IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 December 2023, accepted 10 January 2024, date of publication 12 January 2024, date of current version 22 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3353488

RESEARCH ARTICLE

Enhancing Network Intrusion Detection Through the Application of the Dung Beetle Optimized Fusion Model

YUE LI, JIALE ZHANG, YITING YAN, YUTIAN LEI, AND CHANG YIN
School of Computer Science and Technology, Donghua University, Shanghai 201620, China

Corresponding author: Jiale Zhang (jialez1023@163.com)

This work was supported by the Shanghai Science and Technology Innovation Action Plan Project under Grant 22511100700.

ABSTRACT With the rapid development of information communication and mobile device technologies, smart devices have become increasingly popular, providing convenience to households and enhancing the level of intelligence in daily life. This trend is also driving innovation and progress in various fields, including healthcare, transportation, and industry. However, as technology continues to proliferate, network security concerns have become increasingly prominent, making the protection of digital life and data security an urgent priority. Intrusion detection has always played an important role in the field of network security. Traditional intrusion detection systems predominantly rely on anomaly detection technology to identify potential intrusions by detecting abnormal patterns in network traffic. With technological advancements, machine learning-based methods have emerged as the cornerstone of modern intrusion detection, enabling more precise identification of abnormal behaviors and potential intrusions by learning the patterns of normal network traffic. In response to these challenges, this paper introduces an innovative intrusion detection model that amalgamates the Attention-CNN-BiLSTM (ACBL) and Temporal Convolutional Network (TCN) architectures. The ACBL and TCN models excel in processing spatial and temporal features within network traffic data, respectively. This integration harnesses diverse neural network structures to elevate overall model performance and accuracy. Furthermore, a unique approach inspired by dung beetles' natural behavior, incorporating Tent mapping-enhanced Dung Beetle Optimization Algorithm (TDBO), is leveraged for both optimizing feature selection parameters and searching for optimal model hyperparameters. The feature selection parameters obtained from TDBO are then combined with the importance ranking from the Random Forest algorithm, ensuring optimal features can be better selected to enhance model performance. This paper introduces a novel intrusion detection model, the TDBO-ACBLT model, and validates its performance using the UNSW-NW15 dataset. TDBO excels in feature selection compared to common algorithms and achieves superior parameter optimization accuracy over Harris's Hawk Optimization (HHO), Particle Swarm Optimization (PSO), and Dung Beetle Optimization (DBO). The proposed model achieves higher accuracy than prevalent machine learning models.

INDEX TERMS Intrusion detection, network security, machine learning, deep learning, model fusion, dung beetle optimization algorithm DBO.

I. INTRODUCTION

With the rapid advancement of technologies such as the Internet of Things (IoT), big data, and artificial intelligence, networks are becoming increasingly complex, emphasizing the growing significance of network security. A central challenge in network is the provision of robust and potent intrusion detection systems [1]. An intrusion detection system serves as a pivotal component in network security infrastructure, capable of real-time monitoring of network

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

traffic and issuing alerts or taking actions upon detecting anomalous traffic. Thus, it safeguards the confidentiality, integrity, and availability of hosts and networks [2].

Network intrusion detection methods can be broadly categorized into two main types: misuse detection and anomaly detection. Misuse detection involves constructing an intrusion model based on known abnormal behaviors, designating actions conforming to this model as intrusions. This model typically exhibits a low false-positive rate and a high detection rate. On the other hand, anomaly detection entails building a model of normal network traffic behavior and identifying deviations from this established norm as potential intrusions. While this detection method can recognize unknown attacks, it is associated with a higher false-positive rate. Nevertheless, it remains effective in detecting both known and unknown attacks [3].

Traditional intrusion detection systems encounter significant challenges in dealing with the constantly evolving landscape of network threats. Current methodologies mainly rely on machine learning and deep learning techniques. Machine learning models manually extract features from network traffic to construct models, exhibiting low false-positive rates and high detection rates. Deep learning, as a subset of machine learning, utilizes neural networks for automatic feature extraction, thereby enhancing modeling and predictive capabilities in the field of intrusion detection.

However, existing models face limitations in addressing the inherent trade-off between high-dimensional feature spaces and the balance between false positives and detection accuracy. These models struggle to find equilibrium, resulting in suboptimal performance, especially when confronted with complex and dynamically changing network threats. Additionally, the model's performance heavily relies on parameter settings, and manual adjustments can increase the complexity of the work. Population-based intelligent optimization algorithms optimize objectives based on collective behavior, providing advantages in hyperparameter search and feature selection compared to other methods. Typically used for global search, these algorithms help avoid local optima, facilitating the discovery of superior hyperparameter configurations and identifying crucial features for the task. DBO, as a recent and efficient population-based intelligent algorithm, introduces new possibilities in this context.

In this paper, we attempt to address these issues and develop a more efficient detection model. Our contributions are outlined below:

1) We propose the innovative ACBL-TCN model, which integrates ACBL and TCN models. ACBL combines attention mechanism, CNN and BiLSTM, focusing on capturing spatial features in network traffic data. It places special emphasis on the detection of anomalous behaviors and attack patterns in network flows. In contrast, TCN primarily extracts temporal features from network traffic data. This unique fusion model effectively improves the detection rate while reducing the false alarm rate.
2) Faced with the challenges posed by high-dimensional feature spaces and model parameter settings, we employ the Tent-improved Dung Beetle Optimization algorithm (TDBO) for hyperparameter tuning and feature selection. In the feature selection stage, we integrate it with Random Forest. This facilitates the easier selection of more crucial features, thereby enhancing the efficiency and performance of the model.
3) We introduce a novel intrusion detection model, the TDBO-ACBLT model, which leverages TDBO for hyperparameter tuning and feature selection. The model is evaluated using the UNSW-NW15 dataset and experimentally compared with other commonly used feature selection methods, population-based intelligent optimization algorithms, ACBL, TCN, and other machine learning models. The results demonstrate that TDBO-ACBLT exhibits a lower false-positive rate and higher accuracy.

Introduction to relevant sections of the paper: The first section provides the research background and its significance. The second section delves into the relevant literature, outlining their strengths and limitations. The third section provides an overview of the DBO algorithm and the Tent algorithm, elucidating how Tent is utilized in conjunction with DBO. The fourth section comprehensively describes the model's structure and the methodologies employed. The fifth section describes the experimental design and evaluation of the model. Finally, in the sixth section, the paper summarizes the research findings and outlines potential avenues for future research.

II. RELATED WORK

A novel Intrusion Detection System (IDS) model is introduced, which integrates deep learning (DL) with metaheuristic optimization algorithms. The model employs a Convolutional Neural Network (CNN) for efficient feature extraction, followed by a fully connected layer to detect malicious activities [4]. To enhance feature selection, the article proposes a refined version of the reptile search algorithm (RSA), addressing issues related to premature convergence and balancing the exploration-exploitation trade-off. The model achieves higher accuracy compared to seven other algorithms.

A hybrid feature selection method, IGRF-RFE, designed for multi-class network anomaly detection using a multilayer perceptron (MLP) network, is introduced. The proposed approach strategically leverages the strengths of two filtering methods, namely information gain (IG) and random forest (RF), in the initial stage to effectively reduce the search space of feature subsets. Less crucial features are eliminated, thereby enhancing the relevance of the selected features. In the second stage, a machine learning-based wrapper method, recursive feature elimination (RFE), is used to further reduce the feature dimensions while considering the correlation of similar features. Experimental results on the

UNSW-NB15 dataset show significantly improved anomaly detection accuracy [5].

A network intrusion detection system employing a wrapper-based feature selection method named TS-RF [6]. The model combines Tabu Search as a feature search method and Random Forest as a learning algorithm, reducing the complexity of high-dimensional data and enhancing the accuracy and efficiency of intrusion detection. Experimental results on the UNSW-NB15 dataset demonstrate that TS-RF improves accuracy, significantly reduces feature space, lowers misclassification rates, and enhances detection accuracy for attacks with low sample counts.

The research utilizes a range of machine learning classifiers, such as K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Decision Tree (DT), and Stochastic Gradient Descent (SGD), to detect multi-class intrusion attacks within the Internet of Things (IoT) on the MQTT-IOT-IDS2020 dataset [7]. The classifiers demonstrate an overall accuracy ranging from 97.58% to 99.98%.

