A Novel Lightweight IoT Intrusion Detection Model Based on Self-Knowledge Distillation

Zhendong Wang, *Member, IEEE*, Renqiang Zhou, Shuxin Yang, Daojing He, *Member, IEEE*, and Sammy Chan, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) environment contains many different types of devices, each with different functionalities, communication protocols, and security capabilities, which makes the IoT a complex challenge for security protection. Therefore, network intrusion detection (NID) is needed to detect intrusions in the network to secure the IoT. In recent years, deep learning (DL)-based intrusion detection systems have achieved excellent results, but they tend to require high-computational resources and storage space, which is not feasible for most IoT devices. In this article, we propose a lightweight intrusion detection model based on self-knowledge distillation (SKD), namely, tied block convolution lightweight deep neural network (TBCLNN), which improves the detection accuracy while also reducing the number of model parameters and computational cost. Specifically, we use the binary Harris Hawk optimization algorithm (bHHO) for dimensionality reduction of traffic features. We use lightweight convolution, such as tied block convolution (TBC), to design lightweight neural network (LNN) models with residual and inverse residual structures. Moreover, we propose an improved SKD loss function to solve the sample imbalance problem and compensate for the performance degradation caused by lightweight neural networks. The multi-classification accuracy of our proposed method exceeds 99% on all three publicly available IoT datasets. The experimental results show that our method has a small model size and requires only low-computational resources, making it suitable for resource-constrained IoT intrusion detection.**

***Index Terms*—Internet of Things (IoT), intrusion detection, lightweight neural network (LNN), self-knowledge distillation (SKD).**

I. Introduction

THE Internet of Things (IoT) is currently in a phase of rapid development, and its application areas have widely penetrated various fields, such as smart homes, teleworking, healthcare, education, transportation, and industrial control [1]. With the continuous maturity of IoT technology and the expansion of application scenarios, the number of IoT devices is also growing rapidly. The number of IoT devices worldwide is expected to continue to increase in the coming years. This growth is not only driving the expansion of the IoT technology services market, but also opening up unprecedented opportunities across industries. However, the current state of IoT security is also facing serious challenges. Currently, there is a lack of harmonized access standards for IoT devices between the cloud, the device side, and the user's operating side, which has led to an increase in potential security risks, such as hacking, data leakage, and privacy invasion [2]. In addition, the characteristics of the IoT industry chain are that it is too long, and the diversity of devices also increase the difficulty of security management. Specifically, the security vulnerabilities of the IoT are mainly reflected in the leakage of sensitive user information and malicious code implantation. These vulnerabilities can be remotely exploited by malicious attackers through advanced cyberattack techniques to carry out a variety of illegal activities [3], such as stealing personal information, disrupting the normal operation of the system, or even launching cyberattacks. Since IoT devices involve many fields, their security is directly related to personal privacy, corporate interests and national security. Therefore, in-depth study of the security and privacy risks faced by the IoT and exploration of effective countermeasures have become cutting-edge and urgent research topics in the field of cybersecurity. Intrusion detection system (IDS) is the most commonly used network security system in existing networks. IDS are mainly used for real-time monitoring of network traffic and system events, detecting possible intrusions, identifying and analyzing known intrusion patterns and attack methods, responding to and blocking attacks in real time, and generating reports and log records.

There are two main types of IDS: 1) anomaly-based IDS and 2) signature-based IDS [4]. Anomaly-based IDSs detect anomalous behavior primarily by modeling normal system behavior. Deviations from normal behavioral patterns are detected through continuous analysis of system and network activity. This method is able to detect unknown attacks because it does not rely on known attack characteristics. On the other hand, signature-based IDSs perform match detection based on the characteristics of known attacks. It maintains a database of various known attack signatures and matches network traffic or system behavior against these signatures. A match may

Received 23 July 2024; revised 11 October 2024; accepted 20 January 2025. Date of publication 23 January 2025; date of current version 23 May 2025. This work was supported in part by the Natural Science Foundation of China under Grant 62062037, Grant 61562037, and Grant 72261018; and in part by the Natural Science Foundation of Jiangxi Province under Grant 20212BAB202014 and Grant 20171BAB202026. *(Corresponding author: Renqiang Zhou.)*

Zhendong Wang, Renqiang Zhou, and Shuxin Yang are with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China (e-mail: wangzhendong@jxust.edu.cn; 6120230744@mail.jxust.edu.cn; yangshuxin@jxust.edu.cn).

Daojing He is with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China (e-mail: hedaojinghit@163.com).

Sammy Chan is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong (e-mail: eeschan@cityu.edu.hk).

Digital Object Identifier 10.1109/JIOT.2025.3533092

2327-4662 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply.

indicate that an attack may have occurred. This method can accurately detect known attacks but may not be able to cope with new types or variants of attacks. Therefore, anomaly-based IDSs with the ability to recognize unknown attacks are widely used in intrusion detection systems.

In recent years, extensive research has been conducted on the detection performance of deep learning (DL) in intrusion detection systems [5], [6]. Compared to traditional machine learning (ML) models, DL models greatly improve the robustness and accuracy of the models. However, some DL models usually require a large amount of computational resources for training and inference, while IoT devices are usually edge devices with limited computational resources and limited communication and storage capabilities [7]. Therefore, DL models with high-computational overhead are not suitable for deployment on IoT devices. In current research, the main techniques for compressing DL models are model parameter quantization [8], weight pruning [9] and knowledge distillation (KD) [10]. Among them, KD can preserve the performance of the original model to a large extent in the process of model compression, and quantization and pruning, although they can also achieve model compression, may have some impact on the model performance. However, standard KD relies on large external teacher models, which require large computational resources and storage space, and there may be situations where some of the knowledge from the teacher model is not fully applicable to the student model. To solve the above problems, we use self-knowledge distillation (SKD) [11] to train lightweight models, which avoids the dependence on large external pretrained teacher models and enables the models to continuously optimize themselves through self-learning to achieve performance improvement.

Based on the above analysis, we propose a tied block convolution lightweight deep neural network (TBCLNN) model for network intrusion detection (NID), which is a stack of residual and inverse residual blocks composed of tied block convolution (TBC) and tied block squeeze and excitation modules (TiedSE) [12], and better learns the intrinsic patterns and features in the data via SKD. To reduce the complexity and computation of the model while achieving higher accuracy and real-time anomaly detection, the computational complexity of the intrusion detection model proposed in this article is reduced by the following three main strategies. 1) For feature selection, we adopt the binary Harris Hawk optimization algorithm (bHHO), which has the advantages of a wide search range and fast convergence speed and is able to efficiently screen out the optimal feature subset of the dataset, thus avoiding the use of redundant or irrelevant features and reducing the computational complexity of the model. 2) In terms of model compression, we use SKD to enable the model to learn more compact and efficient feature representations, thus reducing the model size and computational complexity. 3) In terms of optimizing the convolutional neural network architecture, the TBC neural network is used instead of the traditional convolutional neural network, which reduces the number of parameters and computations. On the CIC-IDS2017,TON_IoT and BOT_IoT datasets, our proposed model has significant advantages over traditional DL methods and several state-of-the-art models in terms of most performance metrics.

The main contributions of this article to the field of intrusion detection include the following:

1) We propose a lightweight intrusion detection model based on SKD. Through SKD, the model can learn more internal information and knowledge representations, which helps to enhance the robustness and stability of the model. To the best of our knowledge, few studies have used SKD for intrusion detection.

2) We utilize TBC and TiedSE to construct a lightweight residual block and a lightweight inverse residual block, respectively, and propose the TBCLNN model for IoT intrusion detection. Our method improves computational efficiency and extracts features efficiently while reducing model complexity by extending and compressing the structure.

3) The bHHO is introduced for feature selection, which can utilize its advantages of global search, local optimization, diversity and collaboration to find representative and complementary feature combinations quickly and efficiently, improve classification performance and reduce computational overhead.

4) To address the category imbalance problem that exists in multicategorization tasks, we propose an improved progressive self-knowledge distillation (IPSKD) loss function based on progressive self-knowledge distillation (PSKD) [11], which allows us to pay more attention to hard-to-categorize samples during model training.

The remainder of this article is organized as follows. Section II discusses the techniques related to IoT intrusion detection. Section III proposes a modeling framework for KD for IoT intrusion detection. Section IV describes the experimental setup and dataset as well as the preprocessing procedure in detail. Section V discusses the methodology evaluation and analysis of the experimental results. Section VI summarizes the research results.

II. Related Work

In this section, we provide an overview of ML methods and DL methods used in previous research on IoT intrusion detection.The comparative summary of existing approaches for IDS is presented in Table I.

A. ML Methods for IoT

Thaseen and Kumar [13] proposed a method based on the fusion of principal component analysis (PCA) and an optimized support vector machine (SVM), which optimizes the parameters and kernel functions of SVM by introducing an automatic parameter selection mechanism, thus reducing the computational complexity and training time of the model. Azimjonov and Kim [14] proposed a lightweight IDS based on a ridge regressor and stochastic gradient descent classifier (SGDC), where the most relevant and effective subset of features is selected by a fine-tuned ridge regressor and the fine-tuned SGDC is used as a detection model. The system not only improves the detection accuracy but also reduces the computational time complexity. Roy et al. [15] proposed

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

TABLE I
COMPARATIVE SUMMARY OF EXISTING APPROACHES FOR IDS

Ref.	Year	Methodology used	Dataset used	Result	Advantages	Limitations
[13]	2014	PCA & SVM	KDD-CUP-99	Acc = 99%	High accuracy low resource consumption	Tested with one dataset Outdated datasets used.
[14]	2024	SGDC	KDD-CUP-99 BotIoT-2018 N-BaIoT-2021	(KDD)Acc = 98.42% (BotIoT)Acc = 94.76% (N-BaIoT)Acc = 98.42%	Highly accurate lightweight design	Dependent on feature selection long feature selection time.
[15]	2022	B-Stacking	CIC-IDS2017 NSL-KDD	(CIC)Acc = 99.11% (NSL)Acc = 98.50%	High detection rate low overhead	The ability to detect unbalanced datasets and specific attack types still needs further research and validation.
[16]	2024	CRFC	NSL-KDD UNSW-NB15	(NSL)Acc = 99.957%	High accuracy low false alarm rate	Computationally complex dependent on feature selection.
[17]	2024	LSVMs	KDD-CUP-99 BotIoT-2018 N-BaIoT-2021	(KDD)Acc = 95.66% (BotIoT)Acc = 99.48% (N-BaIoT)Acc = 99.81%	Highly accurate lightweight design	dependent on feature selection.
[18]	2022	Chi2 & Smote & XGBoost	ToN-IoT	Acc = 98.3%	Highly accurate	High computational overhead
[5]	2022	ResNet model based on deep transfer learning	Real-world datasets	-	High accuracy efficiency reliability	Complex models high computational resource requirements.
[19]	2022	E-graphsage	BoT-IoT ToN-IoT	(BoT)Recall = 99.99% (ToN)Recall = 86.78%	High accuracy efficiency	Complex models
[20]	2024	iDetector & EdgeNet	Real-world datasets	Acc = 90%	Real-time lightweight design	Dependent on feature selection high computational resource requirements.
[21]	2023	DIS-IoT	ToN-IoT CIC-IDS2017 SWaT	(ToN)Acc = 99.7% (CIC)Acc = 98.7% (SWaT)Acc = 99.6%	High accuracy low false alarm rate	high computational resource requirements.
[22]	2024	KerPCA & CHbO & Dugat-LSTM	TON-IOT NSL-KDD	(ToN)Acc = 98.76% (NSL)Acc = 96.98%	High accuracy low false alarm rate	Complex models high computational resource requirements.
[23]	2021	PCA & LNN	UNSW-NB15 Bot-IoT	(UNSW)Acc = 86.11% (Bot)Acc = 96.15%	Low model complexity small model size	Low generalizability susceptibility to interference its accuracy in detecting multicategorization cases is lower than that of other complex network model.
[24]	2024	improved BERT-of-Theseus based on Knowledge Distillation	ToN-IoT CIC-IDS2017	(ToN)Acc = 99.7% (CIC)Acc = 99.7%	High accuracy low false alarm rate lightweight design	Over-reliance on teacher models.
[6]	2022	TCNN based on Knowledge Distillation	NSL-KDD CIC-IDS2017	(NSL)Acc = 98.44% (CIC)Acc = 99.44%	High accuracy low computational overhead strong stability	Over-reliance on teacher models High demand for computing resources.

