DMSE: An Efficient Malicious Traffic Detection Model Based on Deep Multi-Stacking Ensemble Learning

Saihua Cai[a,b,*], Yang Zhang[a], Yanghang Li[a], Yupeng Wang[a], Jiayao Li[c], Xiang Zhou[a,*]

[a]School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, Jiangsu, China
[b]Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace, Jiangsu University, Zhenjiang, 212013, Jiangsu, China
[c]School of Software, Shanxi Agricultural University, Jinzhong, 030801, Shanxi, China

Abstract

In the context of increasing cyber threats, developing an efficient malicious traffic detection model to recognize the cyber attacks has become an urgent demand in the field of cyber security. This paper proposes an efficient malicious traffic detection model called DMSE based on deep multi-stacking ensemble learning, it is primarily consisted of feature representation module, base model detection module and multi-stacking ensemble learning module. In the feature representation phase, we propose a novel RGB image representation method, which hierarchically represents the global and local features of network traffic by allocating the information to three channels of RGB images. In the base model detection phase, we adopt five different deep learning models—CNN, TCN, LSTM, BiLSTM and BiTCN—as base models for the first-stage prediction. In the multi-stacking ensemble learning phase, we adopt the best-performing BiTCN from extensive experiments as the meta-learner to perform a second prediction using the results from the first stage, thereby obtaining the final detection result. Experiments conducted on USTC-TFC2016, CTU and ISAC datasets demonstrate that the proposed DMSE model significantly outperforms existing ensemble learning-based detection models in terms of accuracy, F1-score, false positive rate (FPR), true positive rate (TPR) and stability. The experimental results indicate that the proposed DMSE model can effectively identify and defend against network attacks, providing the new perspectives and technical support for maintaining a secure network environment.

Keywords: Malicious traffic detection, Deep multi-stacking ensemble learning, Deep learning, Ensemble learning, Meta learner.

1. Introduction

The rapid development of Internet makes network applications become an indispensable part of people's production and life, which also leads to an explosive growth trend in the scale of network traffic generated by accessing various applications [1][2][3]. In recent years, with the increasing complexity of network architecture and the rapid expansion of network applications, network intrusion behaviors such as DoS attacks, network scanning attacks and traffic hijacking attacks have become the main problems that endanger network access. These attacks not only cause service interruptions, data leaks or system damage, but also endanger personal privacy and national security. Therefore, the timely and effective detection of intrusion behaviors has become an important task in the field of cyber security. Compared with normal network access, the intrusion behavior can generate malicious traffic with different features from normal network traffic, leading to the accurately detection of malicious traffic in large-scale network traffic can discover the potential network attack behaviors and effectively maintain the network security [4][5].

To cope with the harm caused by network attacks, scholars have proposed many malicious traffic detection models to prevent potential network attacks and other security threats [6][7]. At the beginning, scholars used port-based methods [8][9] to detect malicious traffic, these methods are simple and easy to implement, but they have the disadvantages such as high false alarm rate and lack of adaptability [10][11]. With the diversification and complexity of network attacks, traditional port-based malicious traffic detection methods are no longer able to cope with new threats, therefore, researchers begun to use machine learning models such as decision tree, Bayesian, SVM, etc. to identify abnormal network behaviors [12][13][14]. Compared with port-based methods, machine learning models have the advantage of automatically learning patterns from network traffic and have strong adaptability, but machine learning models struggle to effectively handle large network traffic with high-dimensional features and prone to overfitting; Furthermore, traditional machine learning heavily relies on manual feature engineering, necessitating domain experts to identify and extract relevant features [15]. Deep learning models can adaptively learn the complex features and achieve better detection results [16][17], thus, deep learning-based method becoming an effective technique for malicious traffic detection [18][19].

At present, deep learning-based malicious traffic detection

* Corresponding author: caisaih@ujs.edu.cn, 1000002653@ujs.edu.cn
Email addresses: caisaih@ujs.edu.cn (Saihua Cai), 3220613047@stmail.ujs.edu.cn (Yang Zhang), 3220613006@stmail.ujs.edu.cn (Yanghang Li), 3220613058@stmail.ujs.edu.cn (Yupeng Wang), lijiayao@sxau.edu.cn (Jiayao Li), 1000002653@ujs.edu.cn (Xiang Zhou)

methods mainly use a single deep learning model for detection [19][20]. In recent years, network environment has shown a complex and ever-changing development trend, resulting in increasingly complex structure of network traffic, which also makes it difficult for a single deep learning model to effectively deal with multiple types of network traffic. In response to the limitations of using a single deep learning model in capturing traffic features, ensemble learning significantly improves the detection performance through using diverse learning strategies and combining the advantages of multiple base models. Currently, various ensemble learning-based malicious traffic detection methods [21][22][23] have been proposed to significantly improve the detection ability of malicious traffic by integrating multiple single detection models through different ensemble strategies. However, some existing ensemble learning-based malicious traffic detection methods use relatively simple voting strategies such as soft voting and hard voting [24][25], which cannot fully utilize the complex relationships between models. Furthermore, many ensemble learning-based malicious traffic detection methods only integrate less deep learning models or integrate the deep learning models with same category [26][27], resulting in only learning highly similar features for the small differences between models, thereby further leading to poor detection performance.

