IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 January 2024, accepted 3 February 2024, date of publication 13 February 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365495

RESEARCH ARTICLE

Research on Dung Beetle Optimization Based Stacked Sparse Autoencoder for Network Situation Element Extraction

YONGCHAO YANG AND PAN ZHAO
School of Big Data and Artificial Intelligence, Chizhou University, Chizhou 247000, China

Corresponding author: Yongchao Yang (yyc0424@mail.ustc.edu.cn)

This work was supported in part by Chizhou University under Grant CZ2022ZRZ05; and in part by the Natural Science Research Key Project of the Education Department of Anhui Province, China, under Grant 2022AH051828.

ABSTRACT Network security situation awareness enables networks to actively and effectively defend against network attacks, relying on the extraction of network situation elements as an initial and decisive step. In existing studies, the stacked sparse autoencoder (SSAE) has been employed to extract features from unlabeled network flows. However, obtaining the optimal hyperparameter combination is challenging due to its numerous hyperparameters. To address this issue, we propose a novel approach named DBO-SSAE that leverages dung beetle optimization (DBO) to select the optimal hyperparameters for SSAE automatically. Applied to the well-known UNSW-NB15 dataset, our model yields an optimal feature subset, which is evaluated across various binary classifiers with different metrics. Experimental results demonstrate that our approach improves *accuracy* and F_1-*measure* by 0.2% to 1.5% while reducing the *false negative rate* (*FNR*) and *false positive rate* (*FPR*) by 0.06% to 7%, surpassing other feature extraction methods on the same classifier for the UNSW-NB15 dataset. Particularly, in conjunction with a lightweight bidirectional long short-term memory (BiLSTM), our model achieves metrics of 98.84% *accuracy*, 98.96% F_1-*measure*, 1.86% *FNR*, and 0.6% *FPR*. This study could provide novel insights into the effective representation of network situation elements and lay the groundwork for a high-efficiency intrusion detection system.

INDEX TERMS Dung beetle optimization, network security, network situation element extraction, stacked sparse autoencoder.

I. INTRODUCTION

Currently, with the continuous improvement of the Internet infrastructure and the steady increase in the number of Internet users, network security problems are increasingly prominent, and network attacks gradually show the characteristics of complexity, organization, and persistence [1]. Traditional network security defense devices, such as firewalls and intrusion detection systems (IDS), have problems such as operating independently, passively defending against unknown attacks, and being unable to cope with the current large-scale and ever-changing network environment [2]. The topic of how to effectively defend against network attacks has become a hot topic in the field of network security.

By extracting and understanding the situation elements contained in multiple sources of information in the large-scale network environment, network security situation awareness (NSSA) [3] can evaluate the current network situation and predict the future development trend of the network situation, which can actively and effectively defend against network attacks. NSSA consists of three layers: situation element extraction, situation assessment, and situation prediction. At present, most of the research on NSSA focuses on the establishment of assessment or prediction models, but the situation element extraction is even more important. With the right features, most of the classifiers will perform excellently, achieving high classification accuracy and fast calculation speed simultaneously.

As the initial stage of NSSA, situation element extraction aims at mining hidden features in the high-dimensional

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

network traffic flow. The core problem is how to achieve dimensionality reduction and feature extraction effectively.

For a long time, the researchers spent a lot of time slowly hand-engineering the features for each dataset [4]. Ideally, we would like to extract the features automatically for different datasets. The two most widely used automatic feature extraction methods are principal component analysis (PCA) and autoencoder (AE) [5]. Furthermore, the training process for both methods does not require labels, which solves the problem of manual labeling caused by the huge network traffic.

PCA performs well in linear data, but the typical characteristic of the network information flow is non-linear. Compared with PCA, the use of AE for feature reduction can better capture nonlinear relationships between features, making it more suitable for deep learning model classifiers. Thus, numerous AEs and their variants have been applied for feature reduction in intrusion detection. As a variant of AE, the stacked sparse autoencoder (SSAE) sets a sparsity parameter close to 0 to compress the dimension of the feature [6], and it has been proven to be useful in various fields, including intrusion detection.

However, one challenge with SSAE is its numerous hyperparameters. With different hyperparameters, the representation ability of features outputted by SSAE varies. Many researchers conduct numerous comparative experiments to identify a relatively superior hyperparameter combination based on their experience [7], [8], investing a considerable amount of time. Swarm intelligence (SI) optimization algorithms are well-suited for solving hyperparameter optimization problems [9], [10], but there has been limited research on SI-based hyperparameter optimization specifically for SSAE in intrusion detection scenarios. Moreover, the existing literature faces several challenges. Firstly, many SI algorithms are susceptible to being trapped in a local optimum, which can impact the final solution [11]. Secondly, the number of nodes in each latent layer influences the performance of the SSAE, but this aspect is often neglected in optimization efforts [12]. Lastly, the fitness function significantly impacts solution quality, but it is often simplistically set as either the loss or accuracy in previous works [13], leading to less noticeable improvements in feature representation ability.

To address these issues, we employ a novel SI optimization algorithm named dung beetle optimization (DBO) [14] and introduce a more appropriate fitness function, optimizing the critical hyperparameters of SSAE for network situation element extraction. The main contributions of this paper are as follows.

1) A novel feature reduction method named dung beetle optimization-based stacked sparse autoencoder (DBO-SSAE) is proposed to reduce the feature dimensionality of network traffic flows, aiming to decrease time costs while maintaining the accuracy of intrusion detection.
2) A novel meta-heuristic DBO is used to automatically tune four crucial hyperparameters (the number of nodes in each latent layer and the sparsity parameter) of the SSAE. DBO is chosen because of its competitive performance in terms of convergence speed and searching accuracy.
3) The quadratic sum of losses, rather than the sum of losses in each layer of SSAE described in previous works, is set as the fitness function of DBO to obtain the feature subset with the least loss.
4) The performance of DBO-SSAE is evaluated in comparison to two other extraction methods (SSAE with manually tuned parameters and PCA) on the benchmark dataset UNSW-NB15 using six machine learning models, demonstrating the superior feature extraction capability of DBO-SSAE.
5) The proposed DBO-SSAE, combined with a lightweight bidirectional long short-term memory (BiLSTM), outperforms many other advanced approaches on the UNSW-NB15 dataset in binary classification, further evidencing its superior performance.

The remainder of this paper is arranged as follows: Section II provides a review of existing work, while Section III introduces the methods of SSAE, DBO, and the proposed model DBO-SSAE. In Section IV, we describe the procedures for the overall experiment and analyze the results. Section V discusses the limitations of the proposed method. Finally, Section VI presents the conclusions of this paper and outlines future work.

II. RELATED WORKS

In this section, we present an overview of automatic network traffic feature extraction and SI-based hyperparameter optimization for both traditional machine learning models and deep learning models.

A. AUTOMATIC FEATURE EXTRACTION

Automatic feature extraction methods help researchers break free from extensive and tedious data analysis tasks, making them more convenient to use. Among various methods, PCA and AE have been extensively utilized in the field of network security.

Some works [15], [16], [17] used PCA to reduce the dimension of network flows and then fed the feature subset into a support vector machine (SVM) classifier on the benchmark datasets (KDD Cup 99 [18], NSL-KDD [19], UNSW-NB15 [20]) for intrusion detection, illustrating the effectiveness of the method. In [21], improved PCA was used to reduce data pollution, combined with a Gaussian naive Bayes (GNB) classifier for the detection of network intrusion. Waskle et al. [22] fused PCA and random forest (RF) on the KDD Cup 99 dataset, enhancing the accuracy and efficiency of classification simultaneously. Hadri et al. [23] adopted fuzzy PCA to keep the most relevant features of network traffic data, and the experimental results showed that compared with standard PCA, fuzzy PCA performs better on the KDD Cup 99 dataset. Elkhadir et al. [24] applied kernel techniques to PCA for feature reduction and then fed the feature

subset to a k-nearest neighbor (kNN) classifier for network traffic data classification. Mashuri et al. [25] combined PCA with Hotelling's T^2 chart to reduce feature dimensions and computational complexity in intrusion classification while maintaining detection accuracy. Abdulkareem et al. [26] identified the most important features before applying PCA for feature reduction, resulting in a reduced feature set from 36 to 5 while maintaining classification accuracy on the Bot-IoT dataset. Udas et al. [27] constructed a hybrid recurrent neural network (RNN) model, adopting PCA as the feature reduction method. T experimental results showed that the model performed well on the NSL-KDD and UNSW-NB15 datasets in anomaly detection. Paper [28] incorporated probabilistic PCA with a generalized additive model (GAM) for feature reduction and capturing non-linear relationships, thereby enhancing the comprehension of intrusion detection. Zong et al. [5] visualized the datasets in a 3D space and extracted the features using AE and PCA, respectively. The result showed that the features extracted by AE perform better than PCA on three classifiers, namely, decision tree (DT), kNN, and Multi-layer Perceptron (MLP) for intrusion detection.