an integrated learning model based on the boosting and stacking (B-Stacking) algorithm as an intrusion detection model by identifying the most critical features used for intrusion detection by eliminating multicollinearity, sampling and PCA dimensionality reduction methods. Under the B-Stacking framework, KNN, random forest (RF) and XGBoost are selected as the underlying weak learners, where XGBoost is used as the boosting learning algorithm. The B-Stacking model is not only high in detection rate and low in false positives but also lightweight enough to be deployed on IoT nodes with limited power and storage resources. Pramilarani and Kumari [16] proposed a cost-based random forest classifier (CRFC). This method uses the maximum variance bound to remove the noise values from the input data and then improves the RF classifier by using the cost matrix calculated from the feature importance, which shows advantages in dealing with high traffic and a high-false alarm rate in IoT networks, and can effectively reduce false alarms and accurately detect intrusions even when there is considerable traffic in the network, which improves the recall rate of the IDS. Azimjonov and Kim [17] designed a lightweight IDS for IoT networks utilizing a fine-tuned linear support vector machine (LSVM) model and feature selection techniques. The feature selection methods used in this approach include importance coefficient based, forward order, backward order, and correlation coefficient-based methods, and the results of these feature selection methods are used to train the LSVM model to detect various attacks on IoT networks. Gad et al. [18] proposed a distributed IDS based on ML. For feature engineering, some attributes that may cause overfitting are removed, and Chi2 and Correlation Matrix methods are applied to select the most valuable features. The system is designed as a distributed architecture that combines the advantages of cloud and fog computing to improve detection efficiency and response speed. While integrated methods like XGBoost perform well, they may be slightly less capable of handling highly nonlinear and complex data patterns compared to DL models. DL methods are able to automatically extract high-level feature representations from raw data to better capture complex relationships in the data.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

B. DL Methods for IoT

While previous research has shown that ML has achieved significant results in IoT intrusion detection, traditional ML algorithms typically rely on hand-designed feature extractors, whereas DL models are able to automatically learn feature representations from the raw data, which gives DL a significant advantage when dealing with high-dimensional, complex data, as data generated by IoT devices tend to have high-dimensional and nonlinear characteristics. Additionally, after training, DL models are able to respond more quickly to new data than are ML models, which is critical for IoT environments that require real-time monitoring and protection.

Mehedi et al. [5] proposed a trusted IDS based on deep migration learning for the IoT, which obtained the P-ResNet network model by migrating the pretrained ResNet network model to obtain datasets from seven IoT sensors and achieved high accuracy. Lo et al. [19] proposed a graph neural network-based IDS for the IoT, where a network graph is constructed from the source IP address, source port, destination IP address, and destination port, using a neural network model, that is, composed of two E-GraphSAGE layers, which permits the capture of the edge features of the graph and the topology information, which enhances feature extraction. Kong et al. [20] utilized deep separable convolution and self-attention mechanisms to design a lightweight deep neural network model called iDetector, which uses a nonlinear feature transformation (NFT) algorithm to increase the information entropy of the samples, thereby improving the classification performance. Experimental evaluation shows that iDetector outperforms most of the current DL-based solutions in terms of classification performance and has faster classification speed on resource-constrained edge gateways. Lazzarini et al. [21] proposed a new approach called Deep Integrated Stacking for the IoT (DIS-IoT) for intrusion detection in IoT environments. The approach is based on stacked integration of DL models. Specifically, DIS-IoT combines four different DL models (MLP, DNN, convolutional neural networks (CNN), and long short-term memory (LSTM)) into a single fully connected DL layer to form a single, independent integrated model.DIS-IoT was evaluated on three open-source datasets: 1) ToN_IoT; 2) CICIDS2017; and 3) SWaT, and demonstrated high accuracy in both binary and multiclassification tasks. However, integrating several different types of models increases the complexity of the overall system. Devendiran and Turukmane [22] used data cleaning, M-squared normalization and extended synthetic sampling methods to process the data. Feature extraction is performed by Kernel-assisted principal component analysis (KerPCA) and optimal features are selected using chaotic honey badger optimization (CHbO). Finally, the network traffic is classified by gated attention dual long short term memory (Dugat-LSTM). Dugat-LSTM as a DL model, especially combining LSTM and attention mechanisms, may have higher computational complexity and longer training time, which may limit its application in resource-constrained environments.

Zhao et al. [23] proposed a lightweight neural network (LNN)-based intrusion detection method for the IoT, in which the LNN model consists of two lightweight units, which realize the downsampling function by lightweight unit A and feature extraction and classification function by lightweight unit B. The method designs the NID loss function to solve the dataset imbalance problem, which improves the accuracy. However, due to the simple network architecture of the LNN model, it may not be robust enough to process the data and easily interferes with noise or outliers, and its accuracy in detecting multicategorization cases is lower than that of other complex network models. Wang et al. [24] proposed a lightweight IoT intrusion detection model based on improved BERT-of-Theseus, which uses the Siamese network to reduce the feature dimension of complex high-dimensional network traffic data, and then adopts KD technology. The Vision Transformer large model serves as the teacher model to guide the Poolformer small model training. The final small model contains only approximately 788 parameters, which reduces the number of parameters by approximately 90% compared with the large model, significantly reducing the number of parameters in the final model and making it more lightweight. This model not only enables efficient intrusion detection in resource-constrained IoT environments, but also performs well in terms of accuracy and detection speed. Wang et al. [6] proposed a lightweight NID method for industrial physical information systems based on KD and depth metric learning. In this method, the binary version of the gray wolf optimization algorithm was adopted to select the optimal feature subset, triplet convolutional neural network was used as the teacher model, and depth separable convolutional neural network was used as the student model. A robust model loss function is designed to improve the stability of model training, and a K-fold cross-training method is introduced to enhance the accuracy of anomaly detection. This model not only greatly reduces the calculation cost and model size, but also achieves significant advantages in detection accuracy.

From the above DL methods, it can be found that most DL models are complex with high-computational complexity and long training time, which limits their application in resource-constrained environments. Lightweight deep neural network models can break this limitation, but they are susceptible to noise interference due to their lightweight model architecture that leads to defective detection performance. Although standard KD can reduce the complexity of DL models and improve the accuracy of student models by training student models with teacher models to obtain lightweight models, the selection of teacher and student models is more complicated, and the teacher models also need to take up some storage space. In contrast, our proposed method lightens the model structure by using TBC and provides gain information to compensate for the performance drawbacks caused by model lightening through a SKD strategy, without allocating additional memory to store the teacher model. The design of the model fully considers the limited computational resources and restricted memory space of IoT devices, which not only improves the accuracy of IoT attack detection, but also has high robustness and generalizability.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

III. Methodology

In this section, we will delve into the components of the proposed lightweight SKD NID method and discuss in detail how these components can be fused to construct an efficient IDS. The whole system can be divided into the following five core components: 1) feature selection; 2) TBC; 3) TBCLNN model based on TBC; 4) IPSKD; and 5) intrusion detection steps for TBCLNN modeling.

A. Feature Selection

Feature selection plays a crucial role in NID. With the continuous development of network technology, network traffic data is growing explosively, which contains a large number of features. However, not all features contribute to the intrusion detection task and some may be redundant, irrelevant or even noisy. Performing feature selection can reduce the data dimensionality, improve the detection efficiency and reduce the number of input dimensions of the model. There are several common feature selection methods: filtered feature selection, embedded feature selection, and wrapped feature selection. Among them, the filtered approach has higher computational efficiency, but may ignore the interrelationships between features. The embedded method embeds the feature selection process into the training process of the model, which can take into account the efficiency of feature selection and model training, but the method usually can only select features with linear relationships. If there are nonlinear relationships between features in the data, the embedded method may not be able to select these features efficiently. Therefore, in this article, we use wrapped feature selection, a method that requires multiple training and evaluation steps of the model, but usually selects a subset of features with better performance and greater relevance.

The Harris Hawk optimization (HHO) algorithm [25] has the advantages of diversity, collaboration, simplicity, and adaptivity, and is able to adaptively adjust the search strategy according to the prey's escape energy, which is gradually transitioning from the exploration phase to the exploitation phase. This adaptivity allows the algorithm to strike a balance between global and local search, and allows the algorithm to efficiently search for the optimal subset of features in the entire feature space and avoid falling into local optimal solutions. Since the algorithm has only two solutions for selecting a subset of features, we use the bHHO [26] as a wraparound feature selection algorithm.

The HHO algorithm simulates several typical behavioral patterns exhibited by a group of falcons during the hunting process, such as searching, encircling, and raiding, and translates them into an optimization-seeking strategy in the search space. The HHO algorithm can be divided into three phases: 1) the exploration phase; 2) the exploration and exploitation conversion phase; and 3) the exploitation phase. The steps of the HHO algorithm are shown below.

Step 1 (Exploration Phase): In the vast, sprawling territory of southern Arizona, Harris's hawks are challenged with long stretches of watching, observing, and tracking prey, which causes them to exhibit an extremely high degree of dispersal within the group. Individual eagles each occupied a number of random spots and probed the surrounding area for potential prey resources based on two strategies: 1) when q $<$ 0.5, Harris's hawks perch according to the location of other members and prey and 2) when $q \geq 0.5$, Harris's hawks randomly perch on a tree within the population's range. Equation (1) is the mathematical representation of the two strategies

$$X(t+1) = \begin{cases} (X_{\text{rand}}(t) - r_1)|X_{\text{rand}}(t) - 2r_2X(t)|, & q \geq 0.5 \\ (X_{\text{rabbit}}(t) - X_{\text{ave}}(t)) - r_3(lb + r_4(ub - lb)), & \text{else} \end{cases} \tag{1}$$

where q, $r1$, $r2$, $r3$, and $r4$ are random numbers in [0,1]; ub and lb are the upper and lower bounds on the search space, respectively; $X_{\text{rand}}(t)$ is the position vector of a randomly selected falcon from the population in iteration t; $X(t+1)$ is the eagle's position vector in the next iteration t; $X(t)$ is the falcon's current position vector; $X_{\text{rabbit}}(t))$ is the prey position vector; and $X_{\text{ave}}(t))$ is the average position vector of all individuals within the population.

Step 2 (Transition From Exploration to Exploitation): Since the prey escape energy E decreases significantly as it attempts to evade pursuit, the Harris Hawk adjusts its ability to hunt strategies and states in real time based on the level of escape energy exhibited by the prey as it flees. Equation (2) is the mathematical calculation of the escape energy E

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{2}$$

where E is the escape energy of the prey and E_0 is the initial energy of the prey. During each iteration, E_0 varies randomly in the interval $(-1, 1)$.

Step 3 (Exploitation Phase): When Harris hawks begin a surprise attack on their prey, the prey often escapes one step ahead of the Harris hawk. Therefore, the Harris hawk has evolved four attack strategies based on the prey's escape behavior and its own pursuit strategy. Assuming that r is the probability of the prey escaping, when r $<$ 0.5, the escape is successful, and when $r \geq 0.5$, the escape fails.