In addition, existing ensemble learning-based malicious traffic detection methods usually convert the original network traffic with the representation of PCAP format into gray-scale images. Although the gray-scale images can represent packet level features such as application layer and network layer, but the convolution kernels cannot fully cover the edges of image when deep learning models perform convolution operations, resulting in rapid loss of edge information in the output image due to size reduction [28][29]. Due to the fact that RGB images can store more features, it is possible to consider converting network traffic into RGB images to solve the problem of edge information loss and improve the training performance of detection model [30]. When representing the features of network traffic using RGB images, due to the insufficient number of features, it is often necessary to fill them with 0X00, which can result in a large amount of information being invalid. Therefore, there is an urgent need for an efficient feature representation method to store more information in RGB images and further improving the accuracy for malicious traffic detection.

In response to the problems in existing methods, this paper proposes a malicious traffic detection model called DMSE based on deep multi-stacking ensemble learning. The main contributions are summarized as follows:

1. This paper proposes an efficient feature representation method for network traffic. Firstly, the original PCAP files of network traffic are split by session using the SplitCap tool to obtain sub PCAP files for different sessions. And then, the script provided by USTK-2016 is used to prune and fill the sub PCAP files according to the required size of image, where the sub PCAP files smaller than image size are filled with 0X00, while the ones larger than image size are trimmed to make them in regular size. Finally, all layer information of PCAP files is placed in the R and B channels and the application layer information is placed in the G channel of RGB images, thus efficiently representing the features of network traffic with RGB images.

2. Through comparing the detection performance as well as the advantages and disadvantages of multiple deep learning models such as CNN, RNN, LSTM, TCN, ResNet and EfficientNet, this paper integrates five base models (including CNN, LSTM, BiLSTM, TCN and BiTCN) to utilize the advantages of each base model in handling different types of network traffic, thereby improving the accuracy of malicious traffic detection.

3. This paper introduces a multi-stacking ensemble strategy to increase the variety and quantity of base learners, thereby improving the flexibility and diversity of the model, as well as reducing the risk of overfitting and achieving better generalization ability and robustness.

4. Due to the fact that BiTCN model not only utilizes the bidirectional convolution to simultaneously consider past and future information to capture the temporal features of network traffic, but also has the convolutional layers to quickly extract local features, this paper uses BiTCN as a meta learner to improve the complexity and diversity of whole detection model and reduce the overfitting problem caused by small differences of model.

5. Extensive experiments are conducted on three widely used network traffic datasets to compare the proposed DMSE model with five advanced ensemble learning-based malicious traffic detection models, and the experimental results show that DMSE can more accurately detect the malicious traffic and has better stability.

The organization of the remaining parts is as follows: Section 2 reviews the related work of existing malicious traffic detection methods and briefly compares the ensemble learning-based models with the proposed DMSE model. Section 3 describes the details of DMSE model, including the framework of DMSE, the feature representation of network traffic, the introduction of base models and multi-stacking ensemble learning strategy. Section 4 conducts extensive experiments to validate the effectiveness of DMSE model. Section 5 summarizes the entire paper and provides the future research directions.

2. Related Work

This section first reviews the port-based malicious traffic detection methods, machine learning-based malicious traffic detection methods, deep learning-based malicious traffic detection methods and ensemble learning-based malicious traffic detection methods. And then, the proposed DMSE method is briefly compared with existing ensemble learning-based malicious traffic detection methods.

2.1. Port-based malicious traffic detection

Port-based detection methods identify the malicious activities such as data theft, malware propagation and denial of service attacks by monitoring whether network access is transmitting malicious traffic using specific network port numbers.

Vugrin et al. [8] proposed a mathematical model using port scanning and detection techniques to describe the attacker port scanning and defender intrusion detection, it first defines static input parameters such as network layout, node characteristics, packet loss rate and IP range, and then sets two different attacker strategies to parameterize the Nmap tool, thereby simulating slow covert and fast overt attack behaviors. This method has significant effectiveness in evaluating port attacks and defense, but the parameter settings and experimental conditions of this model are relatively ideal, which makes it cannot adapt to complex network environment.

Blaise et al. [9] proposed a technique for early detection of new types of botnets and newly exploited vulnerabilities, it first disperses the detection process to different network segments and only preserves distributed anomalies, and then performs the port-level monitoring and uses a change detection algorithm to detect malicious attack behaviors. The experimental results show that this technology can effectively detect abnormal behaviors such as emerging port usage and known network attack patterns, but this technology has a relatively fixed threshold selection for anomaly detection and does not consider adaptive adjustment of thresholds in different network environment.

Wu et al. [10] proposed a packet sampling-based method for detecting covert port scans, it first uses the systematic sampling technology to measure network traffic in the high-speed network, and then selects the accumulated features of packets and uses a random forest to identify malicious traffic. However, the information loss problem makes it difficult to detect extremely short scan streams.