More and more researchers have been using AE and its variants for feature dimensionality reduction in network intrusion detection due to their excellent performance in capturing non-linear relationships among features. Paper [29] introduced a hybrid model that combined AE and density estimation in anomaly detection. Yousefi-Azar et al. [30] stacked several restricted Boltzmann machine (RBM) blocks to build an AE in the pre-training phase for feature reduction in the field of malware classification and network anomaly detection. Mushtaq et al. [31] adopted a deep AE for dimensionality reduction before using the long short-term memory (LSTM) model for network traffic detection. Basati and Faghih [32] proposed a feature reduction model that comprises two parallel AEs. Besides the traditional AE components, one AE includes eight conventional convolution filters to extract the local information while the other AE incorporates eight dilated convolution filters to extract the surrounding information of network flows. Papers [33] utilized a multi-layer stacked denoising autoencoder (DAE) model in malicious software classification and can handle the case of 0-day or unknown attacks. Lopes et al. [34] implemented a DAE to compress the feature dimensionality, and then used a deep neural network (DNN) classifier for intrusion detection on the CICIDS2018 dataset, achieving above 99.6% on many metrics. An and Cho [35] utilized a variational autoencoder (VAE) model to reduce the noise in the network traffic information, enhancing the performance of network traffic classification. Monshizadeh et al. [36] applied a conditional constraint on the input of a VAE model to extract the discriminative features from the network flows, achieving more efficient feature dimensionality reduction. He et al. [37] combined a conditional Wasserstein VAE and a generative adversarial network (GAN) to reduce feature dimensionality while addressing the issue of class imbalance in multi-classification for intrusion detection scenarios. Zhang et al. [38] employed the SSAE model to reduce dimension and learn the latent representation of the original network traffic flow. Yan and Han [39] constructed a four-layer SSAE model to reduce dimension and then used three basic classifiers to perform comparative experiments, proving the advanced nature of the SSAE in feature reduction. In reference [7], a three-layer SSAE model was used for extracting security situation elements, followed by a recurrent neural network model, achieving an *accuracy* of over 95% on the testing set. Zhang et al. [8] proposed a feature reduction method that combines the Person correlation coefficient with an SSAE, achieving both linear and non-linear dimensionality reduction. The optimal feature set is then fed into an improved Gaussian mixture model. Experimental results showed that the model achieves an *accuracy* of 99.36% on the UNSW-NB15 dataset. Previous related work has indicated that a more efficient representation of data can be achieved by stacking AEs. However, this also entails the need to set more hyperparameters.

B. SI-BASED HYPERPARAMETERS OPTIMIZATION

The extraction of situation elements ultimately serves to classify network traffic. After feature extraction, the appropriate features are fed to a machine learning model. Nevertheless, the performance of various machine learning models, including SVM, kNN, RF, XGBoost, and all deep learning models, relies heavily on the accurate determination of their hyperparameters.

For traditional machine learning models. To enhance SVM classifiers, various SI algorithms such as particle swarm optimization (PSO) [40], genetic algorithm (GA) [41], artificial bee colony (ABC) [42], bat algorithm (BA) [43], physarum network (PN) combined with ant colony optimization (ACO) [44], chaos theory based PSO [45], improved whale optimization algorithm (WOA) [46], gray wolf optimization (GWO) combined with PSO [47], grasshopper optimization algorithm (GOA) [48], and Harris hawks optimization (HHO) combined with PSO [49] were used to optimize the hyperparameters of SVM classifiers, resulting in better improvements over regular SVM on the benchmarked datasets for intrusion detection. Turukmane and Devendiran [10] adopted the mud ring algorithm (MRA) to optimize the hyperparameters of a multi-layer SVM classifier, resulting in better improvement on both the CSE-CIC-IDS 2018 and UNSW-NB15 datasets for multi-class intrusion detection. Canbay and Sagiroglu [50] hybridized GA and kNN to detect attacks, GA was used to select k neighbors from an input sample. Liu et al. [51] utilized an improved arithmetic optimization algorithm (AOA) on the kNN classifier to find the optimal hyperparameters of k neighbors and distance weight. Tahira and Nurdan [52] tuned the hyperparameters of SVM, RF, and kNN on the NSL-KDD dataset using PSO and ABC algorithms, respectively, and finally figured out that kNN outperformed the others in the context of intrusion detection. Assiri [53] utilized GA to optimize

the hyperparameters of the RF classifier, improving the performance in network attack detection compared with the standard RF classifier. Jiang et al. [54] proposed a PSO-XGBoost model, which used PSO to search for optimal hyperparameters of the ensemble classifier named XGBoost. The results showed that PSO-XGBoost performs better than RF, Bagging, and AdaBoost on the NSL-KDD dataset. Zivkovic et al. [55] adopted an improved firefly algorithm (FA) to optimize all six hyperparameters of the XGBoost classifier. The experimental results showed that the proposed method performs best compared to standard XGBoost, PSO-XGBoost, and FA-XGBoost on both the NSL-KDD and UNSW-NB15 datasets.

The above section outlines the application of traditional machine learning models optimized by SI algorithms in the field of intrusion detection. With the ongoing progress in deep learning technology and hardware advancements, an increasing number of scholars are employing SI algorithms to optimize deep learning models for intrusion detection.

For deep learning models. The convolutional neural network (CNN) with hyperparameters optimized by SI excels at capturing spatial features in network flow samples. Kan et al. [56] utilized an adaptive PSO to search a 10D space to find the optimal hyperparameters for the 1D-CNN. In comparison to the other three classifiers, the proposed APSO-CNN model outperforms in terms of *accuracy*, *Haming loss*, and *kappa coefficient* in the context of the Internet of Things (IoT) network intrusion detection. Kilichev and Kim [9] optimized the hyperparameters of 1D-CNN by utilizing GA and PSO in a 9D space, respectively, reaching an *accuracy* exceeding 99% on three benchmark datasets. Ponmalar and Dhanakoti [57] adopted a hybrid whale tabu optimization (HWTO) algorithm to search for the optimal six hyperparameters of CNN, thereby enhancing performance across three benchmark datasets. Kumar et al. [58] proposed a deep residual convolutional neural network (DRCNN) with hyperparameters optimized by an improved gazelle optimization algorithm (IGOA) to attain superior IDS performance. The RNN with hyperparameters optimized by SI excels at capturing temporal information, which is more critical in network flow detection. Deore and Bhosale [59] optimized the hyperparameters of LSTM with a combination of chicken swarm optimization (CSO) and chimp optimization algorithm (ChOA), termed ChCSO, resulting in an *accuracy* of 95.96% on the NSL-KDD dataset and 99.17% on the BoT-IoT dataset, respectively. Awad et al. [60] combined the chaotic butterfly optimization algorithm (CBOA) with the PSO algorithm to enhance the accuracy of LSTM, demonstrating significant improvements compared with standard LSTM on both the NSL-KDD and LITNET-2020 datasets. Ma et al. [61] elevated the classification performance of the gated recurrent units (GRU) RNN model using an improved seagull optimization algorithm (ISOA), resulting in an *accuracy* of up to 98.6% on the NSL-KDD dataset. Combining CNN and RNN allows for the simultaneous capture of spatial and temporal features. Kim and Cho [62] devised a hybrid CNN and LSTM model for data security, using PSO to optimize the nine hyperparameters of the proposed CNN-LSTM. Karthic and Kumar [11] developed an enhanced conditional random field-based CNN-LSTM, optimizing its hyperparameters with an adaptive golden eagle optimization (GEO) algorithm. Experimental results showed that the proposed approach outperforms other popular ML classifiers on the NSL-KDD and UNSW-NB15 datasets. Pustokhina et al. [63] presented a CNN-BiLSTM network with hyperparameters optimized by an improved GA, showcasing effectual improvements on the UNSW-NB15 datasets. Several scholars explore the use of SI algorithm to optimize the hyperparameters in other deep learning models, yielding effective outcomes. Elmasry et al. [64] employed a two-step PSO process to classify attack flows and normal flows. Initially, PSO was utilized to select the feature subset. The selected features were then fed into a deep belief network (DBN) with hyperparameters optimized by a second PSO. Saheed et al. [65] proposed a HAEMPSO model, which utilized an autoencoder for feature extraction. Subsequently, PSO was adopted to optimize the hyperparameters of a self-built deep neural network, achieving competitive results in terms of *detection rate* and *accuracy* on the BoT-IoT dataset for IoT anomaly detection. Cao and Qu [12] adopted a structurally fixed stacked AE with weights and the sparsity parameter optimized by the WOA for feature extraction. The optimal features were then fed into a GRU network for intrusion detection, achieving an *accuracy* of 98.69% on the NSL-KDD dataset. Sekhar et al. [13] adopted the fruitfly optimization algorithm (FOA) to optimize the hidden neurons of a deep AE for feature extraction. The extracted features were subsequently fed into a backpropagation network, resulting in an *accuracy* of 94% on the UNSW-NB15 dataset.