1) Soft Besiege: When $r \geq 0.5$ and $|E| \geq 0.5$, the prey still has enough energy, and the Harris hawk adopts a soft siege strategy. Equation (3) is the mathematical representation of the soft besiege strategy

$$X(t+1) = X_{\text{rabbit}}(t) - X(t) - E|J \cdot X_{\text{rabbit}}(t) - X(t)| \tag{3}$$

where $J = 2*(1-r_5)$ is the random jump distance of the prey during the escape process and r_5 is a random number within [0, 1].

2) Hard Besiege: When $r \geq 0.5$ and $|E| < 0.5$, Harris Hawk adopts a hard siege strategy. Equation (4) is the mathematical representation of the hard besiege strategy

$$X(t+1) = X_{\text{rabbit}}(t) - E|X_{\text{rabbit}}(t) - X(t)|. \tag{4}$$

3) Soft Besiege With Progressive Rapid Dives: When $r <$ 0.5 and $|E| > 0.5$, the prey has a chance to escape, and the escape energy is sufficient; then the Harris Hawk adopts the soft siege strategy of progressive fast dive. Equations (5)–(7)

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

are mathematical representations of the soft besiege strategy of progressive rapid dives

$$Y = X_{\text{rabbit}}(t) - E|J \cdot X_{\text{rabbit}}(t) - X(t)| \tag{5}$$

$$Z = Y + S * LF(D) \tag{6}$$

$$X(t+1) = \begin{cases} Y, \text{ if } F(Y) < F(X(t)) \\ Z, \text{ if } F(Z) < F(X(t)) \end{cases} \tag{7}$$

where F is the fitness function, D is the dimension of the problem, S is a random vector of size $1 \times D$, and LF is the Lévy flight function.

4) Hard Besiege With Progressive Rapid Dives: When $r < 0.5$ and $|E| < 0.5$, the prey has a chance to escape, but the escape energy is insufficient; thus, the Harris hawk adopts the hard-surrounding strategy of progressive fast dive. Equations (6)–(8) are mathematical representations of the hard besiege strategy of progressive rapid dives

$$Y = X_{\text{rabbit}}(t) - E|J \cdot X_{\text{rabbit}}(t) - X_{\text{ave}}(t)|. \tag{8}$$

Since there are only two results of feature selection, the continuous HHO algorithm solution needs to be converted to binary. Referring to the method in [23], (9) and (10) are the mathematical representations for converting the continuous positions of the Harris Hawk to binary positions

$$S\left(x_i^j(t)\right) = \frac{1}{e^{-x_i^j(t)}} \tag{9}$$

$$X_i^j(t) = \begin{cases} 0, \text{ if rand} < S\left(X_i^j(t)\right) \\ 1, \text{ else} \end{cases} \tag{10}$$

where $x_i^j(t)$ denotes the velocity of particle i at dimension j in iteration t. All velocity values are converted to probability values in the range [0, 1]. The binary position of the Harris hawk is updated using an S-type transfer function with the update (10).

The fitness function plays a very important role in feature selection, which is responsible for quantitatively evaluating the effectiveness and applicability of a selected subset of features for the target learning task. In this article, the fitness function is referred to the fitness function in [27]. Equation (11) is the mathematical representation of the fitness function

$$\text{Fitness} = \nu \cdot \text{error} + (1 - \nu) \cdot \frac{F}{N} \tag{11}$$

where *error* denotes the classification error rate of a given classifier, F denotes the number of selected features, N denotes the total number of features, and $\nu \in [0,1]$ denotes the importance of classification quality. This fitness function can be used to evaluate the merit of the selected feature subset; the smaller the fitness is, the better the classification performance of the feature subset. In the feature selection in this article, we use a convolutional neural network as a classifier and utilize the network traffic from the training set for feature selection, using (11) as the fitness function. The pseudocode of the bHHO algorithm used in this article is shown in Algorithm 1.

Algorithm 1 Pseudocode of BHHO

Inputs: The population size N and maximum number of iterations T
Outputs: The location of rabbit and its fitness value
Initialize the random population $X_i (i = 1, 2, \ldots, N)$
t← 0(initialize)
while $(t < T)$ **do**
  Binary conversion of X_t using Eq. (10)
  Calculate the fitness values of hawks
  Set X_{rabbit} as the location of rabbit (best location)
  for each hawk (X_i) **do**
    Update the initial energy E_0 and jump strength J
    Update the E using Eq. (2)
    if $(|E| \geq 1)$ **then**
      Update the location vector using Eq. (1)
    else if $(|E| < 1)$ **then**
      if $(r \geq 0.5$ and $|E| \geq 0.5)$ **then**
        Update the location vector using Eq. (3)
      else if $(r \geq 0.5$ and $|E| < 0.5)$ **then**
        Update the location vector using Eq. (4)
      else if $(r < 0.5$ and $|E| \geq 0.5)$ **then**
        Update the location vector using Eq. (7)
      else if $(r < 0.5$ and $|E| < 0.5)$ **then**
        Update the location vector using Eq. (7)
      end if
    end if
  end for
  Binary conversion of X_{rabbit} using Eq. (10)
end while
return X_{rabbit}

B. Tied Block Convolution

TBC, innovatively proposed by Wang and Stella [12], is a novel form of convolution operation, that is, similar in design to group convolution (GC) [28], i.e., it involves dividing the input features into multiple groups and performing the standard convolution (SC) operation independently within the groups. However, TBC is unique in that it introduces a mechanism for sharing parameters between groups, an innovation that greatly improves the parametric efficiency and expressive power of the model.

The core concept of TBC is to uniformly divide the C channels of the input features to form B nonoverlapping blocks, each containing C/B channels. Next, for each block, TBC uses SC for internal processing to ensure that the feature information within the block is fully mined and integrated. Crucially, TBC cleverly utilizes a single-block filter defined only on the C/B channels and shares the parameters of this filter across the blocks to generate B independent convolutional responses.

To understand this in an intuitive way, comparing the workings of SC and GC with TBC, as shown in Fig. 1, the SC filter spans the entire C channel, forming a global feature mapping, whereas in TBC, when B = 2 is set, a single-block filter covers only C/2 channels, and although its field-of-view is somewhat scaled down, the filter is able to produce two separate blocks each, thanks to the parameter-sharing mechanism responses, which are essentially GC shared across groups. When B takes the value of 1, the characteristics

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

of the TBC degenerate into a SC, where the single-block filter actually covers all C channels, and the operation is equivalent to a SC operation without block division.

Suppose the input features are $X \in \mathbb{R}^{c_i \times h \times w}$, the output feature is $\hat{X} \in \mathbb{R}^{c_0 \times h_0 \times w_0}$, where c is the number of channels, and h and w are the height and width of the feature map, respectively. The size of the convolution kernel is $k \times k$. For a clearer description, the assumptions here ignore the bias term.

SC definition formula is shown in

$$\hat{X} = X * W \tag{12}$$

where $W \in \mathbb{R}^{c_0 \times c_i \times h \times w}$ denotes the convolution kernel. The number of its parameters is calculated as shown in

$$\text{num params} = c_0 \times c_i \times k \times k. \tag{13}$$

GC is the process of splitting input features into G features with equal numbers of channels, and each group shares convolution kernel parameters, i.e., standard convolutions within the group, and then concatenates the outputs. The definition formula of GC is shown in

$$\hat{X} = X_1 * W_1 \oplus X_2 * W_2 \oplus \cdots \oplus X_G * W_G \tag{14}$$

where $\oplus$ denotes the splicing operation. W_i is the convolutional kernel information for each group, where $W_i \in \mathbb{R}^{(c_0/G) \times (c_i/G) \times h \times w}$, $\quad i \in \{1, \ldots, G\}$. The number of its parameters is calculated as shown in

$$\text{num params} = G \times (c_0/G) \times (c_i/G) \times k \times k. \quad . \tag{15}$$

TBC defining formula is shown in

$$\hat{X} = X_1 * W' \oplus X_2 * W' \oplus \cdots \oplus X_G * W' \tag{16}$$

where $W' \in \mathbb{R}^{(c_0/B) \times (c_i/B) \times h \times w}$. Its parametric quantity is calculated as shown in

$$\text{num params} = (c_0/B) \times (c_i/B) \times k \times k. \tag{17}$$

From (13), (15), and (17) and Fig. 1, it can be intuitively found that the number of parameters of the TBC are all much smaller than those of SC and GC, and Wang and Stella [12] noted that the TBC has only one fragmentation on GPU utilization, whereas the GC has G fragmentations, which considerably reduces the degree of parallelism. Therefore, the lightweight nature of the TBC construct makes it ideal for building efficient intrusion detection models, especially for node devices deployed in IoT networks with relatively limited computing power. By embedding this innovative convolutional technique, the intrusion detection model can significantly reduce computational resource requirements while maintaining excellent detection performance, thus empowering IoT nodes to achieve fast and accurate intrusion detection and effectively improve the security protection of the entire network.

Wang and Stella [12] further extended the idea of Tied Block Filtering to the field of Fully Connected Layer and innovatively proposed the tied block fully Connected Layer (TFC). This new fully connected layer design also introduces the concept of blocks and implements interblock parameter sharing within the input channels, thus continuing the advantages of bundled block filtering in reducing model complexity and computational overhead at the fully connected layer level. Like TBC, TFC reduces the parameters by a factor of B^2 and the computational cost by a factor of B. The Tied Block Filtering idea was also incorporated into the design of attention modules, developing the TiedSE. The TiedSE module, on the other hand, is an improvement and optimization of the SE module, which is also dedicated to improving the performance of the model under limited resources.

C. TBCLNN Model Based on Tied Block Convolution

Faced with the unique challenge of deploying intrusion detection systems in IoT devices, i.e., ensuring the real-time operation of the detection system while maintaining a high level of intrusion detection accuracy with limited computational resources, traditional solutions are often caught in a dilemma: either choosing ML algorithms that have low-computational complexity but limited detection accuracy and suffering from the resulting potential risks, or forcibly adopting computationally intensive DL models, which may lead to a serious decline in system performance and cannot even be effectively deployed in resource-constrained IoT environments. To overcome this dilemma, this article proposes an innovative lightweight NID model that skillfully integrates the powerful representation learning capability of DL with the lightweight design principle, aiming to achieve accurate and real-time detection of intrusion behaviors, while ensuring efficient and stable operation on IoT devices.

As shown in Fig. 2, the lightweight NID model for the IoT mainly consists of two kinds of lightweight blocks, namely, lightweight inverse residual block and lightweight residual block. Next, we will introduce the TBCLNN model in detail.

To make the model lightweight, we are inspired by [6] to use depth separable convolution (DSConv) instead of traditional 1-D convolution to initialize the convolutions. As shown in Fig. 2(b), DSConv reduces the amount of computation and the number of parameters by decomposing the traditional convolution into two steps: 1) depth convolution and 2) dot convolution. In the deep convolutional layer, separate convolution operations are performed for each input channel. This means that for an input feature map with C channels, the deep convolutional layer applies a different convolution kernel for each channel. In the point convolutional layer, the size of the convolutional kernel is usually 1x1, which linearly combines the outputs of the deep convolutional layers to generate a new output feature map. The role of the dot convolutional layer is to mix the information from different channels to obtain a richer representation of the features. Therefore, compared to the 1-D convolutional initialization convolutional layer, DSConv can largely reduce the amount of computation and the number of parameters, which is very advantageous for resource-limited IoT devices. In addition, due to the reduction in the number of parameters, the model is easier to train and easier to avoid overfitting.