Ying [11] proposed a TCP port scan detection framework based on fuzzy logic controller with the use of fuzzy rule library and Mamdani inference, it enables the network administrators and network security experts to track network behaviors in real time, but it does not fully explore the adaptability of fuzzy rules in complex attack scenarios and only focuses on port scanning detection without in-depth research on the processing capabilities of different types of network attacks.

In summary, port-based detection methods can provide real-time detection capabilities and easy to deploy, but these methods perform poorly and lack adaptive capabilities when dealing with complex and dynamic network environment; In addition, they can only detect known malicious traffic and lack effective verification of real-time and fluctuating network traffic.

2.2. Machine learning-based malicious traffic detection

Machine learning-based detection is a method that utilizes machine learning models to identify potential malicious activities. Compared with port-based detection methods, it is a data-driven model that can automatically learn and adapt to new attack patterns, thereby improving the responsiveness and accuracy of network security systems.

Guendouz and Amine [12] proposed a feature selection method based on the binary dragonfly algorithm for Android malware detection, it first extracts the network traffic and software permissions as feature vectors, and then selects the feature vectors and uses machine learning models such as decision tree, random forest, SVM and Naïve Bayes for classification. The experimental results show that this method can improve Android malware detection and reduce model training time, but its adaptability will be reduced in different types of malware or special Android environment due to the use of small datasets.

Chen et al. [13] proposed a hybrid network intrusion detection system to improve detection performance. This system first uses a BERT model to convert the text information into numerical data, and uses the feature flattening technology to map two-dimensional host features into one-dimensional vectors; Next, the flattened host features are combined with flow features to obtain different mixed feature representations; Finally, a two-stage collaborative classifier is constructed using XGBoost model to detect malicious traffic. The experimental results show that the mixed features of network and host can significantly improve the detection performance, but it cannot comprehensively analyze all attack samples due to the exclusion of four minority attack types in the original dataset.

Talukder et al. [14] proposed a machine learning-based detection model, it first reduces the size of dataset by merging the similar classes with low instances, and then uses the random oversampling technique to add random samples to the minority classes to solve the imbalance of dataset, and the principal component analysis is used to map the high-dimensional features to low dimensional space to accelerate the detection speed. However, the effectiveness of this method has only been validated on small-scale network traffic datasets, which causes it may lack practical deployment universality.

Yan et al. [31] proposed a machine learning-based network intrusion detection system in black box scenarios, it adopts a practical automatic black box evasion attack strategy and implements seven common machine learning models as malicious traffic detection models to cope with different network scenarios. The experimental results show that it can maintain the detection performance for various types of malicious traffic, but the efficiency of attack detection will be reduced due to lack considering the increase of feature dimension caused by the passage of payload over time.

Han et al. [32] proposed a novel method based on cross entropy and SVM, it first subtracts the non-representative features from network traffic and extends the commonly used 5-tuple representation to a 7-tuple feature vector; And then, the probability distribution and cross entropy of 7-tuple are calculated within a defined statistical window to generate the 7-tuple cross entropy feature vector; Finally, the multi-class SVM classifier is trained by importing the 7-tuple cross entropy feature vector. The experimental results indicate that the proposed method can achieve a higher detection, but the use of SVM model leads to a significant increase in time consumption.

Gowda et al. [33] proposed an anti-phishing technique based on random forest and extraction rule framework, and constructed a novel browser architecture. Specifically, the extrac-

tion rule framework is first used to extract 30 features of the website; And then, the random forest is used as detection model and the GridSearchCV is used to find the optimal parameters; Finally, the model is trained through K-fold validation. A large number of experiments prove that the proposed detection technology has the advantages of high accuracy and fast speed; However, the experiments are only run on small-scale network traffic datasets, which poses a problem of poor recognition performance for new types of malicious traffic.

Almorabea et al. [34] validated the detection performance of LR, KNN and SVM for ping flood attacks in IoT network, and the results show that the KNN model can achieve good detection performance and also has advantages in parameter settings. However, if attackers use deceptive IP address to hide themselves, it is difficult for the detection method to distinguish between legitimate users and attackers.

In summary, compared to port-based malicious traffic detection methods, machine learning-based detection methods can autonomously learn the features of network traffic and achieve better results in feature adaptability and detection accuracy. However, the accuracy is highly dependent on the quality and diversity of training data, and the machine learning-based methods are prone to the problems such as overfitting and high computational consumption.

2.3. Deep learning-based malicious traffic detection

Compared with machine learning models, deep learning models can automatically extract the features of network traffic and have strong recognition ability for complex attacks, which can adapt to complex and changing network environment.

Islam et al. [16] proposed an effective deep learning-based model to detect malicious network traffic. Specifically, this method first uses the CIC-Flowmeter to convert the captured trajectory of original network traffic into bidirectional structure and normalizes it to obtain a heterogeneous traffic dataset; And then a 1D-CNN is constructed with input data in a hierarchical sequential format rather than 2D image shapes to make predictions on the constructed dataset. The experimental results show that the proposed model has higher accuracy in classifying heterogeneous malicious traffic, but its performance in detecting network traffic with temporal features is unstable.

