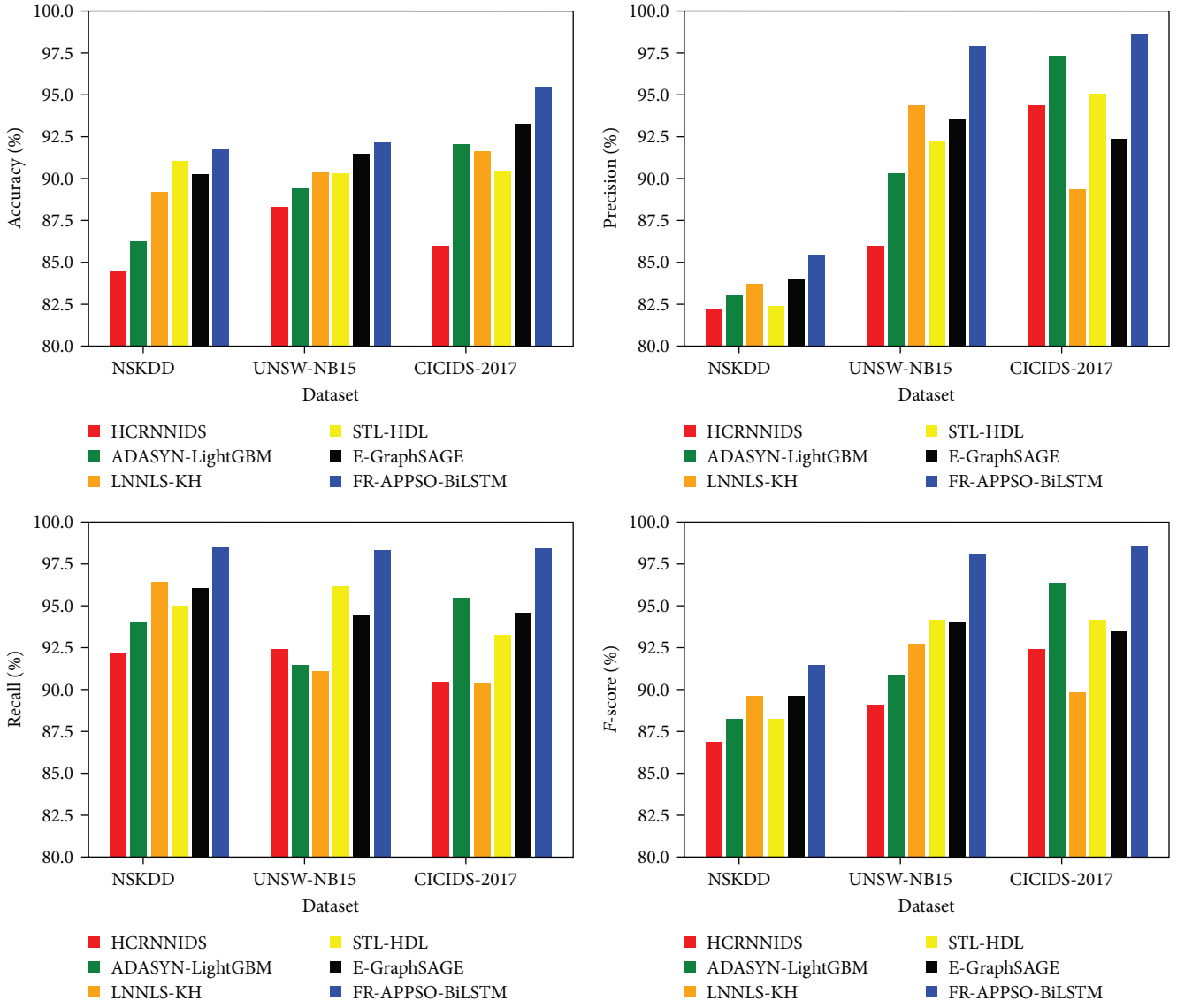

Figure 9: Detection results of the six models.

6. Conclusions and Future Work

To solve the problems of large network data, high feature dimension and high dependence of traditional machine learning algorithms on data labels in anomaly intrusion detection, we propose a network traffic anomaly detection model (FR-APPSO-BiLSTM) based on FR and BiLSTM neural network optimization. In FR-APPSO-BiLSTM, hierarchical clustering method and automatic encoder are combined to reduce the characteristics of network traffic data. APPSO is used to optimize the parameters of BiLSTM. The optimized BiLSTM is used as the classifier, and the processed feature data is used as the input of the optimal classifier for network traffic anomaly detection. In the experiment, the APPSO algorithm is compared with other PSO algorithms for benchmark function optimization, and the results show that it has better convergence speed and optimization effect; then NSL-KDD, UNSW-NB15, and CICIDS-2017 datasets are used to conduct network traffic anomaly detection simulation experiments. The experimental results show that the FR-APPSO-BiLSTM model can obtain better evaluation indicators and has better detection performance.