However, there has been limited research on hyperparameter optimization for the SSAE in the context of intrusion detection. Many researchers set these hyperparameters on the basis of their experience. For instance, in reference [7], the authors conducted numerous tests to identify the most suitable hyperparameters, including the number of layers, the number of neurons in each latent layer, and the sparsity parameter for SSAE, achieving better performance. Thus, we will propose using an SI optimization algorithm to select the best hyperparameters for SSAE automatically in this work.

III. METHODOLOGY

This section introduces the SSAE model, the DBO algorithm, and then outlines the overall framework of the proposed model.

A. SSAE NETWORK

AE, illustrated in Fig. 1, is an unsupervised neural network comprising an encoder and a decoder. The former identifies correlations between features, enabling input data dimensionality reduction. Subsequently, the latter reconstructs the original data from the reduced features. The assessment of

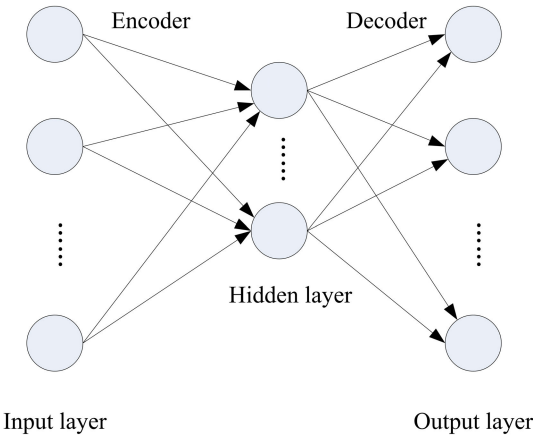

FIGURE 1. Structure of AE.

reduced features involves measuring the reconstruction error between the input and output.

Suppose that the unlabeled input data is x with N examples, the formulas for the encoder phase and decoder phase are as follows:

$$h = f(x) = \sigma_e(W_e x + b_e) \tag{1}$$
$$y = \hat{x} = g(h) = \sigma_d(W_d h + b_d) \tag{2}$$

where h is the output of the hidden layer, σ_e and σ_d are the activation functions, which are chosen to be non-linear functions, such as sigmoid, ReLU, or tanh. W_e is the weight of the encoder, W_d is the weight of the decoder; b_e is the bias of the encoder, and b_d is the bias of the decoder. The output is denoted as y. The error of the AE, commonly referred to as *loss*, is typically quantified using the *mean squared error* (MSE) function:

$$L(W, b) = \frac{1}{N}\sum(y - x)^2 \tag{3}$$

The target of an AE is to make y approximate the input x, in other words, to minimize the constructed error $L(W, b)$.

Sparse AE (SAE) adds a sparsity constraint on AE to suppress the output of most hidden layer neurons, achieving the effect of sparsity and feature dimension reduction. To achieve the above effect, the sparsity parameter ρ is generally taken to be a value near 0 (typically 0.05), which represents the activated proportion of neurons in the hidden layer:

$$\rho = \hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} f_j(x_i) \tag{4}$$

where $f_j(\cdot)$ represents the input of the hidden unit j. Then the KL-divergence is used to penalize $\hat{\rho}_j$ deviating from ρ, the formula is given as follows:

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{5}$$

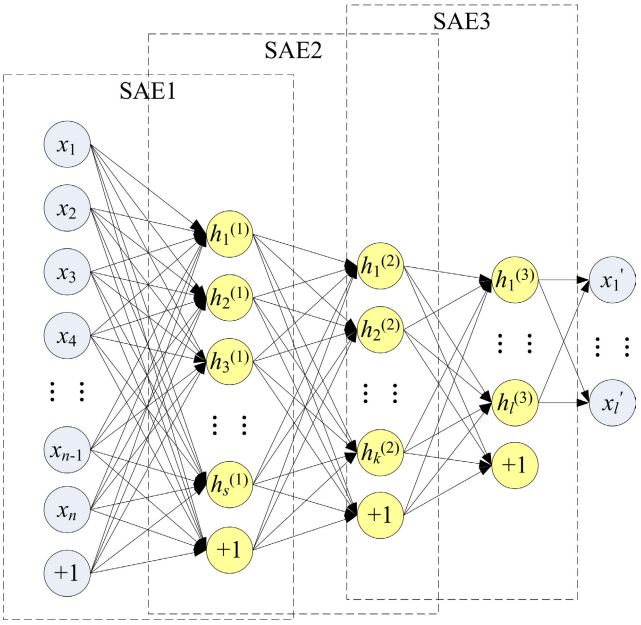

FIGURE 2. A three-layer SSAE model.

The constructed error of SAE can be expressed as

$$L_{sparse}(W, b) = J(W, b) + \beta \sum_{j=1}^{k} KL(\rho \parallel \hat{\rho}_j) \tag{6}$$

Here, $L(W, b)$ is as defined in (3), β is the penalty parameter belonging to (0, 1), and k is the number of neurons in the hidden layer.

If multiple hidden layers are added directly to SAE, it may lead to the loss of certain feature elements. A more commonly used approach is to stack multiple SAE encoding structures, *i.e.* SSAE, where the hidden layer output of the previous SAE becomes the input of the next SAE. The SAEs are trained layer by layer in an unsupervised manner, employing a greedy algorithm approach to minimize the loss between the input and output of each SAE, which helps to reduce the loss rate of feature elements. Fig. 2 illustrates a stacked three-layer SAE model.

B. DBO ALGORITHM

The DBO algorithm was recently proposed by Xue and Shen [14], inspired by the behaviors of dung beetles. As a new swarm intelligence optimization algorithm, DBO is superior to other optimization algorithms on almost all of the CEC-BC-2017 test functions. This superiority arises from its consideration of not only the impact of the global best individual but also that of the global worst individual. In this algorithm, individuals are regarded as dung beetles in a multidimensional search space, and each dung beetle is a potential solution to the optimization problem. The DBO algorithm consists of five steps.

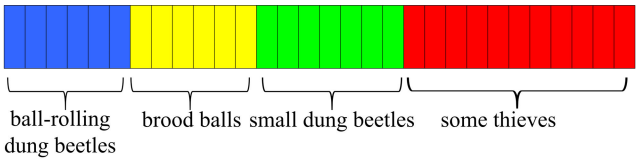

FIGURE 3. The distribution of dung beetles [14].

1) INITIALIZATION

Suppose that n agents are in a d-dimensional space, then the population X is expressed as follows:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \tag{7}$$

All dung beetles should be distributed across the entire solution space initially, to avoid missing out on the optimal solution.

2) CALCULATE FITNESS VALUE

In this stage, the fitness values of all agents are calculated. Thus, the global best dung beetle corresponds to the minimum fitness value, and the fitness function depends on the problem to be solved.

3) UPDATE LOCATIONS

All dung beetles are divided into four types, namely, ball-rolling dung beetle, brood ball, small dung beetle, and thief, which increase the diversity of updated locations. Fig. 3 shows the suggested proportion of each type. Suppose that the number of the initial population is 30, among them, 6 are ball-rolling dung beetles, 6 are brood balls, 7 are small dung beetles, and the rest of them are thieves. Each type updates its position according to different formulas.

a: BALL-ROLLING DUNG BEETLE

There is a 90% possibility that the ball-rolling dung beetles will roll the ball. They update their positions following the formula:

$$x_i(t+1) = x_i(t) + \alpha \times k \times x_i(t-1) + b \times \bigtriangleup x \tag{8}$$

$$\bigtriangleup x = \left| x_i(t) - X^w \right| \tag{9}$$

Here, X^w denotes the position of the global worst dung beetle up to the previous iteration. α is assigned 1 or -1. k and b are constants valued within the interval $(0, 0.2]$ and $(0, 1)$, respectively. $x_i(t)$ means the location of the i-th dung beetle at iteration t.

A 10% possibility that they will dance and update their locations as follows:

$$x_i(t+1) = x_i(t) + \tan(\theta) \left| x_i(t) - x_i(t-1) \right| \tag{10}$$

Here, θ is randomly valued from a uniform distribution over the interval $[0, \pi]$, representing the angle of deflection.

b: BROOD BALL

The dung beetles of brood ball type update their locations as follows:

$$\begin{aligned} x_i(t+1) = X^* &+ b_1 \times (x_i(t) - Lb^*) \\ &+ b_2 \times (x_i(t) - Ub^*) \end{aligned} \tag{11}$$

Here, X^* is the best position among n dung beetles at the current iteration, that is, the local best location. The spawning area is restricted by $\left[Lb^*, Ub^*\right]$. b_1 and b_2 are random vectors of size $1 \times d$, uniformly distributed over the interval $(0, 1)$.

c: SMALL DUNG BEETLE

The small dung beetles update their locations following the formula:

$$\begin{aligned} x_i(t+1) = x_i(t) &+ C_1 \times (x_i(t) - Lb^b) \\ &+ C_2 \times (x_i(t) - Ub^b) \end{aligned} \tag{12}$$

Here, C_1 is randomly valued following a Gaussian distribution, C_2 is a random value over the interval $(0, 1)$, and the optimal foraging area is restricted by $\left[Lb^b, Ub^b\right]$.

d: THIEF

The thieves update their locations following the formula:

$$\begin{aligned} x_i(t+1) = &X^b + S \times g \\ &\times \left(\left| x_i(t) - X^* \right| + \left| x_i(t) - X^b \right| \right) \end{aligned} \tag{13}$$

Here, X^b denotes the best position of the dung beetles until now, that is, the global best location. S represents a constant value, and g is a random vector sampled from a Gaussian distribution with a size of $1 \times d$.