The lightweight residual block consists of a residual structure, as shown in Fig. 2(a), where the main path first groups the input channels by a certain block size (B) using the TBC introduced in Section III-B and performs the convolution

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

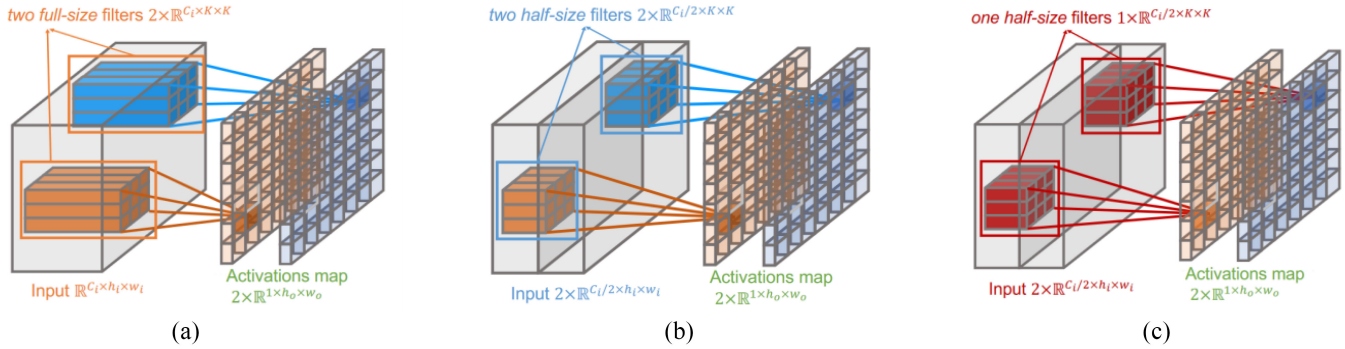


Fig. 1. Comparison of SC and GC with TBC. To generate two activation maps, SC requires two full-size filters, and GC requires two half-size filters, whereas TBC requires only one half-size filter, i.e., the parameters are reduced by a factor of four [12]. (a) SC. (b) GC. (c) TBC.

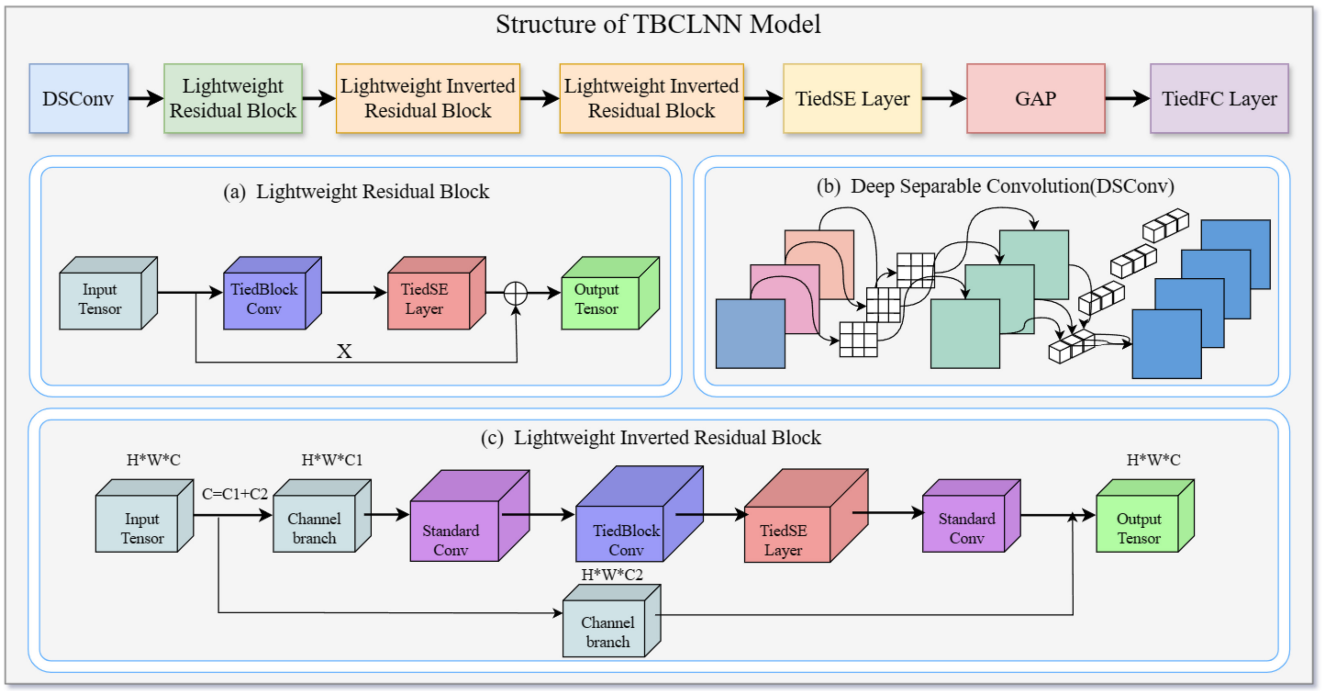


Fig. 2. Proposed framework for the IoT intrusion detection TBCLNN model.

operation independently within each group. The convolutions within each subgroup are independent of each other, which helps to capture local spatial features while reducing the coupling between parameters, but introduces a parameter-sharing mechanism between different channel blocks to increase group-to-group connections and reduce the number of parameters. TiedBlockConv2d also sets an optional interchannel block dropout probability, which randomly discards a portion of the channel block's information during the training process to prevent overfitting and enhance the model generalization ability. After the convolution of the main path, a TiedSE layer is embedded, which enhances the channel selectivity of the network by global average pooling (GAP) of the input features, and then learning to obtain the importance weights of each channel through a two-layer fully connected network, then applying these weights to the input features. Subsequently, the feature tensor obtained from the main path and the input tensor are merged and connected, and the hardswish activation function is applied to enhance the expressiveness of the network, which helps to improve the expressiveness and training stability of the model.

The lightweight inverse residual block draws on the width expansion strategy to implement dynamic expansion and compression in the channel dimension to optimize the network structure and save computational resources. As shown in Fig. 2(c), the module uniformly divides the input feature channel into two parts, where one channel branch first applies 1×1 convolution to perform a channel expansion operation to expand the initial lower dimensional feature mapping to a higher channel space to facilitate the capture of a richer feature representation. Next, Branch I uses TiedBlockConv2d on the augmented channels for feature extraction, which effectively reduces the computational complexity and model capacity through the corresponding parameter optimization. After this, the TiedSE layer is introduced to perform inter-channel attentional adjustment of the feature map, which

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

empowers the model to dynamically emphasize or suppress specific feature channels by learning the importance weights of each channel. After completing the channel reweighting, Branch I again performs a channel compression operation via 1x1 convolution to ensure that the number of channels in its output feature map is exactly equal to half of the final number of required output channels. Moreover, Branch II adopts a constant mapping strategy, i.e., if the condition is that the number of input channels is equal to the number of output channels and the stride is 1, the original input features are passed directly; otherwise, they are also adjusted by appropriate 1x1 convolution to match the number of output channels and the spatial dimension. Finally, the results of these two branches are spliced and merged in the channel dimension, and together they form the output features of the lightweight inverse residual block, which achieves the effective capture and transmission of high-level feature information while keeping the model lightweight.

To improve the model's ability to select the importance of individual channels in the input features, we again incorporated a TiedSE layer after two lightweight stacks of inverse residual structures. The purpose of this design is to highlight valuable information and weaken redundant or noncritical feature expressions by intelligently adjusting the weight distribution of different channels. Considering that the fully connected layer may increase the number of parameters, which affects the lightweight property of the model, we discard the traditional fully connected layer for feature dimensionality reduction, and instead adopt GAP, which is able to efficiently map high-dimensional features to low-dimensional vector representations without the need for a large number of additional parameters. After the above channel attention adjustment and feature dimension compression, we further use the TiedFC layer for the final classification or regression prediction. The TiedFC layer also adheres to the lightweight design principle by splitting the input features into multiple chunks to be processed in parallel to achieve efficient computation while maintaining the lightweight nature of the model at the channel level. This set of design improvements helps the overall model significantly reduce computational complexity and model size while maintaining high performance.

D. IPSKD

KD is a technique for transferring knowledge from one model (teacher model) to another (student model). The core idea is to enable a small, more computationally efficient student model to learn and reproduce the knowledge and behavior of a large, complex teacher model. This process goes beyond simple parameter replication and captures the decision boundary and complexity of the teacher model to some extent by allowing the student model to learn the output probability distribution of the teacher model. Compared to KD, SKD allows the model to learn a richer representation of features through self-direction without the need for additional selection of the teacher model, which can eliminate the training step for the teacher model, simplify the overall training process and resource requirements, and enable the model to learn a smoother probability distribution.

Inspired by [11], we adopt the PSKD strategy, which improves the generalization performance of the model with the help of an internal self-iterative mechanism. Specifically, the core of this strategy is that each training cycle treats the model of the previous epoch as the teacher model, while the model of the current epoch acts as the student, drawing on the prediction experience accumulated during previous training as an informative additional instructional signal to participate in the current epoch of learning. By continuously looping this process, the model is able to not only absorb the real label information of the standard training set in successive iterations, but also fully utilize the knowledge structure embedded in its own historical predictions, thus obtaining better generalizability and prediction accuracy. The distillation step for IPSKD is as follows.

First, we adopt the suggestion of Hinton et al. [10] by taking each input sample x and the k-dimensional target label y as well as the prediction vector $z(x) = [z_1(x), \ldots, z_k(x)]$ generated by the model, outputting the predicted probabilities via the Softmax function and using temperature adjustment to soften the predicted distribution probabilities

$$p_i(x;\tau) = \frac{\exp(z_i(x)/\tau)}{\sum_j \exp\left(z_j(x)/\tau\right)} \tag{18}$$

where τ is the temperature scale used to soften the probability distribution for better distillation.

After obtaining the predicted distribution probabilities for the previous epochs and the current epoch, among all previous models of teacher candidates, we use the model for epoch t-1 as the teacher model because it provides the most valuable information among the candidates. Specifically, the hard label at epoch t and the predictive distribution probability at epoch t-1 are softened to $((1-\alpha)y+\alpha P^S_{t-1}(x)$, and then the training loss of the predictive distribution probability of the soft label and the present epoch is computed by the standard cross-entropy loss function. By continuously optimizing the training loss for soft labels, the model is able to gradually strengthen its generalization ability and stable performance during the learning process. The loss function is shown in

$$L_{\text{PSKD}} = H\Big((1-\alpha_t)y + \alpha_t P^S_{t-1}(x), P^S_t(x)\Big) \tag{19}$$

where H is the standard cross-entropy loss function, α is the hyperparameter, y is the hard label of epoch t, $P^S_{t-1}(x)$ is the predicted value of epoch t-1, and $P^S_t(x)$ is the predicted value of epoch t. The parameter α is the key point of the self-distillation loss function, which represents the degree of our trust in the teacher's model knowledge in (19).

In the traditional KD framework, α is generally assigned a constant value throughout the training process because the teacher model remains static. In the PSKD technique, it is necessary to dynamically adjust the reliability of the teacher model because the model does not yet fully understand the data in the early stages of training. As training advances, we need to gradually increase the value of α to reflect the

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

growth of the model's understanding of the data as training progresses. Similar to the learning rate tuning strategy, there exist multiple schemes for incrementing α by epoch, such as step-up, exponential growth, or linear growth patterns. To simplify the hyperparameter configuration and reduce its complexity, we adopt a linear growth approach to show the variation in α, ensuring that the model properly weighs the existing knowledge with the newly learned information at different stages of training. The equation for calculating α at epoch t is as follows:

$$\alpha_t = \alpha_T \times \frac{t}{T}. \tag{20}$$

To solve the problem of the standard dataset, which suffers from extreme category imbalance as well as difficult to categorize samples, we introduce the focal loss [29] strategy on the basis of the PSKD strategy. Focal Loss is an adaptive scheme for cross-entropy loss, whose core feature is the use of a dynamic adjustment factor that can reduce the focus on those easy-to-classify samples during the training process in real time, which in turn quickly directs the model to focus on attacking more difficult and less discriminatory sample instances. In this way, Focal Loss can effectively improve the training efficiency and final classification performance of the model when dealing with the category imbalance problem. The Focal Loss calculation equation is as follows:

$$FL(p_t) = -\beta_t(1-p_t)^{\gamma}\log(p_t) \tag{21}$$

where p_t is the predicted value of a category of training samples, β_t is the weight given to different categories of samples, which can be increased for fewer positive samples, and γ is the key adjustment factor. When the model's prediction value for a sample is already quite high, indicating that the sample is an easy-to-categorize sample, the mechanism of action of β_t and γ reduces the contribution of the loss function for that sample accordingly. Specifically, the β_t and γ coefficients reduce the model's learning pressure on samples that already have a high level of classification confidence through a dynamic regulation mechanism, thus shifting the training focus more on samples that are difficult to differentiate or have a high probability of classification error.