A feature selection approach utilizing Genetic Algorithm (GA) applied to the NSL-KDD dataset is presented [8]. This approach combines Logistic Regression (LR) and Decision Tree (DT) to enhance the detection rate (DR) and accuracy (ACC) of intrusion detection systems. By synergistically leveraging LR and DT, the model efficiently processes a larger set of variables while maintaining high accuracy and recognition rates. The Genetic Algorithm is used to select optimal attribute subsets, and the results confirm the effectiveness of this approach.

The study employs Genetic Algorithm (GA) for feature selection, incorporating the Random Forest (RF) model into the GA fitness function. Various classifiers, including RF, Linear Regression (LR), Naïve Bayes (NB), Decision Tree (DT), Extra-Trees (ET), and Extreme Gradient Boosting (XGB), are utilized in the intrusion detection process [9]. Experimental results on the UNSW-NB15 dataset indicate that GA-RF achieves an accuracy of 87.61% and an Area Under the Curve (AUC) of 0.98 in binary classification.

It is proposed to employ two adaptive search techniques, Genetic Algorithm and Particle Swarm Optimization, with the aim of enhancing the efficiency of 1D-CNN in intrusion detection systems [10]. The findings demonstrate notable enhancements in terms of accuracy, precision, recall, and F1 score. The Moth–Flame Optimizer (MFO) as the search algorithm and the Decision Tree (DT) as the evaluation algorithm in model [11]. This innovative framework focuses on identifying the optimal feature subset with minimal network traffic features while ensuring maximum accuracy in intrusion detection. To enhance the MFO, new operators are introduced, and the cosine similarity measure is utilized for binary conversion, overcoming limitations associated with the traditional sigmoid function. It is evaluated on three network datasets (KDDCUPP9, NSL-KDD and UNSW-NB15) and compared with existing frameworks in terms of accuracy, F-score, true positive rate (TPR) and false positive rate (FPR). The results show superiority in many aspects.

Developing an effective anomaly intrusion detection system utilizing evolutionary and machine learning techniques is the objective of the study. The emphasis is on specific attack features for precise discrimination between normal and malicious activities, achieving high accuracy and rapid learning speed [12]. A novel algorithm, Chaotic Adaptive Grasshopper Optimization Algorithm (CAGOA), incorporating a chaos concept, is employed for parameter optimization in Support Vector Machines (SVM). Furthermore, the research introduces a hybrid algorithm, ECAGOA, merging Ensemble of Feature Selection (EFS) and CAGOA, demonstrating superior performance in terms of accuracy, detection rate, and false alarm rate.

Introducing a hybrid FDO-XGBoost approach to address the common issue of false positive detections in network intrusion detection systems (NIDS) [13]. The proposed method leverages the FDO-XGBoost hybrid model, integrating machine learning and swarm intelligence optimization algorithms to enhance the overall classification accuracy of NIDS. The FDO, a fitness-dependent optimizer inspired by particle optimization (PSO) and bee swarming, is employed to dynamically optimize the parameters of the XGBoost classifier. Notably, the model demonstrates increased accuracy in detecting various types of attacks, particularly minority attack groups.The experimental results highlight the superior performance of the FDO-XGBoost model compared to other methods, suggesting potential applications of swarm intelligence approaches in NIDS. The research employs Particle Swarm Optimization (PSO) as the feature selection method and utilizes Naïve Bayes as the classifier, resulting in a higher level of accuracy [14].

Introducing an advanced variant, the BWO-BOA integrates the Butterfly Optimization Algorithm (BOA) with Black Widow Optimization (BWO) [15]. This novel approach introduces a dynamic adaptive search strategy in the global search phase of BOA and employs the movement search process of BWO as the local search. Additionally, a small probability mutation strategy is applied to filter redundant features, addressing challenges associated with BOA converging to a local optimum during the local search phase. Experimental results affirm the effectiveness of the BWO-BOA algorithm in improving feature selection model performance for network intrusion detection, leading to significant reductions in feature dimensions.

A hybrid feature selection optimization method that combines an improved binary Grey Wolf Optimization algorithm (bGWO) with a binary Particle Swarm Optimization algorithm (bPSO) is proposed [16]. In terms of feature selection, the paper introduces an enhanced bGWO through an improved position change technique and the incorporation of a multi-objective fitness function aimed at selecting the most important feature subset. Additionally, the combination

of bGWO and bPSO further boosts the effectiveness of the Intrusion Detection System (IDS). Experimental results demonstrate significant improvements in detection accuracy, detection rate, false alarm rate, feature quantity, and processing time compared to existing technologies.

The research introduces a robust intrusion detection model, termed the Remora Whale Optimization (RWO)-based Hybrid deep model. The model integrates the Remora Optimization Algorithm (ROA) and Whale Optimization Algorithm (WOA), forming the RWO optimization algorithm [17]. The methodology involves preprocessing input data through Z-score data normalization, transforming data using the holoentropy method, and extracting effective features using Convolutional Neural Network (CNN). Feature selection is performed using the RV coefficient, and intrusion detection is executed with a Hybrid deep model comprising Deep Maxout Network and Deep Auto Encoder. The proposed approach surpasses existing methods, demonstrating outstanding performance in testing accuracy (0.938), precision (0.920), recall (0.932), and F1-score (0.926).

Proposing an efficient hybrid intrusion detection model, the method employs the Enhanced Genetic Algorithm and Particle Swarm Optimization (EGA-PSO) for advanced feature selection. Additionally, it integrates the Improved Random Forest (IRF) technique to mitigate overfitting concerns [18]. The model is evaluated using the NSL-KDD dataset and compared with existing machine learning methods (SVM, RF, LR, NB, LDA, CART). The obtained results indicate superior performance compared to the mentioned machine learning methods. The primary contributions include addressing data imbalance, overfitting issues, and enhancing detection accuracy through EGA-PSO and IRF methods.

Table 1 shows comparative analysis of the proposed models, considering limitations and advantages.

In summary, numerous machine learning integrated swarm intelligence optimization algorithms have been proposed for network intrusion detection in recent years. However, most of them still exhibit certain limitations. For instance, many algorithms lack sufficient improvements to better adapt to the intrusion detection environment, potentially leading to convergence to local optima. While the combination of population optimization algorithms and machine learning yields favorable performance, experiments often neglect time-related analyses, limiting the comprehensive assessment of model efficiency. In the context of intrusion detection, the use of a single machine learning model may not be optimal for addressing the complexities of the environment.

To address these issues, we propose the integration of Tent Dung Beetle Optimization (TDBO) and ACBLT, assessing the performance of this hybrid model across various aspects. By effectively optimizing the ACBLT model using TDBO, we evaluated the algorithm's performance on the UNSW-NB15 dataset and observed a significant enhancement in its effectiveness.

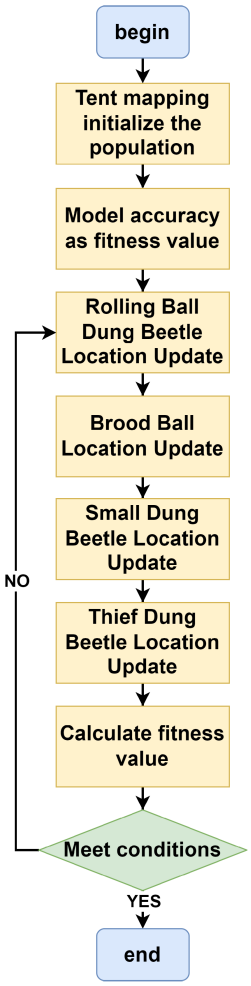

FIGURE 1. TDBO framework flow chart.

III. TENT DUNG BEETLE OPTIMIZATION

In this study, we optimized the Dung Beetle Optimization (DBO) algorithm using the Tent mapping. Figure 1 shows TDBO framework flow chart.

A. DUNG BEETLE OPTIMIZATION ALGOIRITHM

The Dung Beetle Optimization (DBO) algorithm is a heuristic optimization algorithm inspired by the foraging behavior of dung beetles [19]. This algorithm has excellent optimization capabilities and fast convergence speed. Within a dung beetle population, the differences in abundance of individuals seeking food result in variations in individual fitness. Based on fitness ranking, individuals are categorized into four roles: ball-roller dung beetles, brood ball, small dung beetles, and thief dung beetles, sorted from high to low fitness. Roller dung beetles are individuals with higher fitness, and their primary goal is to move dung balls to a suitable location for reproduction. Female dung beetles will slightly relocate the dung ball of a roller dung beetle and lay their eggs on it, creating a brood ball. After hatching, small dung beetles search for food around the brood ball.

TABLE 1. Comparative analysis of proposed models, considering limitations and advantages.