4) JUDGE IF OUT OF BOUNDARY

After the last phase, each dung beetle updates to the new location. However, some agents may be outside the boundary of the solution space. In this case, these agents will be set to the value of the boundary.

5) LOOP OR TERMINATE

Repeat from the beginning of step 2 until the termination condition is met.

C. DBO-SSAE MODEL

Considering the training effect and algorithm time complexity comprehensively, a three-layer stacked SSAE structure is designed in this paper, as shown in Fig. 2. With different hyperparameters for SSAE, the performance of extracting network situation elements varies, leading to different classification effects when fed into the network intrusion detection model. Lin et al. [7] designed 12 comparison experiments to confirm the optimal combination of SSAE hyperparameters, containing the number of layers, the number of neurons in each latent layer, and the sparsity parameter. As an improvement, we use the DBO algorithm for this task.

Each dung beetle individual, acting on the SSAE model, is represented by a four-dimensional vector containing the sparsity parameter and the number of hidden neurons for all three layers. The formula for the location of the i-th dung beetle is:

$$x_i = [\rho_i,\ l1_i,\ l2_i,\ l3_i] \tag{14}$$

where ρ_i is the sparsity parameter, $l1_i$, $l2_i$, and $l3_i$ represent the number of neurons in the first, second, and third hidden layers, respectively.

The fitness function is set to the quadratic sum of the loss of SSAE's every layer, rather than the sum of losses, to prevent the loss of a certain layer from being too large and other layers too small. Consider a scenario where the loss of one layer is large, while the losses of the other two layers are close to zero. In this case, the sum of the losses for the three layers might be small, but it could still result in significant feature loss. The utilization of squared error helps avoid this situation, ensuring that the error for each layer remains relatively small, contributing to a small sum of squared errors. The loss of SSAE corresponding to the i-th dung beetle is represented as follows:

$$f(x_i) = Loss_{all} = \sum_{j=1}^{3} Loss_{ij}^2 \tag{15}$$

Here, $Loss_{ij}$ represents the MSE loss of the j-th layer in SSAE corresponding to the i-th dung beetle. The current best dung beetle corresponds to the minimum fitness value among $f(x_i)$. The global best dung beetle is the individual with the minimum fitness value among all the individuals up to now. After initializing the dung beetles and calculating their initial fitness, we update the locations of the agents via (8)–(13) according to their types. Subsequently, we feed the location of the agent to the SSAE model and use the vector as the hyperparameters of SSAE. Next, the fitness value is calculated via (15), and the current and global best dung beetles are updated. Repeat this process until the termination condition is met. Finally, output the global best dung beetle and the feature subset. The flowchart of the DBO-SSAE process is illustrated in Fig. 4.

IV. EXPERIMENT AND RESULTS

The work presented in this article is divided into two main parts. The first part focuses on obtaining the optimal hyperparameters of the SSAE model, as detailed in Section III. In the second part, the feature subset resulting from dimensionality reduction using SSAE is fed into the machine learning model. The quality of these features is assessed by evaluating their classification performance using various classifies on the network benchmark dataset, as depicted in Fig. 5.

We implemented the experiments on a machine configured with Intel(R) Xeon(R) Gold 5318Y CPU, NVIDIA GeForce RTX 4090 GPU, Ubuntu 18.04.1 LTS operation system, Python 3.8, and Pycharm Community 2020.3. The model was set up with Tensorflow-gpu 2.5.0, Keras 2.4.3, Pandas, and scikit-learn modules using Python programming language.

A. EXPERIMENTAL PROCEDURE

1) DATASET DESCRIPTION

Moustafa et al. [20] generated UNSW-NB15 from real-world network flows, establishing it as one of the new benchmark datasets in the field of network intrusion detection. UNSW-NB15 is divided into a training set with 175,341 samples and a testing set with 82,332 samples. These samples are categorized into normal traffic flows and attack traffic flows, as outlined in Fig. 6. The distribution of the training set is as follows: 31.9% for *Normal* and 68.1% for *Attack*. Meanwhile, the distribution of the testing set is as follows: 44.9% for *Normal* and 55.1% for *Attack*. Therefore, the UNSW-NB15 dataset is suitable for binary classification without any issues of unbalanced samples. Each sample of UNSW-NB15 consists of 43 features and 2 labels.

2) DATASET PREPROCESSING

Since SSAE can filter features automatically, it is only necessary to modify features to a format that can be processed by classifiers during the data preprocessing stage. Three features of UNSW-NB15, such as *proto*, *service*, and *state*, are non-numeric. Therefore, they need to be transformed into numerical characteristics before they can be fed into most classifiers. In this paper, we adopt *one-hot* encoding for this purpose. We also removed the *id* since it does not contribute to the classification. Since this paper focuses solely on binary classification, the *attack_cat* was removed, as it provides detailed descriptions of traffic categories. Furthermore, we utilized min-max normalization to scale the features within the range of [0, 1]. After feature scaling, these features were transformed as follows:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{16}$$

After preprocessing, all samples in the UNSW-NB15 dataset consist of 196 features and 1 label with two values, where 0 represents normal and 1 for attack.

3) DBO-SSAE TRAINING

At this stage, we employed the DBO algorithm to generate the hyperparameters of each SSAE model. Subsequently, these SSAE models were individually trained. During training, the weights and biases of each layer in the SSAE were adjusted to minimize the MSE loss, and this process followed a greedy layer-wise training method. The parameters of DBO are shown in Table 1. The initial size of the dung beetle swarm was $N = 30$, among them, 6 were ball-rolling dung beetles, 6 were brood balls, 7 were small dung beetles, and the remaining 11 were thieves. The maximum number of iterations was $t = 20$ (involving 630 calls to the fitness function, including the initialization). The hyperparameter ranges of the SSAE were determined through a combination of experiments and a comprehensive literature review, as outlined in Table 2. After DBO-SSAE training, the feature dimension was greatly reduced.

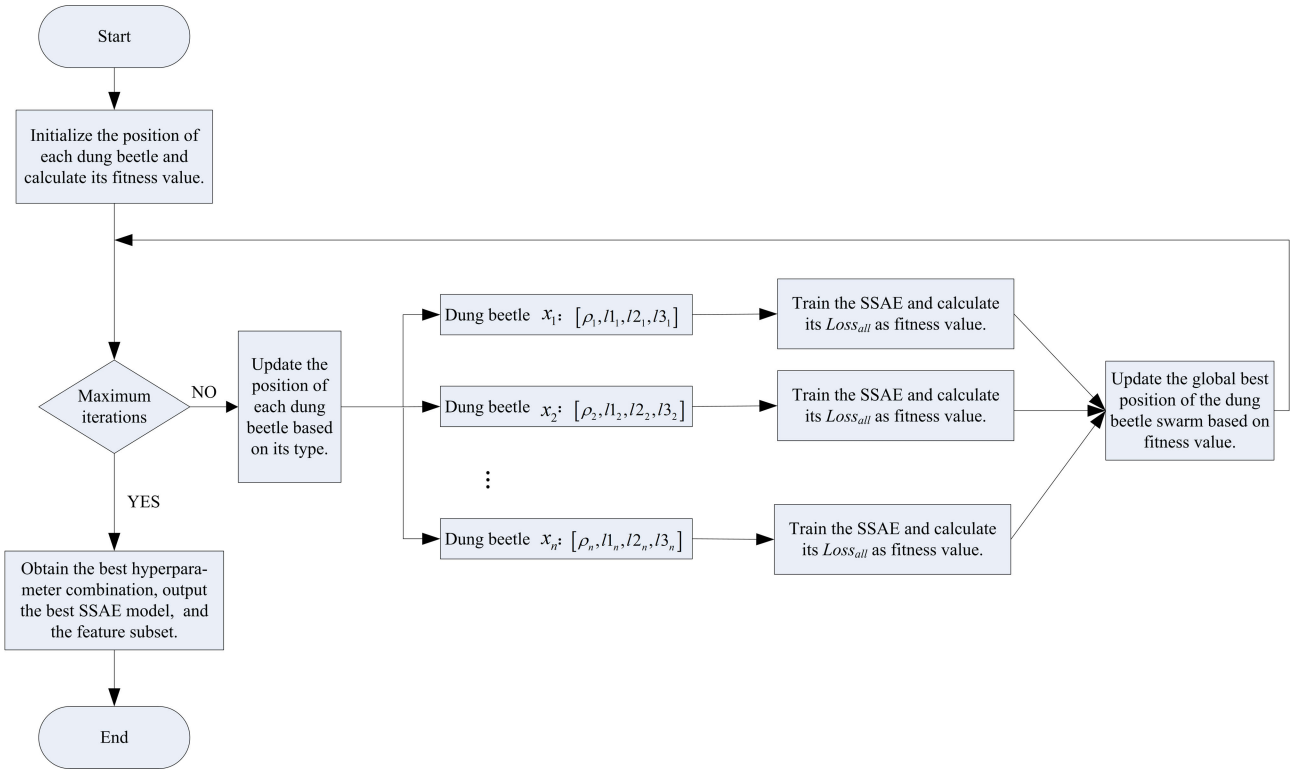

FIGURE 4. Flowchart of proposed DBO-SSAE.