Combining the PSKD technique and the Focal Loss principle, we design an IPSKD loss function, aiming at more efficiently balancing the two key aspects of the model training process: 1) the information of difficult-to-categorize samples and 2) the reuse of the model's own knowledge. The improved SKD loss function is shown in (22). We introduce the tradeoff parameters μ and σ, which each assume an important role. μ serves as a tradeoff coefficient responsible for regulating the weighting between the hard-labeled information and the model's self-distilled knowledge, ensuring that the model does not deviate from the essential characteristics of the data while fully absorbing its own deep insights during the learning process. σ regulates the allocation of the model's attention to hard-to-categorize samples, which, through dynamic adjustment, enables the model to flexibly allocate learning resources in different training phases based on the difficulty of the samples. This comprehensive strategy can not only accelerate model convergence, but also improve the accuracy and robustness of the model in complex tasks

$$L_{\text{IPSKD}} = \mu \cdot L_{\text{PSKD}} + \sigma \cdot FL(p_t). \tag{22}$$

The overall process of model SKD is shown in Fig. 3 model training via the self-distillation section.

E. Intrusion Detection Steps of TBCLNN Model

The specific process of the TBCLNN intrusion detection model proposed in this article can be summarized as follows.

Step 1: The data containing characters in the dataset are numerically processed, and then the dataset as a whole is subjected to data normalization and data imbalance processing. The specific operations are described in detail in Section IV-C.

Step 2: The bHHO algorithm is used to filter the most discriminative and influential subset of optimal features from the large intrusion detection dataset.

Step 3: Based on the optimal feature subset screened in the previous step, we further train the TBCLNN model using IPSKD until the TBCLNN model converges and the optimal network model is selected. The learning efficacy and generalization ability of the TBCLNN model during the training process are further optimized by incorporating the advantages of targeted feature selection, and gradually enhancing the model's ability to internalize and transfer existing knowledge.

Step 4: The preprocessed test dataset is fed into the trained optimal network model to run, thereby obtaining the classification results for each network traffic.

The overall flowchart of the TBCLNN model is shown in Fig. 3.

IV. Experimental Setup And Preprocessing

A. Experimental Setup

The hardware platform used for this experiment is a Windows 11 laptop with the following specifications: an Intel Core i5-9300H 9th generation Core processor, 8 GB of RAM, and a 512-GB ROM SSD for storage. In addition, the laptop also integrates an NVIDIA GeForce GTX 1650 GDDR6 memory 8-GB discrete graphics card, which provides strong support for the training and inference of DL models.

B. Dataset

In this study, we validate the effectiveness of the TBCLNN model for intrusion detection in IoT security by conducting intrusion detection experiments on the IoT benchmark datasets CIC-IDS2017 [30], TON_IoT [31], and BOT_IoT [32], as well as dividing the dataset into 80% for training data and 20% for test data, where in the training data are in turn divided into 20% as validation data.

The CIC-IDS2017 dataset was released by the Canadian Institute for Cybersecurity Research in 2017 and is known for its comprehensiveness and real-time nature. While earlier datasets, such as KDDCUP99 and NSLKDD, had limitations in terms of timeliness, traffic diversity, coverage of attack types, and information integrity due to data anonymization, the CIC-IDS2017 dataset is a breakthrough in both design and

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

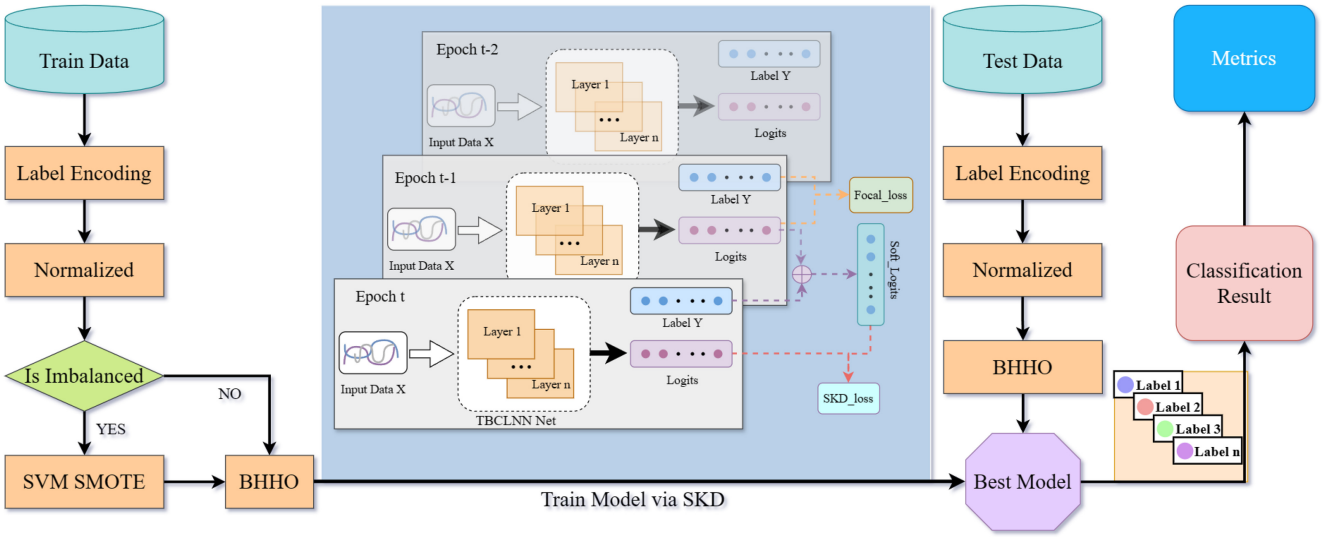


Fig. 3. Flowchart of TBCLNN model.

TABLE II
NETWORK TRAFFIC DISTRIBUTION FOR CIC-IDS2017

Category	Training Set	Validation Set	Test Set
Benign	281,391	70,328	87,964
GoldenEye	6,639	1,668	1,986
Hulk	147,165	36,833	46,126
Slowhttptest	3,557	887	1,055
Slowloris	3,740	908	1,148
Total	442,492	110,624	138,279

TABLE III
NETWORK TRAFFIC DISTRIBUTION FOR TON_IOT

Category	Training set	Validation set	Test set
Normal	191877	48016	60107
Backdoor	12884	3171	3945
Ddos	12780	3174	4046
Dos	12857	3155	3988
Injection	12760	3258	3982
Password	12768	3242	3990
Ransomware	12771	3258	3971
Scanning	12841	3157	4002
Xss	12858	3183	3959
Mitm	671	153	219
Total	295067	73767	92209

collection. The dataset includes but is not limited to Brute Force FTP, Brute Force SSH, DDoS attacks, Heartbleed, Web attacks, Infiltration, and Botnet, ensuring that the experimental results are more realistic. CIC-IDS2017 stores approximately 3 million network traffic records in eight separate CSV files, each consisting of an exhaustive 78 feature attributes and their corresponding labels. To effectively minimize the model training cycle and meet the requirements of a multiclass classification task, we chose the Wednesday-workingHours.csv file for our experiments. Due to the relatively sparse number of Heartbleed attack samples in this dataset (only 11), we chose to temporarily exclude such samples in this study to focus on other richer and more representative attack types. To mitigate overfitting and improve model generalizability, we distribute the remaining data into training, validation, and test sets. The network traffic distribution of the CIC-IDS2017 dataset is shown in Table II.

TON_IoT dataset was created by the Cyber Range and IoT Lab at the University of New South Wales in Canberra. The dataset is composed of heterogeneous data sources collected from telemetry data of IoT/IIoT services, as well as operating system logs and network traffic of IoT networks, which include nine types of cyber-attacks are Scanning, DoS, DDoS, Ransomware, Backdoor, Injection, XSS, Password, and MITM. The dataset has two main folders "Processed_datasets" and "Train_Test_datasets" and the datasets in the files are in CSV file format. In this study, the Train_Test_Network dataset in the "Train_Test_datasets" file is selected as a suitable dataset for training and testing the IDS in the IoT environment. The network traffic distribution of the TON_IoT dataset is shown in Table III.

The BOT_IoT dataset, created in 2018 and released in 2019 by the Cyber Range Lab at the Canberra campus of the University of New South Wales, is a modern and authentic dataset. The dataset is available in multiple forms of source files, covering raw pcap grab files, files generated by the argus tool, and files in csv format for easy analysis and processing. It contains normal IoT network traffic and four attack scenarios, including DoS, DDoS, reconnaissance, and theft. Since there are too many traffic records in the original CSV file of this dataset, we randomly filtered some of the traffic records through the pandas library for experimental evaluation, which included approximately 600 000 traffic records. The filtered dataset contains a total of 43 independent features, which we assign to the training, validation and test sets. The network traffic distribution of the BOT_IoT dataset is shown in Table IV.

C. Data Preprocessing

Since there may be errors, duplicate or irrelevant information, and missing values in the dataset, we need to preprocess the data before applying the traffic data to the

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

TABLE IV
NETWORK TRAFFIC DISTRIBUTION FOR BOT_IOT

Category	Training set	Validation set	Test set
Benign	297	70	110
DoS	220830	54921	69184
DDoS	107846	27396	33730
Reconnaissance	58529	14484	18069
Theft	46	17	16
Total	387548	96888	121109

model. The preprocessing step is critical, and by carefully cleaning and organizing the data, we can ensure that the model receives high-quality, reliable inputs, leading to more accurate and reliable analysis results. The preprocessing steps are as follows.

1) Character Data Numericization: Numericalization of character data is the process of converting nonnumerical character data into numerical data. Due to the limited memory resources of IoT devices, when there are more categories, tag encoding only takes up one number as an identifier, while single thermal encoding produces a binary vector with the same length as the number of categories, which significantly increases the feature dimensions, so we use tag encoding to numericalize the character data. Taking the BOT_IoT dataset as an example, where the prototype feature has five attributes, which are TCP, UDP ARP, ICMP, and IPV6-ICMP, these five attributes are processed as 0, 1, 2, 3 and 4, respectively. Similarly, other character-based data in the dataset are quantized.

2) Data Normalization: Since the scales of different features may vary greatly, without normalization, the differences in the scales of these features can have a significant impact on the training process of the ML model, which may result in the model focusing too much on those features with a large range of values and ignoring the features with a smaller range, but which may be just as important, so we need to normalize the data. The main normalization operation commonly used today is min-max scaling, but it is sensitive to outliers, and if extreme outliers exist, scaling can greatly affect the data distribution [33]. Therefore, we adopt the use of the hybrid data preprocessing method proposed by [6], whose normalization method is shown in

$$\bar{x}_i = \begin{cases} \frac{x_i - x_{i_{\min}}}{x_{i_{\max}} - x_{i_{\min}}}, & \text{if } x_{i_{\max}} < 10 \\ \log_{10}(1 + x_i), & \text{if } x_{i_{\max}} > 10 \text{ and } x_{i_{\min}} > -1 \end{cases} \tag{23}$$

where $\bar{x}_i$ denotes the normalized value of the ith feature, x_i denotes the original value of the ith feature, $x_{i_{\min}}$ denotes the minimum value of the same feature, and $x_{i_{\max}}$ denotes the maximum value of the same feature.