Devendiran and Turukmane [17] proposed a deep learning-based network intrusion detection method using chaos optimization strategy, this method preprocesses the network traffic using normalization and balances the categories using an extended synthetic sampling method; And then, a gated attention dual-based LSTM model is used for detecting intrusion behaviors. The experimental results show that the proposed method has better accuracy and robustness than existing methods, but it lacks certain generalization.

Keshk et al. [18] proposed an IoT network intrusion detection method based on LSTM, it first extracts the important featuresthrough the feature importance ranking strategies, and then trains the extracted features using LSTM to obtain the final detection results. However, the dynamic network environment causes the importance of features to change constantly, which can affect the effectiveness of the detection method.

Nandanwar and Katarya [19] proposed a deep learning-based botnet attack detection method, it first augments the dataset to increase the size of attack samples, and then constructs an adaptive CNN-GRU model for detecting botnet attacks. This method can achieve efficient and stable detection results in multiple scenarios, but its performance may vary with the evolution of attack strategies, making it difficult to adapt to constantly changing network environment.

Clinton et al. [20] proposed a method for identifying DDoS attacks using deep learning model, it first divides the captured original network traffic into small windows and normalizes the values in the window array to 0-255; And then, the OpenCV2 is used to generate RGB images; Finally, a 32-layer complex CNN model is constructed for classification. The proposed model exhibits comparable or better performance than other models in detecting DDoS attacks, but the high number of hidden layers in the model leads to high computational cost.

Zhao et al. [35] proposed an improved model based on TCN and attention mechanism to improve the detection ability of network attacks. This model first maps the relevant text features to numerical representations, as well as removes the missing values and performs hot encoding on discrete features; Next, the spatiotemporal attributes are extracted in parallel and the attention mechanism is used to assign different weight to different attack features; Finally, the TCN model is used for detecting attacks. Experimental results on KDD CUP99 and UNSW-NB15 datasets show that the proposed model has better detection performance, but its detection efficiency for novel attacks is poor due to overfitting issues.

Wang et al. [36] proposed an attention-based BiLSTM model for detecting collaborative network attacks, it first models the HTTP traffic as a series of natural language sequences, and then establishes the BiLSTM model under a multi-domain machine learning framework to detect the processed network traffic. The experimental results show that the proposed method can identify abnormal network attacks in different domains and effectively solve the problem of heterogeneous distribution of multi domain data, but its generalization ability is weak.

In summary, deep learning-based malicious traffic detection models have strong feature learning capabilities and can automatically learn the features through multi-layer structures, reducing the need for manual feature engineering compared to machine learning. In addition, when dealing with high-dimensional network traffic, deep neural networks can capture the complex spatiotemporal feature relationships, which can effectively improve the accuracy of malicious traffic detection. However, the single deep learning model-based detection methods cannot adapt to different types of network attacks and have strong limitations in capturing the features of network traffic.

2.4. Ensemble learning-based malicious traffic detection

Compared with using a single machine learning model or deep learning model, ensemble learning strategy can improve the detection accuracy and robustness of the model by combining the prediction results of multiple learning models.

Ren et al. [22] proposed an ensemble network intrusion detection method, it divides the network traffic samples into

Table 1
The comparison of different ensemble learning-based malicious traffic detection methods

Models	Feature Representation	Base Detection model
DUEN [22]	CSV	CART, RF, GBDT
Lin et al. [23]	PCAP	Random Forest, Light Gradient Boosting Machine
Crespo-Mart´ınez et al. [24]	PCAP	KNN, LG, RF, SGD, SVM
RepuTE [25]	CSV	KNN, Extra Tree Classifier, Quadratic Discriminant Analysis Classifier
Alghamdi and Bellaiche [26]	PCAP	LSTM, CNN, ANN
ENIDS [37]	CSV	CNN, LSTM, GRU, DNN
DIS-IoT [38]	CSV	MLP, DNN, CNN, LSTM, FCNN
Alsaffar et al. [39]	CSV	MLPRF, CatBoost, XGBoost
DMSE (proposed)	RGB images	CNN, LSTM, BiLSTM, TCN, BiTCN

three types of easy, boundary and noise through dynamic undersampling techniques, and then uses the Boosting as basic ensemble framework to ensemble the base models of CART, RF and GBDT. The experimental results show that the proposed method has strong generalization and robustness to noisy samples, but its detection performance may fluctuate in real network environment.

Lin et al. [23] proposed a hypergraph-based ensemble network intrusion detection method, it first uses an adversarial sample generation module to generate the adversarial samples to simulate unknown adversarial attacks, and then the hypergraph is used to model port scanning activity and generate a hypergraph-based feature set; Finally, a random forest and a lightweight gradient boosting machine are used to construct an ensemble model to detect intrusion activities. The experimental results show that the proposed method performs well in terms of accuracy, recall, and F1-score, but it would consume high computational resource for its relatively complex structure.

Crespo-Martínez et al. [24] proposed an ensemble learning-based malicious traffic detection method based on machine learning models to deal with SQL injection attacks, it selects the KNN, LR, SVM, SGD and RF as base models and adopts the majority voting strategy to detect malicious traffic. The experimental results demonstrate that this method can accurately detect SQL injection attacks, but its detection performance may be affected by network fluctuations.