TABLE 1. Parameters of DBO.

Population size	Iterative times	Dimension of search	k	b
30	20	4	0.1	0.3

TABLE 2. Ranges of SSAE's hyperparameters.

ρ	$l1$	$l2$	$l3$
[0.01, 0.1]	[101, 160]	[61, 100]	[20, 60]

4) CLASSIFIER TRAINING

The anomaly traffic detection in this paper can be regarded as a binary classification task. The classifier was trained using a training set consisting of features with lower dimensions and one label, and its performance was verified by a testing set.

Additionally, to assess the effectiveness of feature extraction by the DBO-SSAE model, various classifiers were trained based on features extracted by PCA, features extracted by the SSAE model with hyperparameters set to [0.04, 128, 32, 32] as confirmed in [7], and features extracted by the DBO-SSAE model.

Six different classifiers, namely, DT, logic regression (LR), RF, kNN, LSTM, and BiLSTM were employed to assess the performance of the three feature extraction methods above on the UNSW-NB15 dataset. The *random_state* in both DT and RF was set to 10; the k in kNN was set to 3. The core hyperparameters of LSTM and BiLSTM are listed in Table 3. The structure of LSTM was simple: a single LSTM layer, a fully connected layer, and an output layer. By contrast, the structure of BiLSTM was more complex: two BiLSTM layers, the first of whose *return_sequence* was set to True, followed by a fully connected layer and an output layer. Besides, to take advantage of the time serial nature of network flow, the *time_step* of BiLSTM was set to 8.

TABLE 3. Hyperparameters of LSTM and BiLSTM.

LSTM	BiLSTM
Dense(32, ReLU)	BiLSTM(24, tanh)
LSTM(16, tanh)	Dropout(0.5)
Dropout(0.3)	BiLSTM(12, tanh)
Dense(10, ReLU)	Dropout(0.5)
Dense(1, sigmoid)	Dense(6, ReLU)
—-	Dense(1, sigmoid)
Adam	Adam

5) EVALUATION

In this paper, we use four metrics to evaluate the performance of the classification. The most commonly used performance measure for a binary classifier is *accuracy*. High accuracy indicates that the model possesses a strong predictive ability. Furthermore, for intrusion detection tasks, the *false negative rate* (*FNR*) and *false positive rate* (*FPR*) are also significant and should be as small as possible. If the *FNR* is 0, it means that the classifier can detect all the attack traffic, while an *FPR* of 0 means that the classifier does not misjudge any normal traffic as attack traffic. Nevertheless, there are instances where we must strike a balance among *accuracy*,

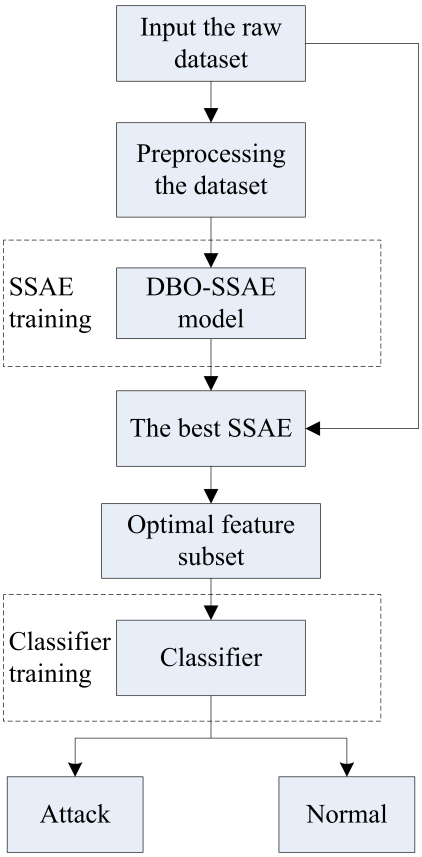

FIGURE 5. The overall framework of the experiment.

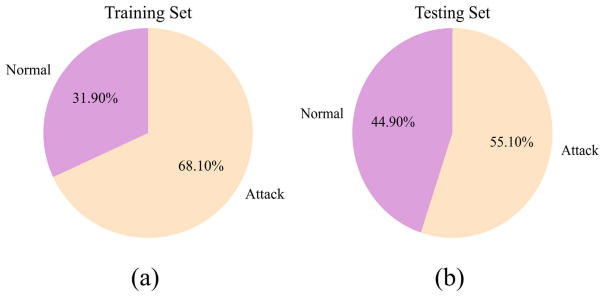

FIGURE 6. The distribution of UNSW-NB15, (a) Training set distribution, (b) Testing set distribution.

FNR, and *FPR*. In such cases, using the F_1-*measure* becomes essential as it provides a comprehensive measure of binary classification tasks, and the F_1-*measure* should be as high as possible.

These metrics are calculated based on the confusion matrix as depicted in Table 4, and the formulas are as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

$$FNR = \frac{FN}{TP + FN} \tag{18}$$

$$FPR = \frac{FP}{FP + TN} \tag{19}$$

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{20}$$

TABLE 4. Confusion matrix.

Reality	Predict	
	Attack	Normal
Attack	TP	TN
Normal	FP	FN

TABLE 5. Top 3 of the best agents.

item	1	2	3
Fitness value	0.01543	0.016548	0.01663
ρ	0.0104065	0.0103366	0.010546
$l1$	159	155	139
$l2$	100	91	96
$l3$	57	42	51
Call	277*th*	457*th*	417*th*

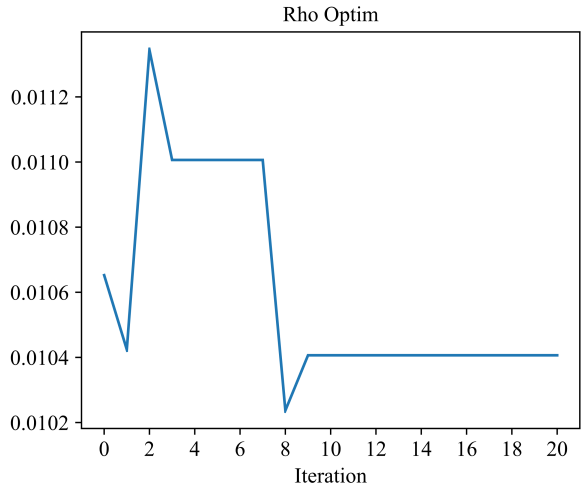

FIGURE 7. The variation of the sparsity parameter ρ.

6) EXPERIMENTAL RESULTS

After 630 calls to the fitness function, the top 3 smallest fitness values were 0.01543, 0.01663, and 0.016548, respectively, corresponding to the positions of the dung beetles listed in Table 5. The best combination of the SSAE hyperparameters was identified at the 277*th* function call. Fig. 7 to Fig. 10 show the variation of the global best hyperparameters in 20 iterations.

The results of the UNSW-NB15 testing set are presented in Table 6. To demonstrate performance differences, the columns of *features via PCA* correspond to the outcomes of classifiers using features extracted by PCA ($n\_components = 57$, $random\_state = 10$). The columns of *features via SSAE* correspond to the outcomes of classifiers using features extracted by the SSAE with manually tuned hyperparameters set to [0.04, 128, 32, 32]. The columns labeled *features via DBO-SSAE* correspond to the results of classifiers that utilize the features extracted by the proposed DBO-SSAE model.

Fig. 7 to Fig. 10 show that after 9 iterations of DBO, the hyperparameters of the SSAE model finally stabilize at [0.010406519792, 159, 100, 57] with the minimum fitness equal to 0.01543. From these figures we can observe that the DBO algorithm was able to find the best hyperparameters

TABLE 6. Comparison with the other two feature extraction methods on the testing set of UNSW-NB15.