In the future, we will enhance the pattern matching speed to analyze the high-speed networks in a better way to detect and protect them from network attacks. Different types of more attacks will take into account to check the proposed model's performance with more performance parameters.

Data Availability

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

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors' Contributions

Hanqing Jiang performed the experiments. Shaopei Ji analyzed the data and wrote the paper. Hanqing Jiang and Shaopei Ji contributed equally to this work. Guanghui He and Xiaohu Li supervised the research and critically revised the paper. All authors have read and agreed to the published version of the manuscript. Hanqing Jiang and Shaopei ji are the co-first authors.

Acknowledgments

This research was supported by the Joint fund for enterprise innovation and development of the National Natural Science Foundation of China (U20B2049).

References

[1] M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee, "Multiresolution knowledge distillation for anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14902–14912, IEEE, 2021.

[2] Y. Wang, X. Feng, and T. Qian, "Disguised user intrusion detection based on CNN and LSTM deep network," *Journal of Frontiers of Computer Science and Technology*, vol. 12, no. 4, pp. 575–585, 2018.

[3] R. Vijayanand and D. Devaraj, "A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network," *IEEE Access*, vol. 8, pp. 56847–56854, 2020.

[4] M. Labonne, A. Olivereau, B. Polvé, and D. Zeghlache, "A cascade-structured meta-specialists approach for neural network-based intrusion detection," in *Proceedings of the 16th IEEE Consumer Communications & Networking Conference*, pp. 1–6, IEEE, Piscataway, 2019.

[5] W. H. Lin, H. C. Lin, P. Wang, B. H. Wu, and J. Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *Proceedings of the 2018 IEEE International Conference on Applied System Innovation*, pp. 1107–1110, IEEE, Piscataway, 2018.

[6] R. Blanco, P. Malagón, S. Briongos, and J. M. Moya, *Anomaly Detection Using Gaussian Mixture Probability Model to Implement Intrusion Detection System*, pp. 648–659, Springer, Cham, 2019.

[7] R. K. Phanindra, S. C. Noorullah, and L. B. Rajkumar, "An anomaly-based intrusion detection system using recursive feature elimination technique for improved attack detection," *Theoretical Computer Science*, vol. 6, no. 6, pp. 474–481, 2022.

[8] L. Su, Y. Yao, N. Li, J. Liu, Z. Lu, and B. Liu, "Hierarchical clustering based network traffic data reduction for improving suspicious flow detection," in *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 744–753, IEEE, New York, NY, USA, 2018.

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

[10] J. O. Onah, M. Abdullahi, I. H. Hassan, and A. Al-Ghusham, "Genetic algorithm based feature selection and Naïve Bayes for anomaly detection in fog computing environment," *Machine Learning with Applications*, vol. 6, Article ID 100156, 2021.

[11] H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Generation Computer Systems*, vol. 122, pp. 130–143, 2021.

[12] J. Liu, J. He, W. Zhang et al., "ANID-SEoKELM: adaptive network intrusion detection based on selective ensemble of kernel ELMs with random features," *Knowledge-Based Systems*, vol. 177, no. 1, pp. 104–116, 2019.

[13] A. O. Alzahrani and M. J. F. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, Article ID 111, 2021.

[14] L. Jinghao, M. Siping, and F. Xiaomei, "Intrusion detection model based on ICA algorithm and deep neural network," *Information Network Security*, vol. 3, pp. 1–10, 2019.

[15] S. Long, Y. Lina, W. Nannan, and J. Shaopei, "Network anomaly detection model based on GSA and DE optimized hybrid kernel ELM," *Computer Engineering*, vol. 6, pp. 146–153, 2022.

[16] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, Article ID 102031, 2020.

[17] S. C. Zhang, X. Y. Xie, and Y. Xu, "Intrusion detection method based on dCNN," *Journal of Tsinghua University (Natural Science Edition)*, vol. 59, no. 1, pp. 46–54, 2019.

[18] 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.

[19] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, and D. Malerba, "Exploiting the auto-encoder residual error for intrusion detection," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops*, pp. 281–290, IEEE, Piscataway, 2019.

[20] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential LSTM neural networks autoencoders," in *Proceedings of the 26th Signal Processing and Communications Applications Conference*, pp. 1–4, IEEE, Piscataway, 2018.