3) Sample Imbalance Processing: As we can clearly observe from Tables III and IV, the percentage of specific types of attack events, such as Mitm attacks in the TON_IoT dataset, is extremely low, at only 0.23% of the overall samples. This significant category imbalance leads to a tendency for the classifier to learn and predict categories with a greater number of samples during the training process, which increases the probability of misjudging attacks of a few categories (e.g., Mitm attacks), which ultimately manifests itself in a higher false alarm rate. Therefore, effectively addressing the category imbalance of the dataset is crucial to improve the accuracy of classifiers in detecting various categories of attacks. To address the above problem of a high-false alarm rate triggered by data category imbalance, we adopt the SVMSMOTE algorithm [34] to implement an oversampling strategy for attack categories with a small number of samples in the training set. To prevent the model from relying too much on artificially generated data in the evaluation phase, we only perform the oversampling technique for the training set and do not use the same oversampling technique for the test set session, thus ensuring the model's ability to generalize to new data and the accuracy of the truthful performance evaluation.

V. EXPERIMENTAL RESULTS

A. Evaluation Metrics

To objectively measure the performance of our proposed model on specific tasks and help us understand its predictive ability, we use Accuracy, Recall, Precision and F1-Score as evaluation metrics for intrusion detection models. The definition formulas for each evaluation indicator are as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{24}$$

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}} \tag{25}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{26}$$

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{27}$$

where true positive (TP) denotes the number of positive samples correctly predicted as positive; true negative (TN) is the number of negative samples correctly predicted as negative; false positive (FP) is the number of negative samples incorrectly predicted as positive; and false negative (FN) is the number of positive samples incorrectly predicted as negative.

In addition, we evaluate the computational cost of FLOPs, parameter count, and model size to compare the required implementation resources.

B. Classification Performance of TBCLNN Model

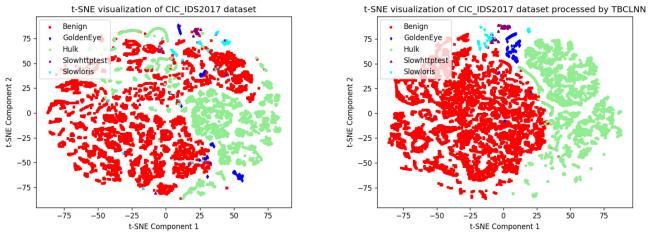
In this study, we use the bHHO algorithm to reduce the dimensionality of input features and enhance the intrusion detection capability of the model. We apply the method to three original datasets, CIC-IDS2017, TON_IoT and BOT_IoT. Among them, the feature dimensions of the CIC-IDS2017 dataset decreased from 78 to 32, the feature dimensions of the TON_IoT dataset decreased from 43 to 24, and the feature dimensions of the BOT_IoT dataset decreased from 43 to 15. Subsequently, we put the dimensionality reduced datasets into the TBCLNN model and classify them using the SKD strategy. Table V shows the classification results of each dimensionality reduced dataset in the TBCLNN model.

To visually demonstrate the superior efficacy of our proposed model in multiclassification tasks, we randomly selected 30 000 network traffic samples from the data

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

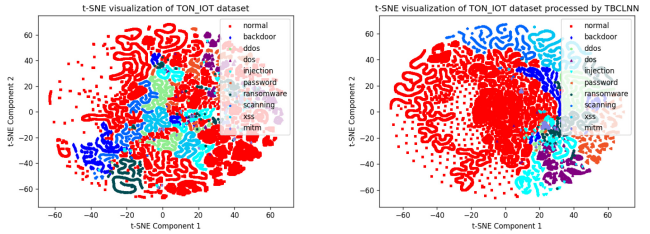
TABLE V
TBCLNN MODEL MULTICLASSIFICATION RESULTS

Dataset	Accuracy	Precision	Recall	F1-score
CIC-IDS2017	99.71%	99.72%	99.71%	99.70%
TON_IoT	99.10%	99.13%	99.10%	99.11%
BOT_IoT	99.92%	99.94%	99.92%	99.93%



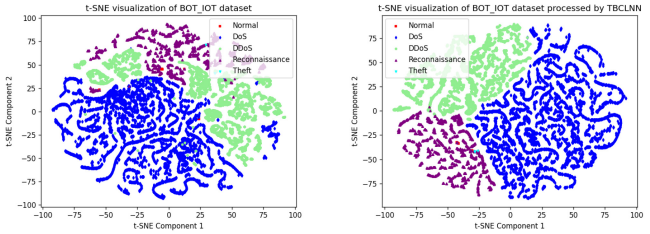
(a) (b)

Fig. 4. t-SNE visualization of CIC-IDS2017. (a) CIC-IDS2017. (b) Processed CIC-IDS2017.



(a) (b)

Fig. 5. t-SNE visualization of TON_IoT. (a) TON_IoT. (b) Processed TON_IoT.



(a) (b)

Fig. 6. t-SNE visualization of BOT_IoT. (a) BOT_IoT. (b) Processed BOT_IoT.

and tested them with the optimized TBCLNN model. Subsequently, we transform the processing results into easy-to-understand visualized images with the help of the t-SNE algorithm [35]. As a control, we also applied the t-SNE algorithm to the same set of 30 000 samples without any treatment to gain insight into the differences before and after treatment. The visualization results cover three important datasets: 1) CIC-IDS2017; 2) TON_IoT; and 3) BOT_IoT, which are displayed in Figs. 4–6, respectively.

In particular, as shown in Fig. 4(a), in the unprocessed CIC-IDS2017 dataset, different types of attack data are obfuscated with the distribution of Normal traffic data, e.g., GoldenEye, Slowloris, and Slowhttptest are promiscuously interspersed with the Normal traffic and Hulk, which shows a high degree of similarity and overlap. In contrast, Fig. 4(b) demonstrates the view of the data after being optimized by our proposed model processing method, and it can be found that the differentiation between the various categories of attacks has significantly improved. More explicit and well-defined clusters are formed between processed Normal traffic, Hulk, and GoldenEye, effectively revealing the essential differences among the data categories. In Fig. 5(a), we can find that the data distribution of the TON_IoT dataset is more complex than that of the CIC-IDS2017 dataset, which contains more attack classes, in which normal traffic is intertwined with a variety of attack modes, which constitutes a significant obstacle to identification and increases the difficulty of classification. In contrast, the processed view of the data in Fig. 5(b) shows significantly improved data visualization, with clearly identifiable boundaries emerging between normal traffic and various types of malicious attacks, and each class of attack activity tends to self-aggregate into well-defined clusters. Particularly noteworthy is the fact that things, such as backdoor, XSS, and password, are effectively differentiated after processing, and they are self-contained at the edges of other data clusters, with significant clustering effects. For the BOT_IoT dataset, the original data distribution state in Fig. 6(a) appears to be rather confusing; in particular, the blurred boundaries between DDoS attacks and DoS and Reconnaissance attacks are deeply intertwined with each other, which undoubtedly exacerbates the difficulty of categorization, and makes it extremely difficult to effectively differentiate these attack types directly from the data. However, a significant optimization of the data distribution pattern can be clearly observed in Fig. 6(b). DDoS, DoS and Reconnaissance are no longer intermingled, but each forms a clear, independent cluster, and the boundaries between them become clearly distinguishable.

The transformation of the data view distribution before and after the processing of the three datasets not only reflects the substantial improvement in the data structuring capability of the processing, but also strongly demonstrates that our proposed TBCLNN model, which effectively facilitates the differentiation between different attack types, even in datasets containing highly complex and overlapping attack patterns, such as TON_IoT, achieves an orderly separation of the data, thus greatly enhancing the accuracy and efficiency of multiclassification experiments.

To more intuitively and exhaustively illustrate the multi-classification performance of our model on each dataset, we employ the powerful visualization tool of multiclassification ROC curves. In Fig. 7, the model proposed in this study demonstrates excellent generalization ability on the CIC-IDS2017 dataset, with the area under the ROC curve reaching 0.99 and above for all types of attacks. Its macro-mean area under the ROC curve reaches 0.99, which highlights the extremely high-classification accuracy and sensitivity. In Figs. 8 and 9, the model also maintains high performance on both datasets, TON_IoT and BOT_IoT. Each class of attacks in both datasets is recognized accurately, and the area under the ROC curve for each class of attacks tends to be 1.00, and the two macro-averaged area under the ROC curve reaches 1.00 and 0.99, respectively, which further confirms the stability and high efficiency of the model in different data environments. These results collectively show that our model

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

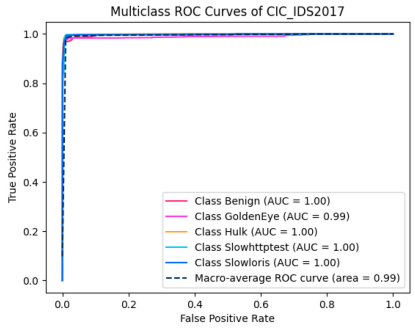


Fig. 7. Multiclass ROC curves of CIC-IDS2017.

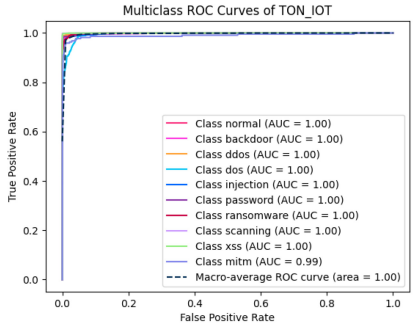


Fig. 8. Multiclass ROC curves of TON_IoT.

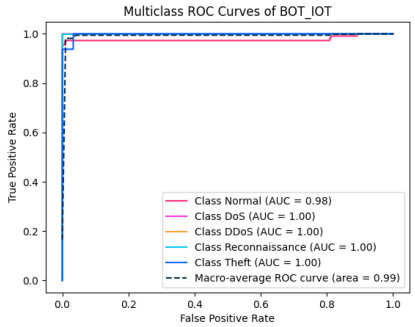


Fig. 9. Multiclass ROC curves of BOT_IoT.

is not only able to accurately distinguish between categories in a multiclassification task, but also has good generalizability and robustness.

C. Ablation Study

To systematically validate the efficacy and superiority of our proposed model and its built-in SKD strategy, we carefully designed a series of ablation experiments for three major datasets. These experiments focused on the influence of three core elements on intrusion detection performance: whether the bHHO optimization algorithm is applied for feature selection, whether the SKD technique is employed, and the difference between PSKD and our proposed IPSKD. Through this series of rigorous experiments, we analyze the contribution of each key technology component to the overall performance of the model, and then gain insight into its uniqueness in improving the accuracy and efficiency of intrusion detection.

Fig. 10 shows the discernible effect of the impact of each implemented key technology component on the intrusion

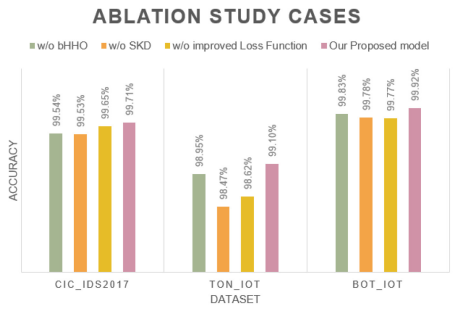


Fig. 10. Ablation study results.