Ref	Algorithm	Datasets	Classifier	Algorithm Usage	Limitations	Advantages
[4]	RSA	KDDCup99, NSL-KDD, CICIDS2017, BoT-IoT	CNN	Feature selection	Increasing time complexity.	high accuracy and high recall rate.
[5]	IGRF-RFE	UNSW-NB15	MLP	Feature selection	The accuracy of detection is lower than other models.	Combining IGRF for feature selection and RFE for feature elimination can obtain better features and improve performance.
[6]	TS	UNSW-NB15	Random forest	Feature selection	The accuracy of detection is lower than other models.	Reduce computational complexity through feature selection.
[7]	-	MQTT-IoT	KNN, SVM, DT	-	Lack of improved algorithms.	Model implementation has less complexity.
[8]	GA	NSL-KDD	LR, DT	Feature selection	GA algorithm is weaker than GWO.	Reduce computational complexity through feature selection.
[9]	GA-RF	UNSW-NB15	RF	Feature selection	The initial population lacks diversity and is unable to discover crucial features.	Reduce computational complexity through feature selection.
[10]	GA, PSO	UNSW-NB15, CICIDS2017	1D-CNN	Hyperparameter search	Lack of improved algorithms.	High detection rate and low false alarm rate.
[11]	MFO	KDDCup99, NSL-KDD, UNSW-NB15	DT	Feature selection	Not tested on large data sets.	Reduce computational complexity through feature selection.
[12]	ECAGOA	NSL-KDD	SVM	Feature selection	Increased time complexity.	Achieve high accuracy.
[13]	FDO	NSL-KDD	XGBoost	Hyperparameter search	Untested time complexity.	Achieve high accuracy.
[14]	PSO	KDDCup99	Naïve Bayes	Feature selection	Lack of improved algorithms.	Achieve high accuracy.
[15]	BWO-BOA	UNSW-NB15	KNN	Feature selection	Untested time complexity.	Ensure higher accuracy while reducing feature selection redundancy.
[16]	bGWbPS	NSL-KDD, UNSW-NB15	SVM, DT	Feature selection	The accuracy of detection is lower than other models.	High detection rate and low false alarm rate.
[17]	RWO	NSL-KDD, CICIDS, UNSW-NB15	CNN	Feature selection	Untested time complexity and low accuracy.	The proposed new optimization algorithm RWO can better extract the features required by CNN.
[18]	EGA-PSO	NSL-KDD	IRF	Feature selection	Untested time complexity.	Achieve high accuracy.

Thief dung beetles, on the other hand, attempt to steal dung balls from other dung beetles. Each role corresponds to specific positional adjustment strategies. After each foraging event, an individual's fitness ranking determines its role in the next foraging event. Through this role assignment mechanism based on fitness ranking, individuals optimize their foraging behavior strategies, finding positions that are suitable for the survival and reproduction of the population.

1) Ball-Roller Dung Beetles: Ball-roller dung beetles roll food balls to a location suitable for reproduction. They navigate and roll food balls in a straight line, guided by celestial cues like the sun, moon, and polarized light. In the absence of light, their path may become curved or even circular. When encountering obstacles, ball-roller dung beetles may climb onto the food ball and dance to determine the direction of movement. The position update formula for ball-roller dung beetles is as follows:

$$x_i(t+1) = \begin{cases} x_i(t)+\alpha \cdot k \cdot x_i(t-1)+b \cdot \Delta x, & \text{if } R < 0.9 \\ x_i(t)+\tan(\theta) \cdot |x_i(t)-x_i(t-1)|, & \text{if } R \geq 0.9 \end{cases} \tag{1}$$

$$\Delta x = |x_i(t) - X^w| \tag{2}$$

where $x_i(t)$ represents the position information of the i-th dung beetle at the t-th iteration, $k \in (0, 0.2]$ represents a constant, which is expressed as a deflection coefficient, $b \in (0, 1)$ is a constant, α represents the natural coefficient assigned a value of -1 or 1, X^w represents the global worst position, Δx is used to simulate light intensity changes. When $R \geq 0.9$ indicates that the dung beetle needs to adjust its direction when it encounters an obstacle, and θ indicates the deflection angle.

2) Brood Ball Dung Beetles: Brood ball dung beetles are another category of high-fitness individuals whose goal is to move food balls to a suitable location for laying eggs. They relocate the food ball to a new position and lay eggs on it, creating a brood ball. After hatching, small dung beetles search for food around the brood ball. The position update formula for brood ball dung beetles is as follows:

$$\begin{cases} Lb^* = \max(X^*(1-R), Lb) \\ Ub^* = \min(X^*(1+R), Ub) \end{cases} \tag{3}$$

$$R = 1 - \frac{t}{T_{\max}} \tag{4}$$

$$x_i(t+1) = X^* + b_1 \cdot (x_i(t) - Lb^*) + b_2 \cdot (x_i(t) - Ub^*) \tag{5}$$

where X^* represents the current local optimal position, Lb^* and Ub^* respectively indicate the lower and upper bounds of the oviposition area, $T_{\max}$ represents the maximum number of iterations, Lb and Ub represent

the lower and upper bounds of the optimization problem, b_1 and b_2 represent two independent random variables with a size of $1 \times D$, where D represents the dimension.

3) Small Dung Beetles: Small dung beetles search for food by rolling food balls. Some food balls are used for egg laying and raising the next generation, while the rest serve as a source of food. They bury food balls, and female beetles lay eggs in them. The food balls provide a place for larval growth and essential nourishment. The position update formula for small dung beetles is as follows:

$$\begin{cases} Lb^b = \max(X^b \times (1-R), Lb) \\ Ub^b = \min(X^b \times (1+R), Ub) \end{cases} \tag{6}$$

$$x_i(t+1) = x_i(t) + C_1 \cdot (x_i(t) - Lb^b) + C_2 \cdot (x_i(t) - Ub^b) \tag{7}$$

where X^b represents the global optimal position, Lb^b and Ub^b respectively indicate the lower and upper bounds of the best foraging area. C_1 represents a random number following a normal distribution, and C_2 represents a random variable belonging to the range (0, 1).

4) Thief Dung Beetles: Some dung beetles exhibit the behavior of stealing food balls from other dung beetles. They may compete with other dung beetles to acquire food balls for their own purposes. The position update formula for thief dung beetles is as follows:

$$x_i(t+1) = X^b + S \cdot g \cdot (|x_i(t) - X^*| + |x_i(t) - X^b|) \tag{8}$$

where S represents a constant value and g is a $1 \times D$ random vector following a normal distribution.

B. TENT OPTIMIZATION FOR DBO ALGORITHM

Tent mapping is a technique employed in Tent Dung Beetle Optimization (TDBO). The initial population of DBO is randomly generated, which may result in poor population quality. We utilize the Tent chaotic algorithm for population initialization because it can generate Gaussian-distributed random numbers, exhibiting good randomness and exploratory characteristics. This enhances population diversity and initial quality, ensuring an even distribution of initial solutions in the solution space, thereby improving optimization performance. Tent mapping formula is as follows:

$$X_{t+1} = \begin{cases} \dfrac{X_t}{b}, & \text{if } 0 \le X_t < a \\ \dfrac{1 - X_t}{1 - b}, & \text{if } a \le X_t < 1 \end{cases} \tag{9}$$

where X_t represents the value of X at time t, and a is randomly chosen from (0.5, 1). In out study, we set $X_0 = 0.5$. We employ Equation 10 to obtain the initial values for the DBO population. For values that exceed the boundaries, we apply Equation 11 for normalization.

$$x_i(t) = X_t \cdot (Ub - Lb) + Lb \tag{10}$$

$$x_i(t) = \begin{cases} Lb, & \text{if } x_i(t) < Lb \\ Ub, & \text{if } x_i(t) > Ub \end{cases} \tag{11}$$

IV. THE PROPOSED MODEL

In this section, we introduce an intrusion detection model named "TDBO-ACBLT." The model integrates the strengths of TDBO and ACBLT, resulting in higher accuracy in identifying intrusion detection attacks. The TDBO algorithm is employed for feature selection and hyperparameter search. The combination of TDBO and the advantages of random forest in feature selection enhances the selection of high-quality features and reduces model complexity. This contributes to the reduction of memory and time required for training and deploying the model. Hyperparameter search enables us to obtain the optimal model, improving accuracy. ACBLT combines the strengths of ACBL and TCN, identifying features in both spatial and temporal dimensions, adapting well to diverse environments, and demonstrating effective performance. For better understanding, Figure 2 shows the overall structure of the proposed intrusion detection model.