Classifier	Features via PCA				Features via SSAE				Features via DBO-SSAE			
Metric	*Accuracy*	*FNR*	*FPR*	F_1	*Accuracy*	*FNR*	*FPR*	F_1	*Accuracy*	*FNR*	*FPR*	F_1
DT	84.77%	26.91%	5.7%	87.2%	84.91%	27.18%	5.22%	87.4%	**85.19%**	**26.99%**	**4.87%**	**87.6%**
RF	84.9%	30.94%	2.18%	87.7%	84.38%	32.23%	2.06%	87.3%	**85.29%**	**30.71%**	**1.64%**	**88.04%**
*k*NN	85.17%	28.17%	3.94%	87.7%	84.48%	29.01%	4.51%	87.13%	**85.64%**	**27.92%**	**3.29%**	**88.12%**
LR	80.6%	40.25%	2.39%	84.7%	80.32%	40.71%	2.52%	84.51%	**82.52%**	**36.99%**	**1.55%**	**86.12%**
LSTM	88.15%	24.29%	**1.68%**	90.13%	88.89%	20.69%	3.29%	90.55%	**90.37%**	**17.72%**	3.02%	**91.7%**
BiLSTM	97.8%	4.11%	0.66%	98.03%	98.01%	3.61%	0.68%	98.22%	**98.84%**	**1.86%**	**0.60%**	**98.96%**

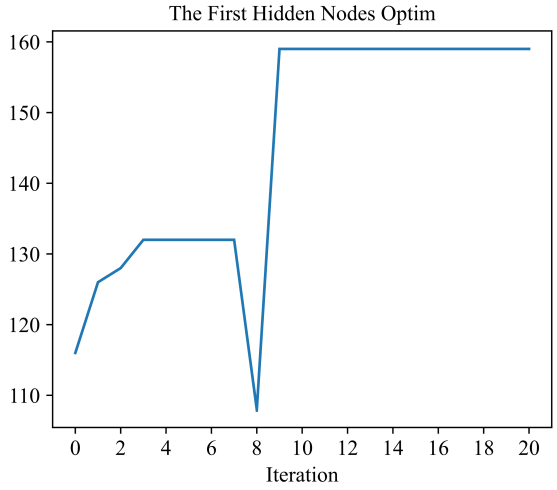

FIGURE 8. The variation of the number of neurons in the first layer.

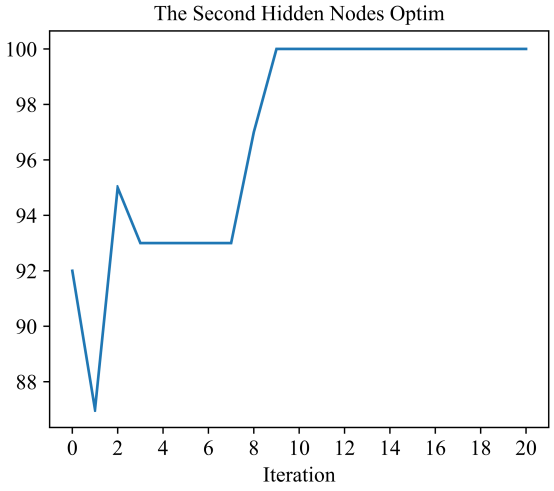

FIGURE 9. The variation of the number of neurons in the second layer.

within 9 iterations, indicating its fast convergence speed. Even after 9 more iterations, the randomness of the DBO agents remained significant, for example, the variation range of ρ reached a maximum of approximately 0.095 and a minimum of approximately 0.01004. This implies that the DBO algorithm has the advantage of being easy to escape from the local optimal.

In Table 6, we observe that the features extracted by SSAE with manually tuned hyperparameters and PCA performed well across all six classification algorithms, achieving an

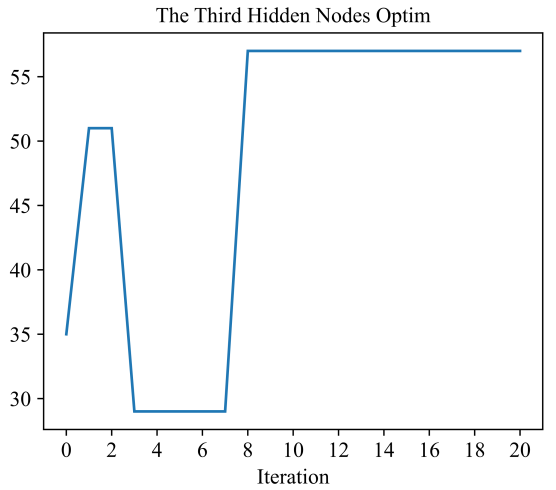

FIGURE 10. The variation of the number of neurons in the third layer.

F_1-*measure* that exceeds 84%, respectively. Notably, when using the BiLSTM, the F_1-*measure* reached 98.22% and 98.03%, respectively. However, the features extracted by the DBO-SSAE proposed in this paper outperformed others in almost all metric terms for all classifiers: higher *accuracy*, higher F_1-*measure*, lower *FNR*, and lower *FPR*. This indicates that the proposed feature extraction method has stronger generalization capabilities than the other two feature extraction methods. The exception was observed in the case of *FPR* when using LSTM with features extracted by PCA. Nevertheless, when the *accuracy* is lower but the *FPR* is either lower, we use the F_1-*measure* as the composite indicator. The F_1-*measure* of features extracted by DBO-SSAE was 1.5% higher than that of features extracted by PCA. Furthermore, the highest F_1-*measure* of the same classifier was obtained from the features extracted by DBO-SSAE. It can also be seen that the LSTM and BiLSTM models performed better than the four other traditional classifiers. This is because network traffic records can be viewed as time series data, making it more appropriate to use the RNN model, such as LSTM and BiLSTM, for classification. In particular, when BiLSTM was used after utilizing the features extracted by DBO-SSAE, we achieved an *accuracy* and F_1-*measure* of 98.84% and 98.96%, with *FNR* and *FPR* at only 1.86% and 0.6%, respectively. Fig. 11 shows the confusion matrix by BiLSTM using the features extracted through DBO-SSAE on the UNSW-NB15 testing set.

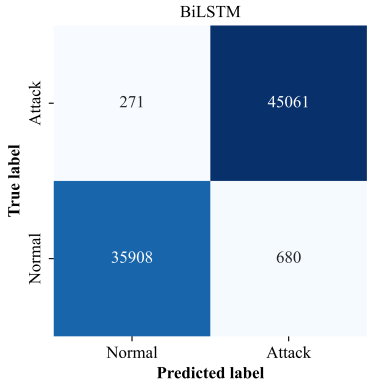

FIGURE 11. Confusion matrix of DBO-SSAE-BiLSTM on UNSW-NB15.

TABLE 7. Comparison with other advanced approaches for binary classification on UNSW-NB15.

Method	F_1	Year
SSAE-BiGRU [7]	98.33%	2019
FA-AE-BNP [13]	92.84%	2021
SPIDER [27]	86.12%	2022
GA-1D-CNN [9]]	98%	2023
SIGMOD [8]	96.07%	2023
GRU-BWFA [67]	98.79%	2023
MF-Net [66]	91.4%	2024
Proposed DBO-SSAE-BiLSTM	**98.96%**	2024

Table 7 presents a comparison between the DBO-SSAE-BiLSTM model and recently advanced approaches. To ensure fairness, we utilized the comprehensive F_1-*measure* metric on the same dataset for binary classification. Among these approaches, SSAE-BiGRU [7], FA-AE-BNP [13], SIGMOD [8], and MF-Net [66] adopted AEs for feature reduction. SSAE-BiGRU [7], SPIDER [27], GRU-BWFA [67], and MF-Net [66] utilized RNNs as classifiers; FA-AE-BNP [13], GA-1D-CNN [9], and GRU-BWFA [67] leveraged SI optimization algorithms to optimize hyperparameters of the deep learning models. It is evident that the proposed DBO-SSAE-BiLSTM achieves the highest F_1-*measure* of 98.96% among the listed methods, underscoring its advantage in intrusion detection.

V. DISCCUSION

The presented experimental results underscore the effectiveness of the DBO-SSAE model. Applying DBO to optimize the hyperparameters of SSAE in a network situation element extraction scenario, we could obtain an optimal feature set in comparison to other feature extraction methods. *Accuracy*, F_1-*measure*, *FNR*, and *FPR* were compared by inputting the features extracted by DBO-SSAE, SSAE with manually tuned hyperparameters, and PCA into several classifiers, indicating the stronger feature extraction ability of DBO-SSAE. Furthermore, when combined with a BiLSTM network, the DBO-SSAE enhanced the performance of intrusion detection.