Verma and Chandra [25] proposed an ensemble learning-based method for classifying DoS/DDoS and Sybil attacks in the Fog-IoT field, it first uses a mixed feature selection technique to calculate the feature importance scores and mutual information gains, and then integrates the base models including KNN, extra tree classifier and quadratic discriminant analysis classifier using a soft voting ensemble strategy. Experiments show that the proposed model has better detection performance in terms of accuracy, precision, recall and F1-score, but it is difficult to perform real-time calculations during deployment.

Alghamdi and Bellaiche [26] proposed an ensemble deep learning-based network intrusion detection model, it first performs the data preprocessing and utilizes the Fit_Transform tool in the Scikit_Learn library to transform the features of network traffic into inputs suitable for deep learning; And then, the ANN, CNN and LSTM are used as base models and the majority voting strategy is adopted to detect network intrusions. Experiments show this ensemble strategy can improve the detection accuracy, but it has the problem of insufficient generalization ability in practical deployment.

Sayem et al. [37] proposed a deep learning-based ensemble framework for network intrusion detection, it uses the SMOTE technology to sample the network traffic and inputs the obtained data into the base models of CNN, LSTM and GRU in the first layer of stacking for training; In the second layer, the DNN model is used as a meta learner to connect the results of three base models as input to generate the final prediction result. Experimental results show that this method has good performance in intrusion detection, but it cannot detect new attacks due to the limitations of used feature extraction method.

Lazzarini et al. [38] proposed a new intrusion detection method for IoT based on stacking ensemble learning, it constructs four deep learning models (including MLP, DNN, CNN, LSTM) as base models and builds a fully CNN as meta learner to integrate the predictions of base learners. Experimental results indicate that the stacking ensemble strategy improves the model's ability to analyze highly diverse data, but its generalization capability is unknown.

Alsaffar et al. [39] proposed an intrusion detection method based on hybrid feature selection and stacking ensemble learning, it first performs the feature selection and inputs the extracted features into stacking ensemble model with a MLP as meta learner and random forest, CatBoost and XGBoost as base models for training. The experimental results show that this method has a certain competitive advantage compared to state-of-the-art methods, but it is difficult to determine whether network traffic is the cause of intrusion traffic due to the lack of interpretability.

In summary, using an ensemble learning-based malicious traffic detection model combined with the advantages of multiple models can not only better handle the feature distribution of complex network traffic, but also improve the accuracy and stability of malicious traffic detection; In addition, through comprehensively utilizing the sensitivity of different models to various features, the stronger generalization ability can be achieved and the risk of overfitting can be reduced.

Table 1 compares different ensemble learning-based malicious traffic detection models with the proposed DMSE model. Compared with these ensemble learning-based detection models, DMSE can capture and analyze the temporal features and convolutions of network traffic to obtain local features, and can improve the generalization ability, robustness and the ability to handle complex features of the detection model through the multi-overlap addition of stacking ensemble model, thereby