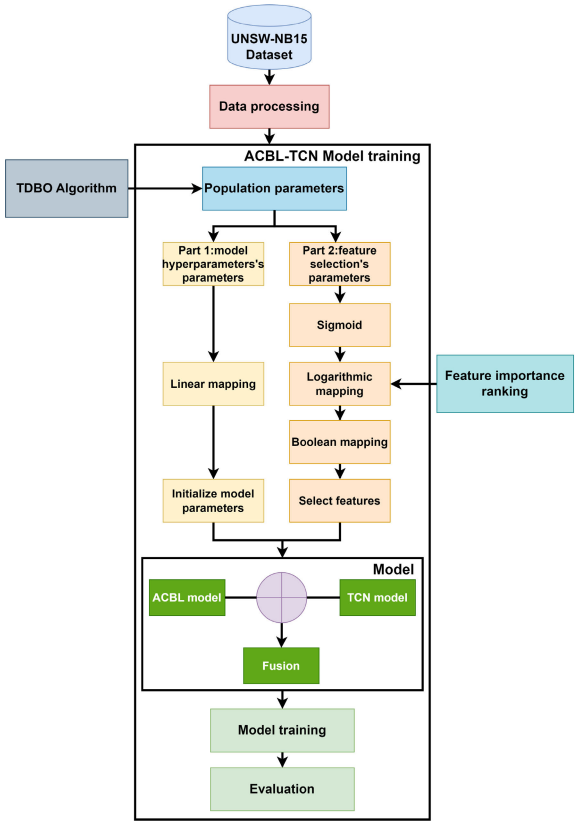

FIGURE 2. Overall structure of the model.

A. DATASET AND PREPROCESSING

To evaluate various intrusion detection models, researchers typically employ a variety of datasets. The UNSW-NB15

dataset [20], provided by the Australian Centre for Cyber Security's Network Security Research Group (ACCS), comprises 2.54 million network traffic samples across nine attack categories. Each sample contains 49 features, two of which are class label features. In this study, the UNSW-NB15 dataset is used to train and test the proposed model.

Data preprocessing is essential for machine learning and deep learning projects. Its objective is to transform raw data into a format suitable for model training, cleaning and reducing noise to enhance model performance. Figure 3 shows the data preprocessing process conducted on the UNSW-NB15 dataset.

First, we used publicly available network traffic datasets, which included both training and testing sets stored in CSV files, with network traffic features in PCAP format. We loaded these datasets using Python's Pandas library. Afterward, we performed data cleaning and removed irrelevant columns. This process involved checking for missing values or duplicates in the data. Fortunately, our dataset had no missing values. We also removed two irrelevant columns, namely 'id' and 'attack_cat,' to simplify the dataset and eliminate information unrelated to the research task. To enable deep neural networks to process these features, we encoded non-numeric features. Categorical features like 'proto,' 'service,' and 'state' were transformed into numerical values using LabelEncoder, making them understandable and processable by the model. Subsequently, we normalized the data to achieve uniform scales across various features. This was achieved by using MinMaxScaler to scale the input data, enhancing the model's learning efficiency and performance. Finally, to address the issue of data imbalance, we applied the SMOTE-ENN method. This involved generating synthetic samples for the minority class while also removing some synthetic samples. The goal was to achieve a more balanced dataset, thereby improving the model's ability to effectively learn from the minority class. Table 2 presents the traffic distribution diagram before and after data preprocessing.

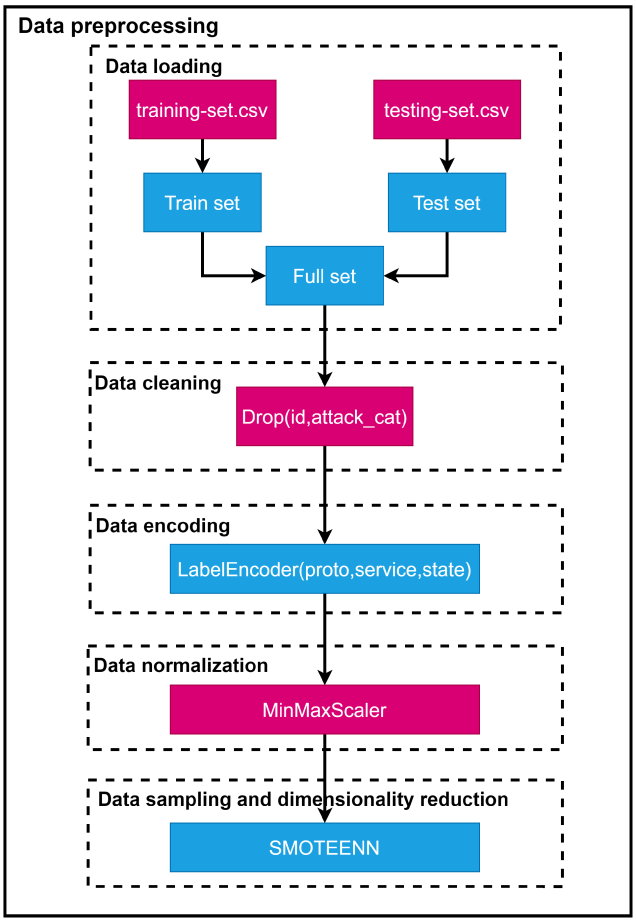

FIGURE 3. Data preprocessing process.

B. FEATURE SELECTION AND HYPERPARAMETER SEARCH

The TDBO algorithm initially generates a population of dung beetles using TENT for initialization. Each population represents a solution, represented by a vector of dimensions equal to the sum of the number of features in the dataset and the number of hyperparameters to be searched. The population size is denoted by pop, and the dimensionality is denoted by dim. The dataset in this study has 42 features, and the number of hyperparameters is 6. The specific parameter settings are as Table 3.

In the quest for optimal population parameters guided by fitness, with accuracy serving as the fitness criterion in this study, the population parameters of TDBO are divided into two parts to fulfill different functions. The first portion of the population parameters is utilized for hyperparameter search, employing linear mapping to automatically convert population parameters into hyperparameters, aiming to obtain optimal hyperparameters and enhance model performance. The second portion of the population parameters is employed for feature selection. Utilizing a combination of logarithmic mapping, random forest feature importance ranking, and Boolean mapping, these parameters are transformed into a binary sequence of 0s and 1s. Here, 0 denotes the exclusion of a particular feature, and 1 indicates its inclusion. This approach favors the selection of features with greater importance, thereby improving algorithm efficiency and accelerating convergence.

1) HYPERPARAMETER SEARCH

For the hyperparameter sequence of TDBO, we perform hyperparameter mapping to initialize the model's hyperparameters. The corresponding hyperparameters and their search ranges are shown in Table 4. Here, "Hidden Size" represents the size of the hidden layer in Bi-LSTM, "Kernel Size" indicates the convolutional layer size in the TCN model, "Dropout1" corresponds to ACBL parameters, and "Dropout2" corresponds to TCN parameters. Population parameters belonging to the [LB, RB] range are denoted as X, and the model hyperparameter values Y fall within the $[MinValue, MaxValue]$ range. The linear mapping formula

TABLE 2. Traffic distribution before and after data preprocessing.

	Train-normal traffic (0)	Train-malicious traffic (1)	Test-normal traffic (0)	Test-malicious traffic (1)
Before	56000	119341	37000	45332
After	111288	105084	40293	39268

TABLE 3. TDBO parameter settings.

Population size (pop_size)	30
Parameter dimension (dim)	48
Upper parameter bound (ub)	10
Lower parameter bound (lb)	-10
Maximum number of iterations (max_iter)	10

TABLE 4. Hyperparameter value range.