SSAE has been widely adopted for feature extraction in previous literature. Although Reference [7] invested time in tuning the hyperparameters of SSAE through numerous experiments, the representation ability of the resulting features is still lower than that of our proposed DBO-SSAE method. This difference can be attributed to the strong and fast search capability of the DBO algorithm, enabling it to automatically find the optimal hyperparameters for SSAE. Reference [13] utilized an SI algorithm FA to optimize the neuron number of each hidden layer in the deep AE. However, the F_1-*measure* still fell short by 6.12% compared to our proposed method. This disparity may be attributed to the superior nature of DBO over FA. Unlike FA, DBO considers not only the best individual but also the worst individual, making it less prone to being trapped in a local optimum. Moreover, we used the quadratic sum of losses in each layer as the fitness function, rather than the sum of losses, to obtain the optimal feature set with the least loss. Furthermore, we applied SSAE with sparsity constraint to mitigate overfitting issues associated with deep AE, thereby enhancing the classification performance on the testing set.

Despite the experimental results are encouraging, it is crucial to acknowledge certain limitations of our proposed method.

A. LONG TRAINING TIME OF DBO-SSAE

In this paper, the DBO algorithm is employed to optimize the hyperparameters of SSAE. However, limited by the inherent characteristics of SI, it requires hundreds of function calls to achieve the goal. In the experiments, approximately 58 seconds are needed for each training instance of a three-layer SSAE. Across the 630 training instances studied in this study, the total training time exceeds 30,000 seconds. Although the training time of DBO-SSAE is significantly higher compared to PCA, the resulting features are superior, making the investment worthwhile.

B. NUMEROUS PARAMETERS OF DBO ITSELF

The variety of individual types and locations of DBO makes it easy to escape from a local optimal, but it also implies that more parameters need to be set. In this paper, the default parameter settings from reference [14] are used. If a better feature subset is to be obtained, these parameters should be set prudently.

VI. CONCLUSION

In this paper, we proposed a novel method for automatically extracting network security situation elements, named DBO-SSAE. This method leverages the strengths of the DBO algorithm to automatically optimize the hyperparameters of the SSAE network without manual tuning, thereby achieving effective and robust generalized feature extraction for network security situations. Experimental results demonstrated that DBO-SSAE outperforms other feature extraction methods in the realm of network security. In particular, when combined with the BiLSTM RNN, our research outperformed

many other works on the UNSW-NB15 dataset in terms of F_1-*measure*.

The successful application of DBO-SSAE extends beyond its immediate contributions to network intrusion detection. Its ability to autonomously extract salient features from network data holds promise for various applications within the broader field of machine learning. Furthermore, the integration of the DBO algorithm for hyperparameter tuning in the SSAE introduces a novel perspective to optimization algorithms in the realm of deep learning.

However, this study has several limitations. Firstly, training the DBO-SSAE requires a significant amount of time. Considering the success of the attention mechanism in the natural language processing field, we will focus on further enhancing SSAE by adding the attention mechanism to address this issue. Secondly, DBO has numerous parameters to set. To overcome this, we will hybridize it with another SI algorithm with fewer parameters, such as PSO, to find the optimal parameters for DBO. Furthermore, we will explore how to extract features from raw network flows using DBO-SSAE and then apply them in real-world intrusion detection scenarios.

REFERENCES

[1] CNNIC. (2023). *Statistical Report on Internet Development in China*. Accessed: Dec. 23, 2023. [Online]. Available: https://cnnic.cn/n4/2023/0828/c199-10830.html

[2] G. Kou, S. Wang, and G. Tang, "Research on key technologies of network security situational awareness for attack tracking prediction," *Chin. J. Electron.*, vol. 28, no. 1, pp. 162–171, Jan. 2019.

[3] G.-H. Yan. *What Is Network Security Situational Awareness?*. Accessed: Dec. 23, 2023. [Online]. Available: https://support.huawei.com/enterprise/en/doc/EDOC1100202622

[4] J. Zhao, X. Jiang, Z. Yan, and W. Pedrycz, "Network traffic classification for data fusion: A survey," *Inf. Fusion*, vol. 72, pp. 22–47, Aug. 2021.

[5] W. Zong, Y.-W. Chow, and W. Susilo, "Dimensionality reduction and visualization of network intrusion detection data," in *Proc. ACISP*, Christchurch, New Zealand, 2019, pp. 441–455.

[6] A. Ng. *Sparse Autoencoder*. Accessed: Dec. 23, 2023. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs294a/cs294a.1104/_sparse Autoencoder.pdf

[7] Y. Lin, J. Wang, Y. Tu, L. Chen, and Z. Dou, "Time-related network intrusion detection model: A deep learning method," in *Proc. GLOBECOM*, Waikoloa, HI, USA, 2019, pp. 1–6.

[8] T. Zhang, W. Chen, Y. Liu, and L. Wu, "An intrusion detection method based on stacked sparse autoencoder and improved Gaussian mixture model," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103144.

[9] D. Kilichev and W. Kim, "Hyperparameter optimization for 1D-CNN-based network intrusion detection using GA and PSO," *Mathematics*, vol. 11, no. 17, p. 3724, Aug. 2023.

[10] A. V. Turukmane and R. Devendiran, "M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning," *Comput. Secur.*, vol. 137, Feb. 2024, Art. no. 103587.

[11] S. Karthic and S. M. Kumar, "Hybrid optimized deep neural network with enhanced conditional random field based intrusion detection on wireless sensor network," *Neural Process. Lett.*, vol. 55, no. 1, pp. 459–479, Feb. 2023.

[12] X. Cao and H. Qu, "Deep feature extraction via sparse autoencoder for intrusion detection system," *Comput. Sci. Inf. Technol.*, vol. 10, pp. 61–73, Jul. 2020.

[13] R. Sekhar, K. Sasirekha, P. S. Raja, and K. Thangavel, "A novel GPU based intrusion detection system using deep autoencoder with fruitfly optimization," *Social Netw. Appl. Sci.*, vol. 3, no. 6, Jun. 2021, Art. no. 594.

[14] J. Xue and B. Shen, "Dung beetle optimizer: A new meta-heuristic algorithm for global optimization," *J. Supercomput.*, vol. 79, no. 7, pp. 7305–7336, May 2023.

[15] X. Xu and X. Wang, "An adaptive network intrusion detection method based on PCA and support vector machines," in *Proc. ADMA*, Wuhan, China, 2005, pp. 696–703.

[16] I. S. Thaseen and Ch. A. Kumar, "Intrusion detection model using fusion of PCA and optimized SVM," in *Proc. Int. Conf. Contemp. Comput. Informat. (IC3I)*, Mysore, India, Nov. 2014, pp. 879–884.

[17] A. Mishra, A. M. K. Cheng, and Y. Zhang, "Intrusion detection using principal component analysis and support vector machines," in *Proc. IEEE 16th Int. Conf. Control Autom. (ICCA)*, Singapore, Oct. 2020, pp. 907–912.

[18] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, "KDD cup 99 data sets: A perspective on the role of data sets in network intrusion detection research," *Computer*, vol. 52, no. 2, pp. 41–51, Feb. 2019.

[19] M. S. Pervez and D. Md. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *Proc. 8th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, Dhaka, Bangladesh, Dec. 2014, pp. 1–6.

[20] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. MilCIS*, Canberra, SYD, Australia, Nov. 2015, pp. 1–6.

[21] B. Zhang, Z.-Y. Liu, Y.-G. Jia, J.-D. Ren, and X.-L. Zhao, "Network intrusion detection method based on PCA and Bayes algorithm," *Secur. Commun. Netw.*, vol. 2018, pp. 1914980–1914990, Nov. 2018.

[22] S. Waskle, L. Parashar, and U. Singh, "Intrusion detection system using PCA with random forest approach," in *Proc. Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Coimbatore, India, 2020, pp. 803–808.

[23] A. Hadri, K. Chougdali, and R. Touahni, "Intrusion detection system using PCA and fuzzy PCA techniques," in *Proc. Int. Conf. Adv. Commun. Syst. Inf. Secur. (ACOSIS)*, Marrakesh, Morocco, Oct. 2016, pp. 1–7.

[24] Z. Elkhadir, K. Chougdali, and M. Benattou, "Intrusion detection system using PCA and kernel PCA methods," in *Proc. MedICT*, Saidia, Morocco 2015, pp. 489–497.

[25] M. Mashuri, M. Ahsan, M. H. Lee, D. D. Prastyo, and Wibawati, "PCA-based Hotelling's T^2 chart with fast minimum covariance determinant (FMCD) estimator and kernel density estimation (KDE) for network intrusion detection," *Comput. Ind. Eng.*, vol. 158, pp. 107447–107456, Jun. 2021.

[26] S. A. Abdulkareem, C. H. Foh, F. Carrez, and K. Moessner, "FI-PCA for IoT network intrusion detection," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Shenzhen, China, Jul. 2022, pp. 1–6.

[27] P. B. Udas, M. E. Karim, and K. S. Roy, "SPIDER: A shallow PCA based network intrusion detection system with enhanced recurrent neural networks," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 10, pp. 10246–10272, Nov. 2022.