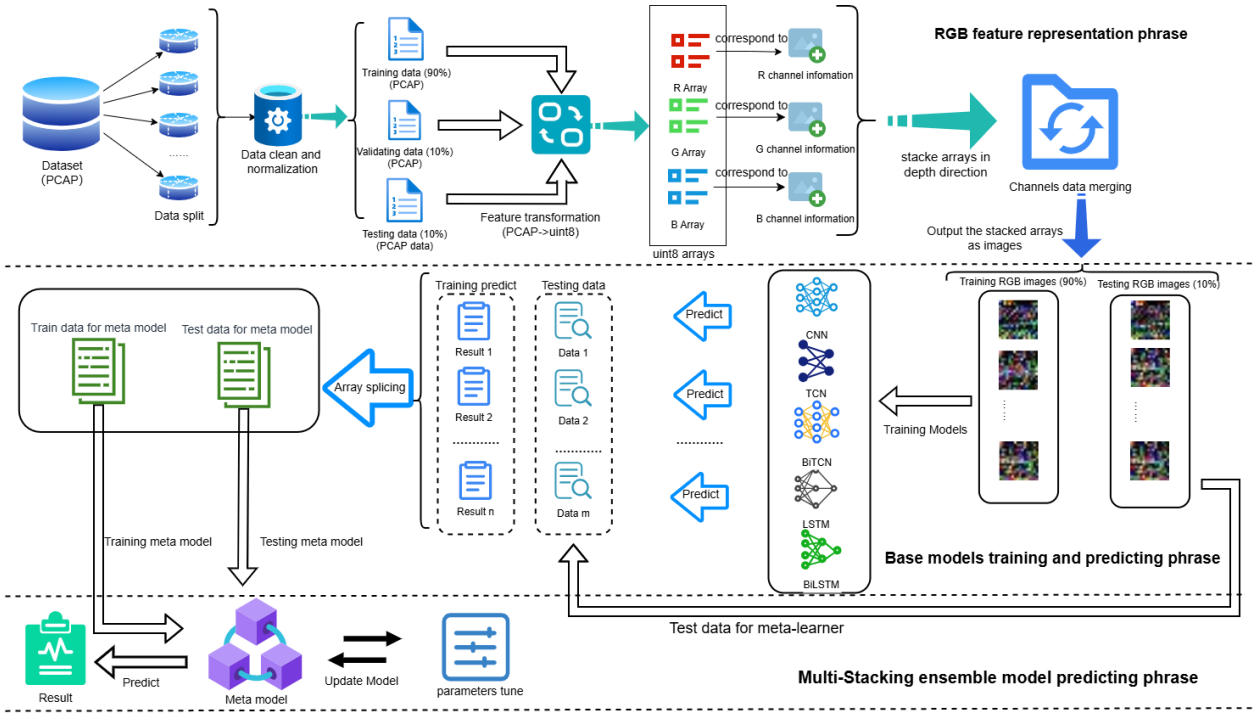


Fig. 1. The framework of DMSE model

more accurately handling different types of malicious traffic.

3. Methodology

This section first detailed describes the overall framework of proposed malicious traffic detection model called DMSE based on deep multi-stacking ensemble learning, and then further describe the feature representation method, the selection and training of base model, and the multi-stacking ensemble strategy.

3.1. The framework of DMSE model

The overall framework of DMSE model is shown in Fig. 1, it mainly includes the feature representation module, base model detection module and multi-stacking ensemble learning module. In the DMSE model, the original network traffic in the format of PCAP files are first segmented and converted into RGB images for feature representation. And then, the obtained RGB images are input into five deep learning base models for training. Finally, the detection results of base models are concatenated and used as the training set for meta learner for multi-stacking ensemble learning, thereby more accurately detecting the malicious traffic in large-scale network traffic.

(1) Feature representation module: This module mainly represents the original network traffic in the format of PCAP files to RGB images through traffic segmentation, data cleaning and feature transformation. In order to make the deep learning model can effectively handle the network traffic, network traffic in the format of PCAP needs to be processed. Traditional methods directly convert it into CSV format or gray-scale images, but these two formats are difficult to distinguish and express complex features of network traffic and suffer from information loss. Compared to these two representations, each pixel in RGB images can simultaneously carry information from three dimensions of R, G and B, this not only accommodates more features but also has a higher spatial utilization rate; In addition, RGB images can be more flexibly mapped for features, and different feature mapping relationships to R, G and B channels can be customized according to requirements. The advantages of RGB images can better display the features of network traffic, thereby helping the detection modelmore accurately detect malicious traffic.

(2) Base model detection module: This module mainly uses the selected deep learning base models to train based on the converted RGB images, thereby generating the training sets and testing sets for multi-stacking ensemble learning module. The selected base models include CNN, LSTM, BiLSTM, TCN and BiTCN; Among them, CNN can extract the local features of network traffic, LSTM and BiLSTM can handle the temporal dependencies in network traffic sequences, while TCN and BiTCN can efficiently capture the long-range dependencies in time series. Through integrating these models, the overall detection performance and robustness can be improved to make them adapt to various complex network environment. In addition, to solve the problem of slow convergence caused by the use of negative logarithmic loss function in these base models, it is modified to FocalLoss function; To solve the problem of neuron "death" caused by the use of ReLU activation function in the base model, it is modified to ELU activation function; These improvements further improve the detection performance of base models.

(3) Multi-stacking ensemble learning module: This mod-

ule mainly integrates the above base models to fully utilize their advantages and improve the detection accuracy. Compared with traditional ensemble strategies such as bagging, boosting and voting, multi-stacking captures different features and patterns in the network traffic by combining different types of base models, which can smooth the prediction, reduce the possible over fitting phenomenon with the use of a single model, thereby improving the generalization ability on unknown network traffic, therefore, it is selected as the ensemble strategy in DMSE model. In the selection process of meta learner in the second layer, compared with other deep learning base models, BiTCN is a powerful model due to its bidirectional processing ability, long-range dependency capture ability, efficient feature extraction ability and robustness, therefore, it is used as a meta learner in DMSE model.

3.2. Feature representation of network traffic

Network traffic is generally stored in the form of PCAP packets [23][24]. To enable the deep learning models to better handle network traffic, it is necessary to convert the PCAP files into CSV or gray-scale images. However, the above two representation manners cannot well describe the features of network traffic [28][29]. Among them, the representation of CSV has the disadvantages of large sizes and a lot of invalid information, while the gray-scale images are difficult to distinguish and express complex features and have the problem of losing feature information. In contrast, RGB images represent the network traffic through R, G and B channels, which can store more complex information than gray-scale images and CSV, and have more flexible expression methods. Therefore, the represention of network traffic using RGB images helps malicious traffic detection models learn more features and obtain better detection results.