[21] P. Contreras and F. Murtagh, "Hierarchical clustering," in *Handbook of Cluster Analysis*, pp. 103–123, Taylor and Francis, Boca Raton, FL, 2015.

[22] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[23] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Applied Soft Computing*, vol. 100, Article ID 106960, 2021.

[24] K. Okamura, "Characterizations of the Cauchy distribution associated with integral transforms," *Studia Scientiarum Mathematicarum Hungarica*, vol. 57, no. 3, pp. 385–396, 2020.

[25] M. Chankaya, I. Hussain, A. Ahmad, H. Malik, and F. P. García Márquez, "Generalized normal distribution algorithm-based control of 3-phase 4-wire grid-tied PV-hybrid energy storage system," *Energies*, vol. 14, no. 14, Article ID 4355, 2021.

[26] C. Shi and S. Yuhui, "Measurement of PSO diversity based on L1 norm," *Computer Science*, vol. 38, no. 7, pp. 190–193, 2011.

[27] L. Jian, H. Xiang, and G. Le, "LSTM-based attentional embedding for English machine translation," *Scientific Programming*, vol. 2022, Article ID 3909726, 8 pages, 2022.
[28] G. Li, F. Li, C. Xu, and X. Fang, "A spatial-temporal layer-wise relevance propagation method for improving interpretability and prediction accuracy of LSTM building energy prediction," *Energy and Buildings*, vol. 271, Article ID 112317, 2022.
[29] R. Zaman H. R. and F. S. Gharehchopogh, "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems," *Engineering with Computers*, vol. 38, pp. 2797–2831, 2022.
[30] S. M. Abedi Pahnehkolaei, A. Alfi, and J. A. Tenreiro Machado, "Analytical stability analysis of the fractional-order particle swarm optimization algorithm," *Chaos, Solitons & Fractals*, vol. 155, Article ID 111658, 2022.
[31] H. Mehmet and M. H. Ibrahim, "A novel multimean particle swarm optimization algorithm for nonlinear continuous optimization: application to feed-forward neural network training," *Scientific Programming*, vol. 4, Article ID 1435810, 9 pages, 2018.
[32] S.-P. Ji, Y.-L. Meng, L. Yan, G.-S. Dong, and D. Liu, "GRU-corr neural network optimized by improved PSO algorithm for time series prediction," *International Journal on Artificial Intelligence Tools*, vol. 29, no. 8, Article ID 2040010, 2020.
[33] A. Darwish, D. Ezzat, and A. E. Hassanien, "An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis," *Swarm and Evolutionary Computation*, vol. 52, Article ID 100616, 2020.
[34] D. Sedighizadeh, E. Masehian, M. Sedighizadeh, and H. Akbaripour, "GEPSO: a new generalized particle swarm optimization algorithm," *Mathematics and Computers in Simulation*, vol. 179, pp. 194–212, 2021.
[35] R. Cong and H. Wang, "Prediction of evolution and development trend in sports industry cluster based on particle swarm optimization," *Scientific Programming*, vol. 2021, Article ID 7607623, 8 pages, 2021.
[36] B. M. Serinelli, A. Collen, and N. A. Nijdam, "Training guidance with KDD cup 1999 and NSL-KDD data sets of ANIDINR: anomaly-based network intrusion detection system," *Procedia Computer Science*, vol. 175, pp. 560–565, 2020.
[37] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Scientific Programming*, vol. 10, no. 12, pp. 151–173, 2023.
[38] N. Gulia, K. Solanki, S. Dalal, A. Dhankhar, O. Dahiya, and N. U. Salmaan, "Intrusion detection system using the G-ABC with deep neural network in cloud environment," *Scientific Programming*, vol. 2023, Article ID 7210034, 15 pages, 2023.
[39] C. Wei, Z. He, Y. Zhang, and W. Pei, "Swarm directions embedded in fast evolutionary programming," *Proceedings of IEEE Congress on Evolutionary Computation*, vol. 10, no. 6, pp. 1278–1283, 2002.
[40] M. A. Khan, "HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, Article ID 834, 2021.
[41] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM," *Computers & Security*, vol. 106, Article ID 102289, 2021.
[42] X. Li, P. Yi, W. Wei, Y. Jiang, and L. Tian, "LNNLS-KH: a feature selection method for network intrusion detection," *Security and Communication Networks*, vol. 2021, Article ID 8830431, 22 pages, 2021.
[43] S. Al and M. Dener, "STL-HDL: a new hybrid network intrusion detection system for imbalanced dataset on big data environment," *Computers & Security*, vol. 110, Article ID 102435, 2021.
[44] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: a graph neural network based intrusion detection system for IoT," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2022.