[28] A. Singh, J. Nagar, J. Amutha, and S. Sharma, "P^2CA-GAM-ID: Coupling of probabilistic principal components analysis with generalised additive model to predict the k-barriers for intrusion detection," *Eng. Appl. Artif. Intell.*, vol. 126, pp. 107137–107149, Nov. 2023.

[29] V.-L. Cao, M. Nicolau, and J. McDermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *Proc. PPSN*, Edinburgh, U.K., 2016, pp. 717–726.

[30] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 3854–3861.

[31] E. Mushtaq, A. Zameer, M. Umer, and A. A. Abbasi, "A two-stage intrusion detection system with auto-encoder and LSTMs," *Appl. Soft Comput.*, vol. 121, May 2022, Art. no. 108768.

[32] A. Basati and M. M. Faghih, "PDAE: Efficient network intrusion detection in IoT using parallel deep auto-encoders," *Inf. Sci.*, vol. 598, pp. 57–74, Jun. 2022.

[33] Y. Wang, W. Cai, and P. Wei, "A deep learning approach for detecting malicious Javascript code," *Secur. Commun. Netw.*, vol. 9, no. 11, pp. 1520–1534, Jul. 2016.

[34] I. O. Lopes, D. Zou, I. H. Abdulqadder, F. A. Ruambo, B. Yuan, and H. Jin, "Effective network intrusion detection via representation learning: A denoising AutoEncoder approach," *Comput. Commun.*, vol. 194, pp. 55–65, Oct. 2022.

[35] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture IE*, vol. 2, no. 1, pp. 1–18, 2015.

[36] M. Monshizadeh, V. Khatri, M. Gamdou, R. Kantola, and Z. Yan, "Improving data generalization with variational autoencoders for network traffic anomaly detection," *IEEE Access*, vol. 9, pp. 56893–56907, 2021.

[37] J. He, X. Wang, Y. Song, Q. Xiang, and C. Chen, "Network intrusion detection based on conditional Wasserstein variational autoencoder with generative adversarial network and one-dimensional convolutional neural networks," *Appl. Intell.*, vol. 53, pp. 12416–12436, May 2023.
[38] B. Zhang, Y. Yu, and J. Li, "Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas, MO, USA, May 2018, pp. 1–6.
[39] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.
[40] J. Wang, H. Xu, R.-R. Ren, and T.-H. Li, "A real-time intrusion detection system based on PSO-SVM," in *Proc. IWISA*, Qingdao, China, 2009, pp. 319–321.
[41] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Appl. Soft Comput.*, vol. 18, pp. 178–184, May 2014.
[42] A.-C. Enache and V. V. Patriciu, "Intrusions detection based on support vector machine optimized with swarm intelligence," in *Proc. IEEE 9th IEEE Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, Timisoara, Romania, May 2014, pp. 153–158.
[43] A.-C. Enache and V. Sgârciu, "Anomaly intrusions detection based on support vector machines with an improved bat algorithm," in *Proc. 20th Int. Conf. Control Syst. Comput. Sci.*, Bucharest, Romania, May 2015, pp. 317–321.
[44] H. Qu, S. Jian, X. Tang, and P. Wang, "Hybrid computational intelligent methods incorporating into network intrusion detection," *J. Comput. Theor. Nanoscience*, vol. 12, no. 12, pp. 5492–5496, Dec. 2015.
[45] Q. Yang, H. Fu, and T. Zhu, "An optimization method for parameters of SVM in network intrusion detection system," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Washington, DC, USA, May 2016, pp. 136–142.
[46] D. Wang and G. Xu, "Research on the detection of network intrusion prevention with SVM based optimization algorithm," *Informatica*, vol. 44, no. 2, pp. 269–273, Jun. 2020.
[47] K. Li, Y. Zhang, and S. Wang, "An intrusion detection system based on PSO-GWO hybrid optimized support vector machine," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Shenzhen, China, Jul. 2021, pp. 1–7.
[48] S. Dwivedi, M. Vardhan, and S. Tripathi, "Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection," *Cluster Comput.*, vol. 24, no. 3, pp. 1881–1900, Sep. 2021.
[49] S. Sokkalingam and R. Ramakrishnan, "An intelligent intrusion detection system for distributed denial of service attacks: A support vector machine with hybrid optimization algorithm based approach," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 27, pp. 7334–7351, Dec. 2022.
[50] Y. Canbay and S. Sagiroglu, "A hybrid method for intrusion detection," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 2015, pp. 156–161.
[51] G. Liu, H. Zhao, F. Fan, G. Liu, Q. Xu, and S. Nazir, "An enhanced intrusion detection model based on improved kNN in WSNs," *Sensors*, vol. 22, no. 4, p. 1407, Feb. 2022.
[52] K. Tahira and B. Nurdan, "Network intrusion detection using optimized machine learning algorithms," *Eur. J. Sci. Technol.*, vol. 25, pp. 463–474, Jun. 2021.
[53] A. Assiri, "Anomaly classification using genetic algorithm-based random forest model for network attack detection," *Comput., Mater. Continua*, vol. 66, no. 1, pp. 767–778, 2020.
[54] H. Jiang, Z. He, G. Ye, and H. Zhang, "Network intrusion detection based on PSO-Xgboost model," *IEEE Access*, vol. 8, pp. 58392–58401, 2020.
[55] M. Zivkovic, M. Tair, V. K, N. Bacanin, Š. Hubálovský, and P. Trojovský, "Novel hybrid firefly algorithm: An application to enhance XGBoost tuning for intrusion detection classification," *PeerJ Comput. Sci.*, vol. 8, p. e956, Apr. 2022.
[56] X. Kan, Y. Fan, Z. Fang, L. Cao, N. N. Xiong, D. Yang, and X. Li, "A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network," *Inf. Sci.*, vol. 568, pp. 147–162, Aug. 2021.
[57] A. Ponmalar and V. Dhanakoti, "Hybrid whale Tabu algorithm optimized convolutional neural network architecture for intrusion detection in big data," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 19, pp. 7038–7052, Aug. 2022.
[58] G. S. C. Kumar, R. K. Kumar, K. P. V. Kumar, N. R. Sai, and M. Brahmaiah, "Deep residual convolutional neural network: An efficient technique for intrusion detection system," *Exp. Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 121912.
[59] B. Deore and S. Bhosale, "Hybrid optimization enabled robust CNN-LSTM technique for network intrusion detection," *IEEE Access*, vol. 10, pp. 65611–65622, 2022.
[60] A. A. Awad, A. F. Ali, and T. Gaber, "An improved long short term memory network for intrusion detection," *PLoS ONE*, vol. 18, pp. 1–44, Aug. 2023.
[61] S. Ma, C.-Z. Wang, A. Liu, Y. Zhang, J.-F. Wang, Y.-G. Chang, and J. Yang, "Improved seagull optimization algorithm to optimize neural networks with gated recurrent units for network intrusion detection," in *Proc. IDAACS*, Cracow, Poland, 2021, pp. 100–104.
[62] T.-Y. Kim and S.-B. Cho, "Optimizing CNN-LSTM neural networks with PSO for anomalous query access control," *Neurocomputing*, vol. 456, pp. 666–677, Oct. 2021.
[63] I. V. Pustokhina, D. A. Pustokhin, E. L. Lydia, P. Garg, A. Kadian, and K. Shankar, "Hyperparameter search based convolution neural network with bi-LSTM model for intrusion detection system in multimedia big data environment," *Multimedia Tools Appl.*, vol. 81, no. 24, pp. 34951–34968, Oct. 2022.
[64] W. Elmasry, A. Akbulut, and A. H. Zaim, "Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic," *Comput. Netw.*, vol. 168, Feb. 2020, Art. no. 107042.
[65] Y. K. Saheed, A. A. Usman, F. D. Sukat, and M. Abdulrahman, "A novel hybrid autoencoder and modified particle swarm optimization feature selection for intrusion detection in the Internet of Things network," *Frontiers Comput. Sci.*, vol. 5, pp. 997159–997171, Apr. 2023.
[66] Z. Ding, G. Zhong, X. Qin, Q. Li, Z. Fan, Z. Deng, X. Ling, and W. Xiang, "MF-net: Multi-frequency intrusion detection network for internet traffic data," *Pattern Recognit.*, vol. 146, Feb. 2024, Art. no. 109999.
[67] P. Rajasekaran and V. Magudeeswaran, "Malicious attacks detection using GRU-BWFA classifier in pervasive computing," *Biomed. Signal Process. Control*, vol. 79, Jan. 2023, Art. no. 104219.

YONGCHAO YANG received the M.S. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China. She is currently a Lecturer with the School of Big Data and Artificial Intelligence, Chizhou University. Her research interests include network security and deep learning.

PAN ZHAO received the M.S. degree in communication and information systems from Hohai University, Nanjing, China. She is currently a Lecturer with the School of Big Data and Artificial Intelligence, Chizhou University. Her research interests include network security and machine learning.