Hyperparameter	Min Value	Max Value
Learning Rate	0.0001	0.001
Batch Size	64	512
Hidden Size	32	512
Kernel Size	3	7
Dropout1	0.1	0.5
Dropout2	0.1	0.5

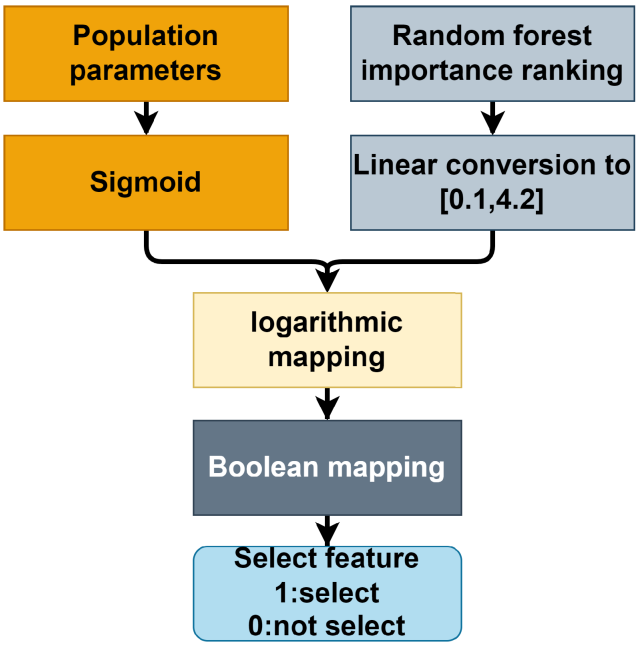

FIGURE 4. Feature selection parameter mapping process.

below allows us to obtain value Y.

$$\frac{Y - \text{MinValue}}{\text{MaxValue} - \text{MinValue}} = \frac{X - \text{LB}}{\text{RB} - \text{LB}} \tag{12}$$

2) FEATURE SELECTION

For the feature selection sequence of TDBO, we apply sigmoid processing and combine it with random forest for feature selection. Figure 4 shows the feature selection process.

1) After data preprocessing, we used a random forest classifier to comprehensively evaluate the importance of each feature. Features were ranked based on their importance, with highly important features receiving a lower numerical rank (e.g., 42 for the most important) and less important features receiving a higher numerical rank (e.g., 1 for the least important). The results of this feature ranking are represented as $rank(i)$ in $[1, 42]$, where i represents the feature's index. We then transformed these ranking values into $feature\_sort(i)$, denoted as $fs(i)$. The specific transformation is as follows:

$$fs(i) = \frac{rank(i)}{10} \tag{13}$$

2) We first perform sigmoid mapping on the parameter X obtained by TDBO to the range $[0, 1]$, denoted as Pa. Next, We use the following logarithmic mapping formula:

$$bf(i) = \frac{\ln(1 + Pa(i) \cdot fs(i))}{\ln(1 + fs(i))} \tag{14}$$

Since we are dealing with 42 processed features, the values of $fs(i)$ fall between 0.1 and 4.2. Figure 5 illustrates the relationship between Pa and fs, corresponding to changes in bf. From the graph, it is evident that Pa and fs are positively correlated. We also find that in the case of $bf = 0.5$, representing the boundary between selection and non-selection, when $fs = 0.1$, the required Pa is 0.4881, and when $fs = 4.2$, the required Pa is 0.3048. Consequently, features with higher rankings will be more likely to be selected.

3) To obtain binary selection data (0 or 1) based on the continuous values, a boolean mapping is applied. In this mapping, 0 represents the feature is not selected, and 1 represents the feature is selected. The specific

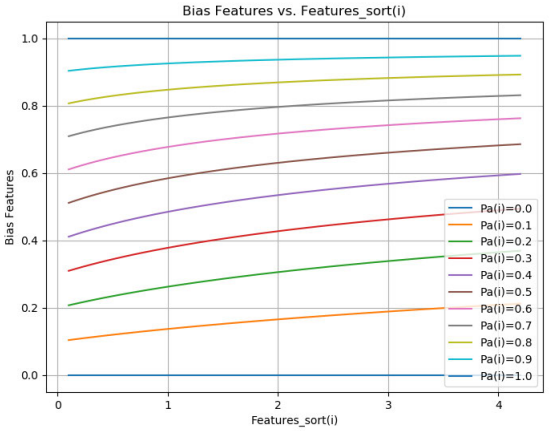

FIGURE 5. Feature selection relationship diagram.

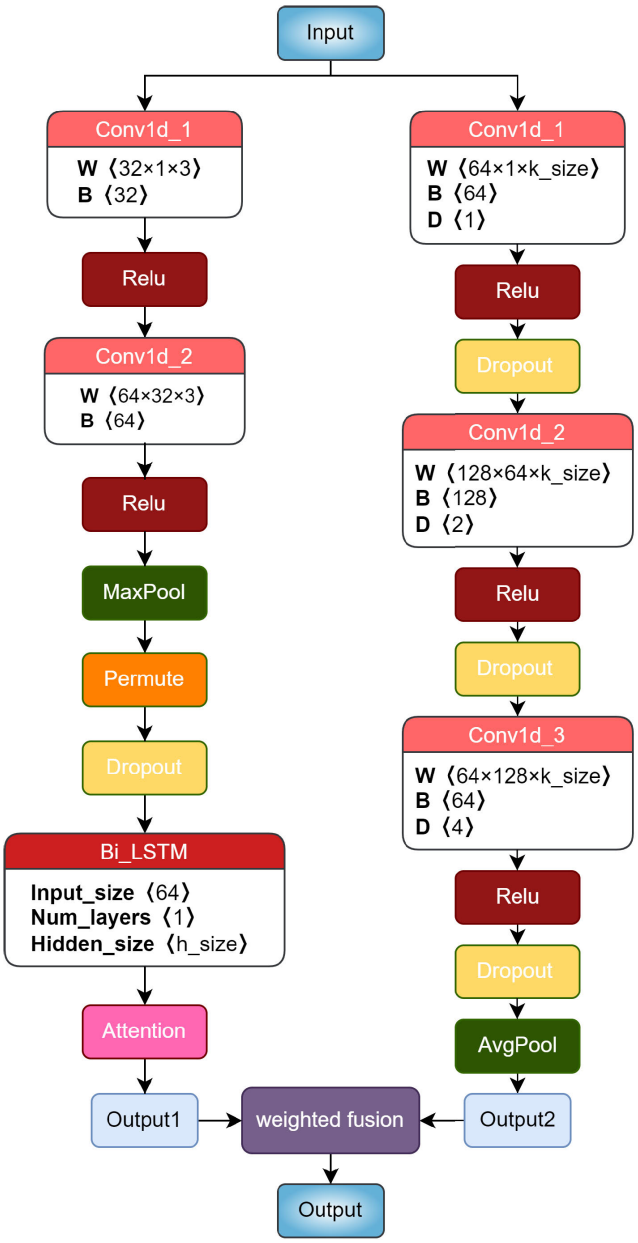

FIGURE 6. ACBL-TCN fusion model structure diagram.

formula for this mapping is as follows:

$$\text{Result} = \begin{cases} 0, & \text{if } bf \leq 0.5 \\ 1, & \text{if } bf > 0.5 \end{cases} \tag{15}$$

C. ACBL-TCN MODEL

ACBL-TCN, consisting of two primary components: the ACBL branch (Attention-CNN-BiLSTM) and the TCN branch (Temporal Convolutional Network). The following provides a detailed breakdown of these components and their collaborative functioning within the fusion model. Figure 6 shows ACBL-TCN fusion model structure diagram. The variables k_size and h_size are the search parameters that will be mentioned later.

1) ATTENTION-CNN-BILSTM PART

The ACBL model, tailored for efficient spatiotemporal pattern learning in network traffic data, seamlessly integrates convolutional and sequential processing through a fusion of 1D convolutional layers and a Bidirectional Long Short-Term Memory (BiLSTM) layer enhanced with an attention mechanism.

Commencing the architecture are two 1D convolutional layers capturing hierarchical spatial features within the input data. The Rectified Linear Unit (ReLU) activation function introduces non-linearity after each convolutional operation. Subsequently, a max-pooling layer downsamples spatial dimensions, enhancing computational efficiency while preserving essential features.

To capture temporal dependencies effectively, the processed data undergoes reshaping using the permute operation before entering the BiLSTM layer. The BiLSTM, a bidirectional variant of the Long Short-Term Memory network, adeptly captures both past and future temporal relationships in sequential data. To counteract overfitting, dropconnect regularization selectively applies to the BiLSTM weights, contributing to robustness during training.

The attention mechanism, implemented through the AdditiveAttention class, replaces the traditional time-step selection process in BiLSTM. It dynamically assigns weights to different parts of the input sequence, enabling the model to focus on salient information and improve its ability to discern relevant spatiotemporal patterns.

The ACBL model seamlessly integrates spatial and temporal processing through convolutional and sequential operations. The incorporation of attention mechanisms and dropconnect regularization enhances the model's capacity to learn intricate patterns in network traffic data.

2) TCN PART

The TCN model, meticulously crafted to capture intricate temporal patterns in time series data, seamlessly integrates convolutional operations with hierarchical temporal processing. This integration is achieved through a stack of Causal Convolutional Layers, precisely designed to extract diverse temporal features across multiple scales.