TABLE VI
ABLATION STUDY ON CIC-IDS2017

Cases	Accuracy	Precision	Recall	F1-score
w/o bHHO	99.54%	99.55%	99.54%	99.54%
w/o SKD	99.53%	99.53%	99.53%	99.53%
w/o IPSKD	99.65%	99.66%	99.65%	99.65%
Our Proposed model	**99.71%**	**99.72%**	**99.71%**	**99.70%**

TABLE VII
ABLATION STUDY ON TON_IOT

Cases	Accuracy	Precision	Recall	F1-score
w/o bHHO	98.95%	98.98%	98.95%	98.96%
w/o SKD	98.47%	98.47%	98.47%	98.47%
w/o IPSKD	98.62%	98.61%	98.62%	98.61%
Our Proposed model	**99.10%**	**99.13%**	**99.10%**	**99.11%**

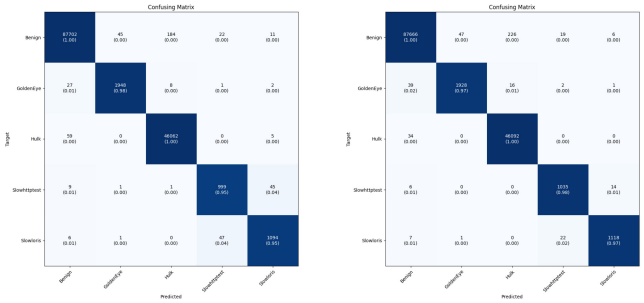
detection accuracy. According to the classification accuracy of the three datasets, our proposed method significantly outperforms the other ablation schemes. Table VI presents in detail the results of the ablation learning performed on the CIC-IDS2017 dataset, highlighting the key contributions of our method. Specifically, in terms of detection accuracy, our proposed method is 0.17%, 0.18% and 0.06% higher than the method that do not employ the bHHO algorithm for feature selection, the method that does not implement the SKD strategy, and the method that does not incorporate our improved loss function, respectively. The precision, recall and F1-score metrics of our method are also higher than those of the ablation experiments mentioned above. Tables VII and VIII show the ablation learning results for the TON_IoT and BOT_IoT datasets, respectively. The experimental results all show that our method achieves the best results in the TON_IoT and BOT_IoT datasets, and all the metrics also outperform the method that does not use the bHHO algorithm for feature selection, the method that does not implement the SKD strategy, and the method that does not adopt our improved loss function. The ablation learning results for the three datasets strongly demonstrate the ability of the components of our model to enhance the effectiveness of intrusion detection systems, especially bHHO feature selection, the SKD technique, and the improved loss function, and further validate the superiority and effectiveness of our proposed approach.

Since it is difficult to rely solely on accuracy metrics to adequately reflect the model's ability to recognize a small number of categories when faced with datasets with uneven data distributions, we turn to the confusion matrix, a more

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

TABLE VIII
ABLATION STUDY ON BOT_IOT

Cases	Accuracy	Precision	Recall	F1-score
w/o bHHO	99.83%	99.90%	99.83%	99.86%
w/o SKD	99.78%	99.80%	99.78%	99.79%
w/o IPSKD	99.77%	99.77%	99.77%	99.77%
Our Proposed model	**99.92%**	**99.94%**	**99.92%**	**99.93%**



(a) (b)

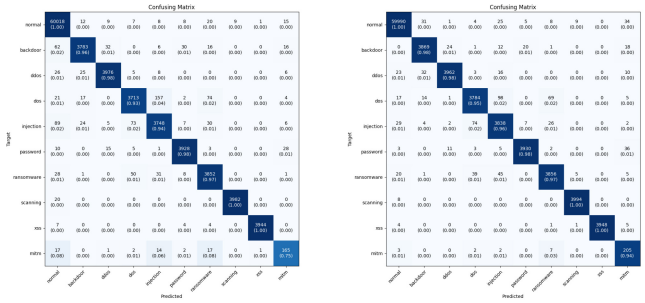
Fig. 11. Confusion matrix for different loss on CIC-IDS2017. (a) W/o IPSKD. (b) Our proposed model.

fine-grained and intuitive evaluation tool, to provide insights into the effectiveness of our improved version of the loss function. The detailed analysis of the confusion matrix enables a clear picture of the distribution of correct and incorrect predictions for each category, especially when dealing with those categories with small sample sizes.

Fig. 11 illustrates the ablation study comparison performed for the CIC-IDS2017 dataset, aiming to evaluate the impact of the improved version of the loss function on the model performance. Specifically, Fig. 11(a) presents the experimental results when the modified version of the loss function is not employed, while Fig. 11(b) reflects the performance after using the modified version of the loss function. By comparison, we can clearly observe that in Fig. 11(b), the Slowhttptest detection accuracy for the minority class of samples is significantly improved by 3% compared to that in Fig. 11(a), reaching a high level of 98%. Similarly, Sloworis detection accuracy in Fig. 11(b) increased by 2%, reaching 97%.

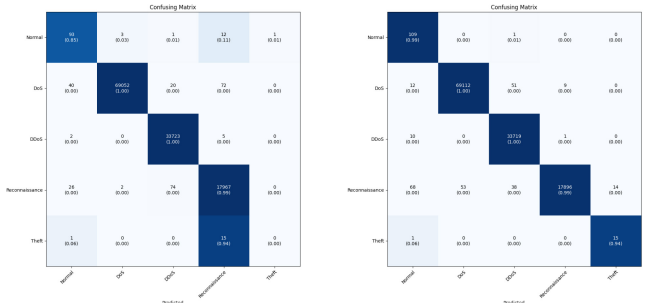
Fig. 12 then focuses on the TON_IoT dataset, and by comparing the experimental results with or without the modified version of the loss function, we find that the detection accuracy of MITM for the minority class samples in Fig. 12(b) has a significant increase of 19% compared to that in Fig. 12(a), reaching an accuracy of 94%.

Furthermore, Fig. 13 shows the results of a similar experiment for the BOT_IoT dataset. The small number of Normal traffic and Theft attack samples in this dataset, especially the fact that there are only a dozen of Theft attack samples, poses a challenge for model training. In Fig. 13(a), the model without the modified version of the loss function achieves only 85% accuracy in detecting normal traffic and fails to successfully detect any of the Theft attack samples. However, when the modified version of the loss function is applied in Fig. 13(b), the accuracy of detecting normal traffic is dramatically improved by 14%. More impressively, the



(a) (b)

Fig. 12. Confusion matrix for different loss on TON_IoT. (a) W/o IPSKD. (b) Our proposed model.



(a) (b)

Fig. 13. Confusion matrix for different loss on BOT_IoT. (a) W/o IPSKD. (b) Our proposed model.

detection accuracy of Theft attack increases from 0% to 94%, demonstrating the significant advantage of the modified loss function in coping with unbalanced datasets and minority class sample detection. All these results indicate that the IPSKD loss function plays a positive role in enhancing the model's ability to detect minority class samples.

D. Compared With Other Intrusion Detection Methods

1) Compared With the Traditional DL Methods: We validate the effectiveness of our proposed TBCLNN model by conducting comparative experiments on three datasets using traditional DL models, such as LSTM networks, CNN, and recurrent neural networks (RNN), and we use ROC curves to visualize the experimental results. To verify the superiority of the model as well as the loss function, we performed the same preprocessing operations on the datasets of the input models. Figs. 14–16 show the experimental results plotted on the CIC-IDS2017, TON_IoT and BOT_IoT datasets, respectively. By deeply analyzing and comparing the ROC curves of each model on the three datasets, we find that the TBCLNN model has the best classification effect, and its area under the ROC curve is significantly higher than that of the CNN, RNN and LSTM models on all three datasets. This also indicates the efficiency of our proposed model in detecting IoT attacks.

Table IX shows the results of the multiclassification experiments with traditional DL models on the CIC-IDS2017, TON_IoT and BOT_IoT datasets. On the CIC-IDS2017 dataset, the TBCLNN model outperforms the LSTM, RNN and CNN models by 1.23%, 1.06% and 1.17%, respectively,

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

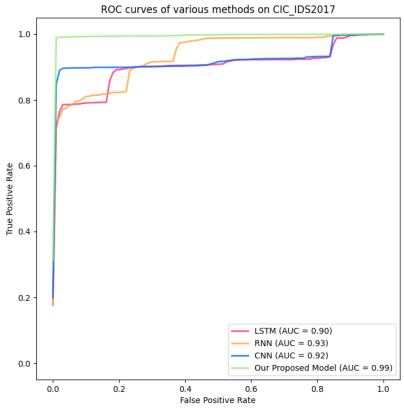


Fig. 14. ROC curves of various methods on CIC-IDS2017 dataset.

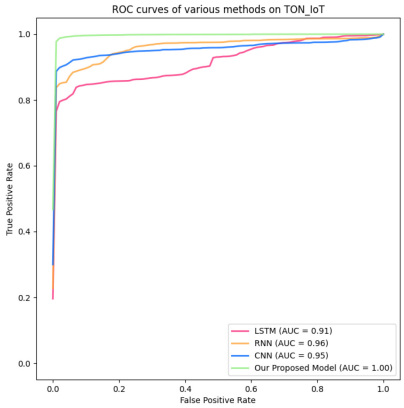


Fig. 15. ROC curves of various methods on TON_IoT dataset.

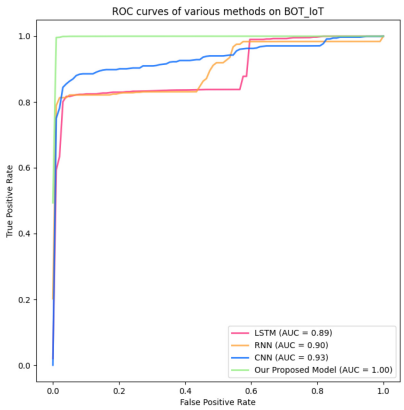


Fig. 16. ROC curves of various methods on BOT_IoT dataset.

in terms of detection accuracy. In terms of the number of participants and model size, the TBCLNN model is 82.72% and 82.76% lower than the LSTM model, 33.36% and 33.51% lower than the RNN model, and 80.75% and 80.80% lower than the CNN model, respectively. In terms of FLOPs, the TBCLNN model is 41.26% higher than the RNN model, but 11.33% and 97.17% lower than the LSTM and CNN models, respectively. On the TON_IoT dataset, the FLOPs of the TBCLNN model, although 47.39% higher than the RNN model, are 12.97% and 97.23% lower compared to the LSTM and CNN models, respectively; on the BOT_IoT dataset, the FLOPs of the TBCLNN model are only 15.23% higher than the RNN model, but 70.26% and 98.27% lower than the LSTM and CNN models, respectively. In addition, all other metrics in Table IX show better performance than the other traditional DL models, which also indicates that the TBCLNN model demonstrates a significant advantage over the LSTM, RNN and CNN models in multiclassification detection on the TON_IoT and BOT_IoT datasets. In summary, the TBCLNN model has fewer parameters and smaller model size than traditional DL models, and requires far fewer computational resources than most traditional DL models. In addition, the TBCLNN model also achieves the best performance on all three datasets in terms of accuracy, recall, precision, and F1-Score metrics.

2) Compared With the Existing DL Methods: To further show the superiority of the TBCLNN model, we also analyze it in comparison with existing work in the field of intrusion detection. The experimental results of existing studies on the CIC-IDS2017, TON_IoT and BOT_IoT datasets are given in Table X, where "−" denotes missing evaluation metrics in the original text.