The traditional represention method with RGB images uses the traffic segmentation and data cleaning techniques to convert the network traffic into RGB images [40]. Specifically, the PCAP packet is first segmented and cleaned into blocks with the size of 768 byte (if the length exceeds 768 bytes, the first 768 bytes are truncated, otherwise 0X00 is added at the end). Next, the PCAP file is opened to read its binary content and convert it into a hexadecimal string. The hexadecimal string is then segmented into bytes (every two characters are one byte), and the one-dimensional integer array is reshaped into a three-dimensional array with the shape of (height, width, 3) to correspond to the height, width and color channels of RGB image. Finally, the reshaped array is mapped to [0, 255] and the PIL library is used to save the RGB images. However, there is a greater feature demand due to the three channels of RGB images, resulting in the need for more 0X00 to fill, while the 0X00 appears as large black areas in RGB images, which has no effect or even has a negative impact on the feature representation, thereby resulting in insufficient feature representation of network traffic.

In order to address the negative impact caused by filling 0X00, this paper proposes a novel and effective feature representation method for network traffic in RGB images. For the selection of information stored in the R, G and B channels, considering that all layers of network traffic contain the complete information and its scale is relatively large, it is placed in the R and G channels; In addition, considering that the application layer is the most direct source of network traffic, the interaction behavior of users ultimately generates data through the application layer, therefore, the application layer information is separated separately and placed in the B channel. By storing the information from all layers and application layers to generate RGB images, deep learning models can learn the complex and comprehensive information, thereby improving the detection accuracy of malicious traffic.

Specifically, the SplitCap tool is used to split the PCAP packets of original network traffic into sessions to obtain session records represented by each small file, where each session record contains the information from all extracted layers (that is, the information of all layers is placed in *AllLayer* folder and L_4 folder respectively, and the information of application layer is placed in L_7 folder), thereby placing them in their respective folders. And then, the obtained small PACP files are normalized to organize them into a size suitable for the input of deep learning model. Considering that the segmented file contains less information, a 16*16 sized RGB image is used for feature representation, which means that each normalized RGB image is in the size of 16*16* 3 bytes. Finally, some improvements are made to the RGB image generation script to enable the model to generate improved RGB images, the specific process is as follows: Firstly, three target directories are created to store the file names of all layer and application layer, where R and G channels are used to store the file names of all layer and B channel is used to store the file names of application layer. And then, according to the data in the directory list, three file contents are read each time to converted them to hexadecimal numbers for storage, followed by converting them into a uint8 array to map their values to [0, 255] interval as shown in formula (1), which are used to generate RGB images by mapping the file data to RGB color values, where X represents the tens of hexadecimal digits and Y represents the units of hexadecimal digits. Finally, the np.dstack is used to stack the matrices of three channels along the third dimension to merge the uint8 data of R, G and B channels (r_matrix, g_matrix and b_matrix) into a three-dimensional array, as shown in formula (2). And then, the Image.fromarray method of PIL library is used to convert the NumPy array into an image object and save it to the specified directory.

$$Value_{unit8} = X * 16^1 + Y * 16^0 \tag{1}$$

$$RGB\_array = np.dstack(r\_matrix, g\_matrix, b\_matrix) \tag{2}$$

Through converting the PCAP files of different levels into RGB images, a multi-dimensional traffic view can be created to make the analysis of network traffic more intuitive and efficient. Compared with gray-scale images, the representation of RGB images can enable quickly capture the overall features and potential anomalies of network traffic through color changes and combinations, thereby providing support for subsequent deep analysis and decision-making.

The proposed feature representation of network traffic in RGB images is shown in Algorithm 1. Firstly, the SplitCap tool is used to split the PCAP packets P of original network traffic into sessions, and the data cleaning and normalization are performed on the sub-files, thereby storing the results in the *AllLayer*, L_4 and L_7 folders respectively (see lines 01-02). And then, the number of files in these three folders is calculated and the minimum value is taken to ensure that no array out-of-bounds error occurs in the subsequent processing (see lines 03-04). Next, the required file size *required_size* is calculated and the *uint8_data* is initialized to store the 8-bit unsigned arrays (see lines 06-07). In the following loop, for each session, the corresponding files are selected from these three folders to read their binary contents, and they are converted to hexadecimal (see lines 08-10). If the file content is not long enough to generate an RGB image, it is padded with 0x00 (see lines 11-13). The processed hexadecimal files are converted to uint8 format to store them (see line 14). And then, the converted data is used for three RGB channels of *R_channel*, *G_channel* and *B_channel* (see lines 16-18). After that, stacking them along the depth to form a three-dimensional array *RGB_array*, thereby converting it to RGB images (see lines 19-21). Finally, the list R of all generated RGB images is return (see lines 23-24). Through the above processing, each RGB image represents a network session, which facilitates further analysis and research through visualization.