Commencing the model architecture are three Causal Convolutional Layers, each tailored to capture specific ranges of temporal relationships in a causally ordered manner. The kernel sizes and dilations are carefully selected to ensure effective feature extraction, preventing future information from influencing the current time step. After each convolutional operation, ReLU activation functions and dropout layers are strategically employed to introduce non-linearity and address overfitting concerns.

To aggregate the temporal features effectively, a global average pooling operation is employed, summarizing the crucial information gleaned from the Causal Convolutional Layers. This operation enhances computational efficiency while preserving essential temporal patterns.

The TCN model excels at capturing intricate temporal relationships by employing a series of causally ordered convolutional layers, allowing it to discern temporal dependencies at various scales. The systematic use of activation functions and dropout layers contributes to the model's adaptability and robustness during training. The incorporation of global average pooling enhances its ability to extract essential temporal features, contributing to its overall effectiveness in temporal pattern learning.

3) MODEL FUSION

The fusion model relies on fully connected layers for fusion, which receives features from the ACBL branch and TCN branch and their combined representations, and produces the final prediction. The fusion process employs a method of learning weights to effectively blend the contributions from the ACBL and TCN branches, facilitating a comprehensive representation of intricate spatiotemporal patterns. This fully connected layer proficiently integrates temporal and spatial features, endowing the intrusion detection model with robust modeling capabilities.

V. EXPERIMENT RESULTS

In this chapter, we present the experimental results and conduct an in-depth evaluation. First, we define the primary evaluation metrics that will be used to measure the performance of different models or methods. Next, we describe the experimental environment, including hardware configuration, operating system, programming language and libraries. Finally, we compare the performance results of different models or methods and provide analysis and discussion. During the TDBO search, our model is trained with the number of epochs (num_epoch) set to 150, using accuracy as the fitness metric. Once the optimal parameters are found, it will be adjusted to 500 for further training.

A. EVALUATION METRICS

In classification experiments, it is essential to employ appropriate evaluation metrics to assess the performance of the classification algorithm. Commonly used indicators include accuracy, precision, true positive rate, false positive rate, F1 score, ROC curve, and AUC. These metrics collectively offer a comprehensive evaluation of classifier performance. The confusion matrix, presented in Table 5, serves as a vital tool for evaluating classification models, especially in binary or multi-class scenarios. Effectively illustrating the relationship between classifier predictions and actual results.

TABLE 5. Confusion matrix.

Actual \ Predicted	Attack	Normal
Attack	TP	FN
Normal	FP	TN

In this study, normal network traffic is designated as the negative class, and network attack traffic is labeled as the positive class. The meanings are as follows:

- True Positives (TP): The model correctly identifies actual network attacks, meaning it accurately predicts an attack.
- False Positives (FP): The model incorrectly classifies cases that are actually normal behavior as network attacks, meaning it erroneously predicts normal behavior as an attack.
- True Negatives (TN): The model correctly identifies actual normal behavior, meaning it accurately predicts normal.
- False Negatives (FN): The model incorrectly classifies actual network attacks as normal behavior, meaning it erroneously predicts an attack as normal.

The evaluation metrics utilized are as follows:

Accuracy represents the proportion of correctly classified samples out of the total samples. The formula for accuracy is as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

Precision measures the accuracy of a classifier in predicting positive samples, representing the ratio of correctly predicted positives to the total predicted positives. Higher precision indicates more accurate positive predictions. The formula for precision is as follows:

$$\text{precision} = \frac{TP}{TP + FP} \tag{17}$$

True Positive Rate (TPR), also known as recall, representing the ratio of correctly predicted positives to the total actual positives. A higher true positive rate indicates better identification of positive examples by the classifier. Its calculation method is as follows:

$$\text{TPR} = \frac{TP}{TP + FN} \tag{18}$$

The False Positive Rate (FPR) refers to the ratio of the number of samples incorrectly predicted as positive samples by the classifier to the total number of actual negative samples. The smaller the false positive rate, the more accurately the classifier can identify negative examples. Its calculation method is as follows:

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

The F1 score is the harmonic mean of precision and recall, allowing it to consider both precision and recall. Its calculation method is as follows:

$$\text{f1\_score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{20}$$

The ROC curve visually depicts a classifier's ability to distinguish between positive and negative samples, while the AUC summarizes its overall performance. A higher AUC signifies better classifier performance, with values ranging from 0.5 (random guessing) to 1 (excellent discrimination).

B. EXPERIMENT ENVIRONMENT

The experimental environment and tools utilized in our experiments are outlined in Table 6 as follows:

TABLE 6. Experiment environment.

OS	Windows 11
GPU	NVIDIA RTX 3070
RAM	16GB
Anaconda	4.5.4
Pytorch	2.0.0+cu118
Python	3.8.10

C. COMPARISON OF DIFFERENT FEATURE SELECTION METHODS

In this section, we conducted experiments using various feature selection strategies, including Recursive Feature Elimination (RFE), Random forest, KBest, LaSSO and PCA. The number of selected features was set to 20 for all methods.

Figure 7 displays the confusion matrices for six feature selection methods. It can be observed that the proposed model exhibits larger numbers on the main diagonal of the confusion matrix, indicating superior performance. Table 7 provides a comparison of methods. In comparison to other feature selection methods, the proposed model demonstrates higher accuracy, reaching 97.32%. This suggests that employing TDBO for feature selection outperforms many other methods.

TABLE 7. Performance comparison under different feature selection strategies.

Method	ACC	PRE	TPR	F1
RFE	0.9315	0.9432	0.9163	0.9296
Random forest	0.9478	0.9494	0.9446	0.9470
KBest	0.9358	0.9120	0.9628	0.9367
LaSSO	0.9590	0.9403	0.9792	0.9593
PCA	0.9439	0.9339	0.9537	0.9437
Our method	0.9732	0.9773	0.9682	0.9727

D. COMPARISON OF DIFFERENT SWARM INTELLIGENCE OPTIMIZATION ALGORITHMS

This section aims to compare the performance of different swarm intelligence optimization algorithms under the same hyperparameter settings. We conducted experiments using various swarm optimization algorithms, including the Harris's Hawk Optimization algorithm (HHO) proposed in 2019 [21], Particle Swarm Optimization (PSO) proposed in 1995 [22], Dung Beetle Optimization (DBO), Tent Dung Beetle Optimization (TDBO), Dung Beetle Optimization with direct linear mapping feature selection (TDBO-linear-mapping), and Dung Beetle Optimization with random feature importance ranking (TDBO-random-importance).

To improve efficiency, we randomly selected approximately 5% of the data for testing in each experiment, with a population size of 30 and 30 iterations. Additionally, the parameters for the PSO algorithm were set to c1=1.5, c2=2.0, and w=1.

Upon examining Figure 8, it becomes evident that with an increase in the number of iterations, the model's accuracy

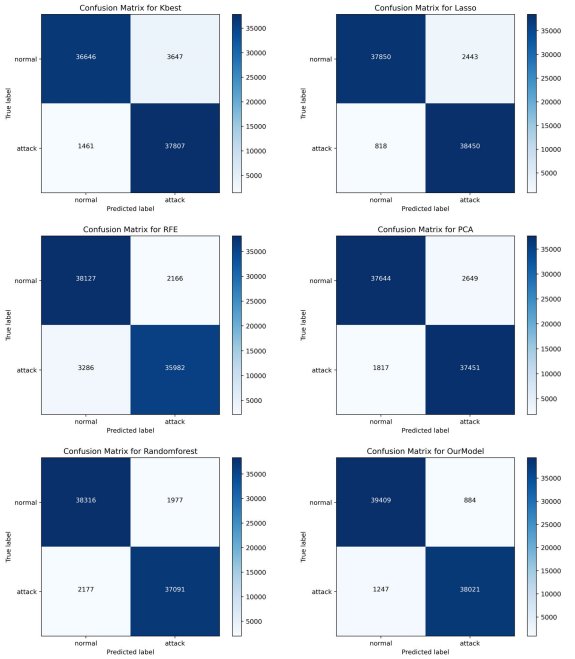

FIGURE 7. Confusion matrix under different feature selection strategies.