From the experimental results in Table X, we can see that our method achieves better results on the CIC-IDS2017 dataset. Compared with [6], our method improves the accuracy, recall, precision and F1-Score metrics by approximately 0.27%, reduces the number of parameters by 1.2%, and reduces the model size by 1.6%, respectively. Moreover, compared to those of method [36], the number of parameters of TBCLNN model is even lower by 8.6%, indicating that our method achieves higher classification accuracy with lighter model parameters. Although the accuracy of the TBCLNN model is lower than that of method [36], it is improved by 1.02% compared to method [21]. Notably, the F1-Score of the TBCLNN model are 0.24%, 3.06% and 1.12% higher than those of the KD-TCNN, LNet-SKD and DIS-IoT models, respectively, which suggests that our method is better able to focus on samples with smaller sample sizes and classify cyberattack traffic more effectively. Compared with the LNet-SKD model, which is also a SKD model, the FLOPs of the TBCLNN model are reduced by 89.76%, which means that our approach has a much lower computational cost than the LNet-SKD model, and is more suitable for deploying on resource-constrained IoT. On the TON_IoT dataset, the TBCLNN model outperforms methods [18], [37] and [22] in terms of accuracy and F1-Score on the TON_IoT dataset, and the model size of our method is much smaller than that of the Dugat-LSTM model. On the BOT_IoT dataset, the number of parameters of the TBCLNN model, although 10.6% higher than that of the method [23], is only 10.2% of the LNN size in terms of model size, and our method improves the accuracy and F1-Score by 3.92% and 3.36%, respectively, compared with those of other methods. Compared with methods [38] and [39], the TBCLNN model improves the accuracy by 0.78% and 0.24% and the F1-Score by 1.13% and 0.42%, respectively. Moreover, the FLOPs of the TBCLNN model are reduced by 72.92% compared to those of the LNN model specifically designed for deployment on IoT devices. This shows that our method can achieve higher classification accuracy with lower spatial execution cost on the BOT_IoT dataset.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

TABLE IX
COMPARED WITH THE TRADITIONAL DL METHODS

Dataset	Model	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	Parameters	ModelSize	FLOPs
CIC-IDS2017	LSTM	98.50	98.46	98.50	98.28	26,437	103.27KB	26,944
	RNN	98.66	98.67	98.66	98.43	6,853	26.77KB	**16,912**
	CNN	98.55	98.47	98.55	98.40	23,733	92.71KB	843,010
	TBCLNN	**99.71**	**99.72**	**99.71**	**99.70**	**4,567**	**17.84KB**	23,890
TON_IoT	LSTM	94.92	94.71	94.92	94.54	19,562	76.41KB	30,592
	RNN	97.89	97.69	97.89	97.78	5,178	20.23KB	**18,064**
	CNN	97.54	97.36	97.54	97.44	24,058	93.98KB	961,837
	TBCLNN	**99.10**	**99.13**	**99.10**	**99.11**	**4,732**	**18.48KB**	26,625
BOT_IoT	LSTM	97.98	97.96	97.98	97.94	21,317	83.27KB	21,824
	RNN	98.03	98.01	98.03	97.99	5,573	21.77KB	**5,632**
	CNN	98.15	98.30	98.15	98.17	23,733	92.71KB	374,850
	TBCLNN	**99.92**	**99.94**	**99.92**	**99.93**	**4,567**	**17.84KB**	6,490

TABLE X
COMPARED WITH THE EXISTING DL METHODS

Dataset	Model	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	Parameters	ModelSize	FLOPs
CIC-IDS2017	KD-TCNN[6]	99.44	99.48	99.47	99.46	4,624	18.1KB	-
	LNet-SKD[36]	**99.90**	96.60	96.89	96.74	5,000	-	233,420
	DIS-IoT[21]	98.70	98.70	98.70	98.60	-	-	-
	TBCLNN	99.71	**99.72**	**99.71**	**99.70**	**4,567**	**17.84KB**	**23,890**
TON_IoT	XGBoost[18]	98.30	94.50	95.30	94.90	-	-	-
	DeepAK-IoT[37]	90.57	89.59	-	88.87	-	-	-
	Dugat-LSTM[22]	98.76	96.98	97.87	97.23	-	128MB	-
	TBCLNN	**99.10**	**99.13**	**99.10**	**99.11**	**4,732**	**18.48KB**	**26,625**
BOT_IoT	LNN[23]	96.15	-	-	96.68	**4,277**	181KB	23,966
	CNN-CapSA[38]	99.15	98.81	98.81	98.81	-	-	-
	IoTSecUT[39]	99.68	99.51	99.51	99.51	-	-	-
	TBCLNN	**99.92**	**99.94**	**99.92**	**99.93**	4,567	**17.84KB**	**6,490**

In summary, our model achieves better experimental results in terms of all metrics during intrusion detection on the three datasets; therefore, TBCLNN has better robustness and generalizability and better lightweight characteristics than do existing models in IoT intrusion detection, and is more suitable for resource-constrained IoT intrusion detection.

VI. CONCLUSION

In this article, we use the bHHO algorithm to downscale traffic features and propose a new lightweight intrusion detection model for resource-constrained IoT devices, namely, the TBCLNN model. We employ lightweight convolution to construct the lightweight module, which enables us to reduce the number of parameters of the model and reduce the computational cost by expanding and compressing the feature map. Moreover, more efficient feature extraction is achieved by the residual structure and inverse residual structure. Our model adopts the SKD technique, which obtains soft labels by softening the hard labels of the current epoch model and the predictions of the previous epoch model, and then continuously optimizes the model parameters by optimizing the training loss between the soft labels and the predictions of the current epoch model to improve the training efficiency and performance of the model. We propose an IPSKD loss function to train the model, which can pay more attention to hard-to-categorize samples to solve the category imbalance problem. We evaluate the proposed TBCLNN model on the CIC-IDS2017, TON_IoT, and BOT_IoT datasets, and evaluate the performance of the proposed method in terms of accuracy, precision, recall, F1-score, number of parameters, and FLOPs. The experimental results show that the model outperforms not only traditional DL models, but also excellent intrusion detection models in existing research. In conclusion, our proposed model provides a reliable solution for implementing lightweight intrusion detection models for the future IoT. In future research, we will also try to optimize our proposed model in terms of user privacy protection.

REFERENCES

[1] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1646–1685, 3rd Quart., 2020.

[2] V. Gugueoth, S. Safavat, S. Shetty, and D. Rawat, "A review of IoT security and privacy using decentralized blockchain techniques," *Comput. Sci. Rev.*, vol. 50, Nov. 2023, Art. no. 100585.

[3] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.

[4] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, 2021, Art. no. e4150.

[5] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable intrusion detection system for IoT: A deep transfer learning based approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 1006–1017, Jan. 2023.

[6] Z. Wang, Z. Li, D. He, and S. Chan, "A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning," *Expert Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117671.

[7] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42Nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, 2018, pp. 664–669.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

[8] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*.
[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[11] K. Kim, B. Ji, D. Yoon, and S. Hwang, "Self-knowledge distillation with progressive refinement of targets," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6567–6576.
[12] X. Wang and X. Y. Stella, "Tied block convolution: Leaner and better CNNs with shared thinner filters," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 10227–10235.
[13] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of PCA and optimized SVM," in *Proc. Int. Conf. Contemporary Comput. Inform. (IC3I)*, 2014, pp. 879–884.
[14] J. Azimjonov and T. Kim, "Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets," *Expert Syst. Appl.*, vol. 237, Mar. 2024, Art. no. 121493.
[15] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Gener. Comput. Syst.*, vol. 127, pp. 276–285, Feb. 2022.
[16] K. Pramilarani and P. V. Kumari, "Cost based random forest classifier for intrusion detection system in Internet of Things," *Appl. Soft Comput.*, vol. 151, Jan. 2024, Art. no. 111125.
[17] J. Azimjonov and T. Kim, "Designing accurate lightweight intrusion detection systems for IoT networks using fine-tuned linear SVM and feature selectors," *Comput. Secur.*, vol. 137, Feb. 2024, Art. no. 103598.
[18] A. R. Gad, M. Haggag, A. A. Nashat, and T. M. Barakat, "A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 6, pp. 548–563, 2022.
[19] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, 2022, pp. 1–9.
[20] X. Kong, Y. Zhou, Y. Xiao, X. Ye, H. Qi, and X. Liu, "iDetector: A novel real-time intrusion detection solution for IoT networks," *IEEE Internet Things J.*, vol. 11, no. 19, pp. 31153–31166, Oct. 2024.
[21] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowl.-Based Syst.*, vol. 279, Nov. 2023, Art. no. 110941.
[22] R. Devendiran and A. V. Turukmane, "Dugat-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy," *Expert Syst. Appl.*, vol. 245, Jul. 2024, Art. no. 123027.
[23] R. Zhao et al., "A novel intrusion detection method based on lightweight neural network for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9960–9972, Jun. 2022.
[24] Z. Wang, J. Li, S. Yang, X. Luo, D. Li, and S. Mahmoodi, "A lightweight IoT intrusion detection model based on improved BERT-of-Theseus," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122045.
[25] 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.
[26] R. A. Khurma, M. A. Awadallah, and I. Aljarah, "Binary Harris hawks optimisation filter based approach for feature selection," in *Proc. Palestinian Int. Conf. Inf. Commun. Technol. (PICICT)*, 2021, pp. 59–64.
[27] L. Sun, M. Li, and J. Xu, "Binary Harris hawk optimization and its feature selection algorithm," *Chin. Comput Sci*, vol. 50, no. 5, pp. 277–291, 2023.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
[29] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
[30] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, Jan. 2018.
[31] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustain. Cities Soc.*, vol. 72, Sep. 2021, Art. no. 102994.
[32] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[33] X. Tang, S. X.-D. Tan, and H.-B. Chen, "SVM based intrusion detection using nonlinear scaling scheme," in *Proc. 14th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, 2018, pp. 1–4.
[34] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *Int. J. Knowl. Eng. Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.
[35] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[36] S. Yang, X. Zheng, Z. Xu, and X. Wang, "A lightweight approach for network intrusion detection based on self-knowledge distillation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 3000–3005.
[37] W. Ding, M. Abdel-Basset, and R. Mohamed, "DeepAK-IoT: An effective deep learning model for cyberattack detection in IoT networks," *Inf. Sci.*, vol. 634, pp. 157–171, Jul. 2023.
[38] M. Abd Elaziz, M. A. Al-qaness, A. Dahou, R. A. Ibrahim, and A. A. Abd El-Latif, "Intrusion detection approach for cloud and IoT environments using deep learning and capuchin search algorithm," *Adv. Eng. Softw.*, vol. 176, Feb. 2023, Art. no. 103402.
[39] A. G. M. Mengara, Y. Yoo, and V. C. Leung, "IoTSecUT: Uncertainty-based hybrid deep learning approach for superior IoT security amidst evolving Cyber threats," *IEEE Internet Things J.*, vol. 11, no. 16, pp. 27715–27731, Aug. 2024.

Zhendong Wang (Member, IEEE) received the M.Eng. degree from Harbin University of Science and Technology, Harbin, China, in 2009, and the Ph.D. degree in computer applied technology from Harbin Engineering University, Harbin, in 2013.

Since 2014, he has been with the Department of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, China, where he is currently an Associate Professor. His research interests include wireless sensor networks, artificial intelligence, and network security.

Renqiang Zhou is currently pursuing the master's degree with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, China.

His research interests include deep learning, network security, and Internet of Things.

Shuxin Yang received the master's degree in engineering from Southern Institute of Metallurgy, Ganzhou, China, in 2003, and the Ph.D. degree from Tongji University, Shanghai, China, in 2009.

He joined Jiangxi University of Science and Technology, Ganzhou, China, in July 2003. His current research interests include information diffusion modeling, sentiment analysis for text, and intelligent computation for law text.

Dr. Yang is a member of the China Computer Federation (CCF) and an Executive Committee Member of Nanchang Member Activity Center, CCF. He is also a member of the Jiangxi Artificial Intelligence Society.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply

Daojing He (Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science from Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2012.

He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include networks and systems security.

Prof. He is on the editorial board of some international journals, such as IEEE NETWORKS.

Sammy Chan (Senior Member, IEEE) received the B.E. and M.Eng.Sc. degrees in electrical engineering from The University of Melbourne, Melbourne, VIC, Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from Royal Melbourne Institute of Technology, Melbourne, in 1995.

Since December 1994, he has been with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong, where he is currently an Associate Professor.

uthorized licensed use limited to: AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF ENGINEERING. Downloaded on January 30,2026 at 09:51:34 UTC from IEEE Xplore. Restrictions apply