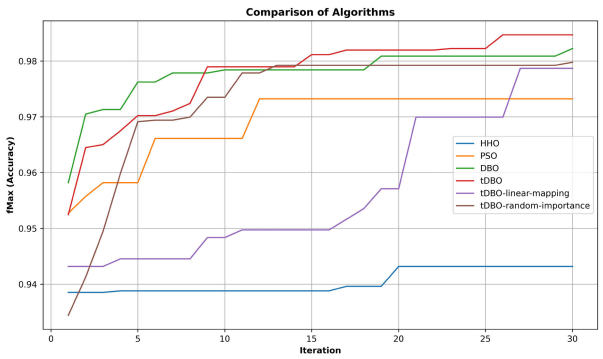

FIGURE 8. Comparison of iteration counts and accuracy for different swarm intelligence optimization algorithms.

consistently exhibits an upward trajectory. Particularly noteworthy is the substantial accuracy improvement achieved by the model employing the Dung Beetle Optimization (DBO) algorithm. In the comparative analysis of the curves for TDBO, TDBO with linear mapping feature selection, and TDBO with random feature importance ranking, we observe that the TDBO curve is significantly higher than the other curves, indicating the superiority of the TDBO curve over other optimization algorithms. Figure 9 illustrates the best fitness and the average time required for all iterations. The utilization of TDBO results in relatively shorter execution times compared to other algorithms, thus contributing to improved algorithm efficiency.

E. COMPARISON WITH MACHINE LEARNING MODELS

In this section, we conduct a comprehensive analysis of the model's training and evaluation results in comparison with other machine learning models.

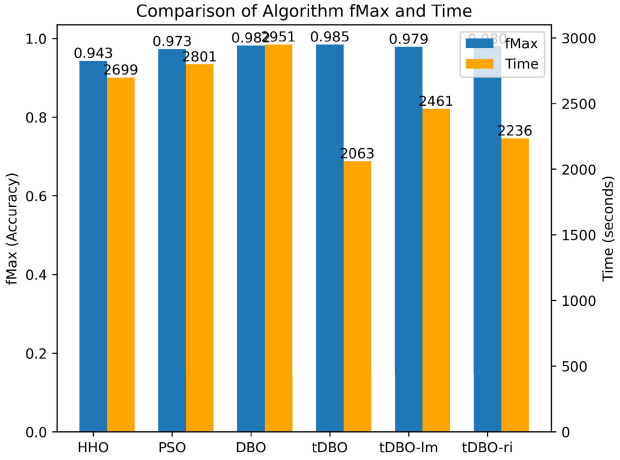

FIGURE 9. Comparison of accuracy and time of different swarm intelligent optimization algorithms.

We compared the accuracy (ACC) of common machine learning models (NB, LR, DT, KNN, SVM, ADABOOST, LSTM, CNN, DNN) with and without feature selection, as shown in Figure 10. Most models experienced a moderate increase in accuracy after feature selection, indicating its positive impact. Particularly, the proposed model demonstrated a significant accuracy improvement when feature selection was applied, emphasizing its effectiveness.

Furthermore, we evaluated our model against other models using Receiver Operating Characteristic (ROC) curves (Figure 11), providing an intuitive performance assessment for positive and negative samples. Our model exhibited the smoothest curve, with an AUC of 1.000, indicating outstanding performance and excellent classification capability. The practicality and wide applicability of our model have been confirmed.

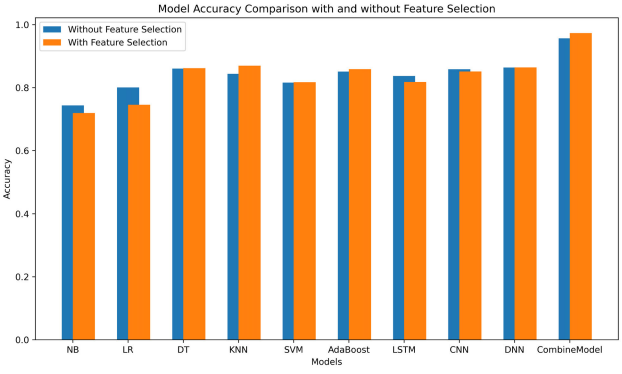

FIGURE 10. Comparison of model accuracy with and without feature selection.

F. ABLATION EXPERIMENT

In this section, we conducted ablation experiments to compare different models with the same hyperparameter settings. The hyperparameters used were the optimal ones obtained by our model, as shown in Table 8, while the optimal features were illustrated in Figure 12.

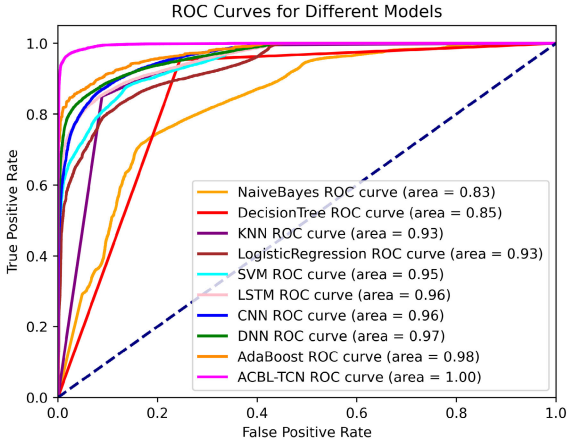

FIGURE 11. Comparison of ROC curves for different models.

TABLE 8. Optimal Hyperparameters.

Parameter	Value
Batch_size	128
learning_rate	0.0001793595368109867
Acbl_hidden_size	305
Tcn_kernel_size	6
Tcn_dropout	0.48728281897670433
Acbl_dropconnect	0.1001591397521149

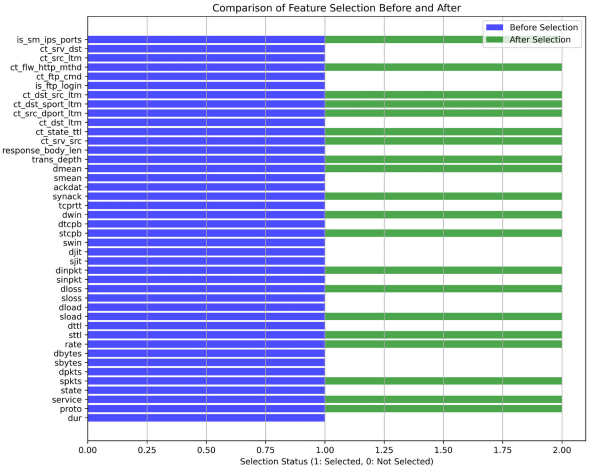

FIGURE 12. Comparison before and after feature selection.

Firstly, we present the loss and accuracy plots for the model across 500 epochs. Observing Figure 13(a), as the training progresses, the model's loss gradually decreases and eventually stabilizes. In Figure 13(b), the training set accuracy gradually improves. However, it is worth noting that the accuracy on the test set initially increases with the number of epochs, reaching its peak around 100 epochs but then experiences a slight decline. This observation suggests that the model may start overfitting the training data after a certain number of epochs, resulting in a decrease in performance on the test data. Choosing the right number of epochs is beneficial to prevent overfitting and

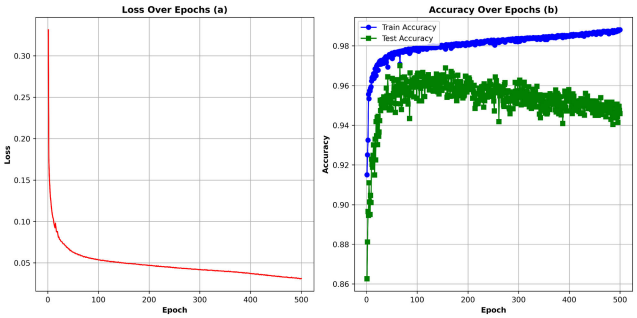

FIGURE 13. Changes in loss and accuracy obtained from training with optimal Hyperparameters.

improve efficiency, especially when training multiple models in swarm intelligence optimization algorithms.

We employed a variety of models for experimental comparisons, including standalone ACBL models, standalone TCN models, the ACBL-TCN integrated model, ACBL models with SMOTEENN, TCN models with SMOTEENN, ACBL-TCN models with SMOTEENN, and the ACBL-TCN model incorporating both SMOTEENN and feature selection, denoted as TDBO-ACBT. The results in Table 9 indicate that as the model complexity increases, its performance gradually improves. The accuracy increased by 8% compared to ACBL and 9% compared to TCN, while the false positive rate decreased by approximately 18%. This demonstrates the rationality and effectiveness of our model design.

TABLE 9. Performance comparison of different models with the same Hyperparameters.

Model	ACC	PRE	FPR	F1
ACBL	0.8903	0.8540	0.2023	0.9065
TCN	0.9036	0.8616	0.1935	0.9182
ACBL-TCN	0.9110	0.8814	0.1597	0.9230
SE-ACBL	0.9440	0.9330	0.0668	0.9439
SE-TCN	0.9525	0.9325	0.0688	0.9529
SE-ACBL-TCN	0.9581	0.9548	0.0443	0.9576
TDBO-ACBT	0.9732	0.9773	0.0219	0.9727

G. COMPARISON ON THE SAME DATASET

We performed a comprehensive comparison and analysis of various models using the identical dataset, as detailed in Table 10. The results show that our proposed model has a significant advantage in terms of accuracy (ACC), false positive rate (FPR), true positive rate (TPR), and F1 score.

Specifically, our model achieved an accuracy of 0.9732, which is significantly higher than the other compared models. This reflects that our model can more accurately classify and identify anomalous behavior in network traffic data. Furthermore, our model has an FPR of only 0.0219, indicating it has a very low probability of misclassifying normal traffic as anomalies. Additionally, our model also performed excellently in TPR and F1 score, reaching 0.9682 and 0.9727, respectively, further ensuring its efficiency in anomaly detection tasks.

In contrast, the other compared models exhibit varying degrees of limitations in these evaluation metrics.

TABLE 10. Comparison of models on the same dataset.

Model	ACC	FPR	TPR	F1
GA-RF [9]	0.867	-	0.986	0.891
CossimMFO [11]	0.924	0.030	0.921	0.942
SigmoidMFO [11]	0.931	0.043	0.923	0.941
BWO-BOA-KNN [15]	0.963	-	0.936	0.952
GIWRF-DT [23]	0.9301	-	0.9476	0.9372
Stacking-DT [24]	0.94	0.06	0.94	-
Our model	0.9732	0.0219	0.9682	0.9727

In particular, some models have relatively high false positive rates (FPR), which can result in a high rate of false alarms. This indicates that the TDBO-ACBLT model effectively identifies optimal hyperparameters and features, showcasing robust generalization capabilities. The efficiency of feature selection arises from its combination with random forests, making important features more readily accessible. The synergy of TDBO's powerful search capabilities with ACBLT's robust spatiotemporal feature extraction contributes to the model's formidable generalization prowess.

VI. CONCLUSION

In this study, we introduce a model named TDBO-ACBLT, which combines ACBL and TCN with the TDBO algorithm. The ACBLT model effectively captures spatiotemporal features of network traffic data, utilizing the enhanced Dung Beetle Optimization algorithm (TDBO) for hyperparameter tuning, and incorporating TDBO and random forests for feature selection. This innovative approach integrates diverse neural network architectures with an advanced population optimization algorithm, addressing complex intrusion detection problems. The design goal of our proposed model is to achieve higher accuracy and lower false positive rates. Experimental results on the UNSW-NW15 dataset demonstrate superior performance compared to other feature selection methods, yielding more outstanding features. In comparison to other population optimization algorithms, the optimized Dung Beetle Optimization (TDBO) algorithm exhibits higher efficiency and accuracy. Furthermore, our model outperforms traditional machine learning methods, with an increased accuracy of 1.6% reaching 97.32% after feature selection. In the ablation experiments, as the model complexity increases, its performance consistently improves. In summary, our model holds significant value in various aspects, making it a noteworthy contribution to the field.

Future research will focus on the following directions: (1) Applying the proposed method to more complex scenarios, including systems with large-scale datasets and a high proportion of irrelevant features, to validate its generalization capability. (2) Exploring new optimization techniques to enhance Dung Beetle Optimization (DBO), thereby further improving the model's performance and efficiency.

REFERENCES

[1] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[2] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," *Measurement*, vol. 154, Mar. 2020, Art. no. 107450.
[3] 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.
[4] A. Dahou, M. A. Elaziz, S. A. Chelloug, M. A. Awadallah, M. A. Al-Betar, M. A. A. Al-Qaness, and A. Forestiero, "Intrusion detection system for IoT based on deep learning and modified reptile search algorithm," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–15, Jun. 2022.
[5] Y. Yin, J. Jang-Jaccard, W. Xu, A. Singh, J. Zhu, F. Sabrina, and J. Kwak, "IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset," *J. Big Data*, vol. 10, no. 1, pp. 1–26, Feb. 2023.
[6] A. Nazir and R. A. Khan, "A novel combinatorial optimization based feature selection method for network intrusion detection," *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102164.
[7] N. Saran and N. Kesswani, "A comparative study of supervised machine learning classifiers for intrusion detection in Internet of Things," *Proc. Comput. Sci.*, vol. 218, pp. 2049–2057, Jan. 2023.
[8] N. Kunhare, R. Tiwari, and J. Dhar, "Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection by genetic algorithm," *Comput. Electr. Eng.*, vol. 103, Oct. 2022, Art. no. 108383.
[9] S. M. Kasongo, "An advanced intrusion detection system for IIoT based on GA and tree based algorithms," *IEEE Access*, vol. 9, pp. 113199–113212, 2021.
[10] 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.
[11] M. Alazab, R. A. Khurma, A. Awajan, and D. Camacho, "A new intrusion detection system based on Moth–Flame optimizer algorithm," *Expert Syst. With Appl.*, vol. 210, Dec. 2022, Art. no. 118439.
[12] S. Dwivedi, M. Vardhan, and S. Tripathi, "An effect of chaos grasshopper optimization algorithm for protection of network infrastructure," *Comput. Netw.*, vol. 176, Jul. 2020, Art. no. 107251.
[13] M. Zivkovic, L. Jovanovic, M. Ivanovic, N. Bacanin, I. Strumberger, and P. M. Joseph, "XGBoost hyperparameters tuning by fitness-dependent optimizer for network intrusion detection," in *Communication and Intelligent Systems*. Singapore: Springer, 2022, pp. 947–962.
[14] A. S. Talita, O. S. Nataza, and Z. Rustam, "Naïve Bayes classifier and particle swarm optimization feature selection method for classifying intrusion detection system dataset," *J. Phys., Conf. Ser.*, vol. 1752, no. 1, Feb. 2021, Art. no. 012021.
[15] H. Xu, Y. Lu, and Q. Guo, "Application of improved butterfly optimization algorithm combined with black widow optimization in feature selection of network intrusion detection," *Electronics*, vol. 11, no. 21, p. 3531, Oct. 2022.
[16] Q. M. Alzubi, M. Anbar, Y. Sanjalawe, M. A. Al-Betar, and R. Abdullah, "Intrusion detection system based on hybridizing a modified binary grey wolf optimization and particle swarm optimization," *Expert Syst. Appl.*, vol. 204, Oct. 2022, Art. no. 117597.
[17] S. V. Pingale and S. R. Sutar, "Remora whale optimization-based hybrid deep learning for network intrusion detection using CNN features," *Expert Syst. Appl.*, vol. 210, Dec. 2022, Art. no. 118476.
[18] A. K. Balyan, S. Ahuja, U. K. Lilhore, S. K. Sharma, P. Manoharan, A. D. Algarni, H. Elmannai, and K. Raahemifar, "A hybrid intrusion detection model using EGA-PSO and improved random forest method," *Sensors*, vol. 22, no. 16, p. 5986, Aug. 2022.
[19] 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.
[20] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[21] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[23] R. A. Disha and S. Waheed, "Performance analysis of machine learning models for intrusion detection system using Gini impurity-based weighted random forest (GIWRF) feature selection technique," *Cybersecurity*, vol. 5, no. 1, p. 1, Dec. 2022.
[24] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, "A tree-based stacking ensemble technique with feature selection for network intrusion detection," *Appl. Intell.*, vol. 52, no. 9, pp. 9768–9781, 2022.

YUE LI received the Ph.D. degree in computer science from the University of Limerick, in 2010. He was the Team Leader of the Smart Grid End User System Project, Research Center, Ningbo Sanxing Electric Company Ltd. He is currently an Assistant Professor with the School of Computer Science and Technology, Donghua University. His current research interests include security in the IoT, key agreement protocol in wireless sensor networks, software development, data security, big data, and blockchain technology.

JIALE ZHANG received the B.S. degree in computer science from Donghua University, in 2022, where he is currently pursuing the degree with the Department of Computer Science. His research interests include information security and the utilization of deep learning for security detection.

YITING YAN is currently pursuing the master's degree with the School of Computer Science and Technology, Donghua University. Her current research interests include federated learning and blockchain.

YUTIAN LEI is currently pursuing the degree with the School of Computer Science and Technology, Donghua University. Her research interests include knowledge graph and natural language processing.

CHANG YIN received the B.S. degree in computer science from Donghua University, in 2022. He is currently pursuing the master's degree with the Donghua University of Computer Science and Technology, Shanghai, China. His research interest is blockchain technology.