Algorithm 1 Feature representation

Input: PCAP files P
Output: RGB images R

1: Split P by sessions using SplitCap
2: Clean and normalize P and store sub-files in *AllLayers*, L_4 and L_7
3: $files\_count$ = min(len(*AllLayers*), len(L_4), len(L_7)) // Avoid boundary crossing errors
4: **for** i in $files\_count$ **do**
5: *required_size* = 16 * 16 * 3 // Calculate required size for RGB
6: $files$ = [*AllLayers*[i], L_4[i], L_7[i]]
7: *uint8_data* = [] // Store uint8 arrays
8: **for** each $file$ in $files$ **do**
9: open $file$ in binary mode as f
10: $file\_hex$ = convert f to hex
11: **if** len($file\_hex$) <*required_size* **then**
12: pad$file_hex$ with 0x00
13: **end if**
14: *uint8_data*.append(convert $file\_hex$ to uint8 array)
15: **end for**
16: *R_channel* = *uint8_data*[0]
17: *G_channel* = *uint8_data*[1]
18: *B_channel* = *uint8_data*[2]
19: *RGB_array*= np.dstack(*R_channel*, *G_channel*, *B_channel*) //Stack R_channel, G_channel and B_channel along the depth axis
20: *RGB* = image_convert(*RGB_array*) // represent *RGB_array* as RGB image
21: *R_image*= image_save(*RGB*) // save RGB
22: **end for**
23: *R* = *R_image*
24: **return** *R*

3.3. Base model selection and training

Compared with using a single learner to train the malicious traffic detection model, ensemble learning-based malicious traffic detection model can better utilize the advantages of multiple base learners. A large number of literatures [24][25][27] have also proved that the use of ensemble learning strategy can achieve better detection results. In ensemble learning-based detection model, the choice of base model has a significant impact on the detection performance. Therefore, after comparing the characteristics of various deep learning models, the following base models are selected as base models. Among them, the Convolutional Neural Network (CNN) model can automatically extract the features from RGB images, and its hierarchical structure also enables it to capture the information at different scales and abstract levels, thereby better identifying potential malicious traffic. The Long Short Term Memory (LSTM) model can use the memory units to retain long-term information and filter short-term noise, thus more accurately capturing the characteristics of attacks when dealing with complex network scenarios. The Bidirectional Long Short Term Memory (BiLSTM) model considers both the past and future information of input sequence, enabling the model to comprehensively understand the dynamic features of malicious traffic and more accurately capture the subtle changes in traffic sequence to improve the detection accuracy and robustness of the model. The Temporal Convolutional Network (TCN) model utilizes the convolution operations to capture the temporal dependencies and can simultaneously process the entire sequence, which not only improves the computational efficiency but also reduces the training time; In addition, its structure allows the model to flexibly adjust the receptive field to adapt to the scale changes of multiple input features, thereby improving its robustness in complex network environment. Bidirectional Temporal Convolutional Network (BiTCN) can simultaneously capture the past and future contextual information of input network traffic, which can comprehensively understand the dynamic features of malicious traffic, this bidirectional processing manner makes the model more accurate and reliable in identifying complex attack patterns. In the following, we will introduce the five selected base models separately.

3.3.1. CNN

In the detection process, if the graphics are located in the different positions of images, it is likely to reduce the detection accuracy. CNN uses a convolution operation that convolves a movable small window with the image to improve the detection efficiency, this small window is a set of fixed weights that can be regarded as a specific filter or convolution kernel. The convolution operation takes a portion of the values in image matrix from left to right and top to bottom with the same size as the filter, and then multiplies them with the values in the filter and sums them up to form a matrix. Compared with other deep learning models, CNN can extract the feature representation at different levels through convolutional layers, and the convolution operation with parameter sharing features also makes it more efficient in processing high-dimensional network traffic. Convolutional

kernels often fail to align their centers with the edges of input image, resulting in a decrease in the size of output images. Therefore, before performing the specific operations, the data padding operation (i.e., adding a certain amount of data around the edges of input content) is introduced to control the output size of convolutional layer, and then calculates the output size of any given convolutional layer using formula (3).

$$O = [\frac{n + 2p - f}{s}] + 1 \tag{3}$$

In formula (3), O represents the size of output RGB image, p represents the size of padding, $n * n$ represents the size of input RGB image, $f * f$ is the size of convolution kernel and s represents the size of stride.

Usually, the non-linear activation functions are applied after convolution operations to help the network learn non-linear features, thereby improving the expressive power of features. CNN achieves more accurate classification and prediction by introducing nonlinearity to better approximate the complex functions. However, traditional CNN models use ReLU activation function for feature mapping. When x<0, the neuron gradient of ReLU is set to 0, resulting in the problem of neuron "death", this forced sparse processing reduces the effective features learned by the model and lowers the detection accuracy of model. Therefore, we intend to replace the ReLU activation function with ELU as shown in formula (4) to ensure its smoothness on the negative half axis and alleviate the gradient vanishing problem.

$$ELU(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{4}$$

In the field of malicious traffic detection, CNN demonstrates its powerful feature extraction capability and sensitivity to local features of network traffic, it automatically learns and recognizes the complex patterns and features through multi-layer convolution and pooling operations. In addition, CNN can quickly adapt to different types of network traffic, and its translational invariance allows it to effectively respond to small changes in network traffic, thereby improving its ability to detect new types of malicious attacks. These advantages of CNN make it being a base model for malicious traffic detection.

3.3.2. LSTM

CNN has become an efficient base model due to its powerful feature extraction ability, but it cannot effectively extract the long-term dependencies of features in network traffic. Compared with CNN, RNN solves the problem of traditional neural networks having a fully connected structure from the input layer to hidden layer and then to output layer to make it have no connection state between nodes in each layer. RNN remembers the previous information of network traffic and apply it to current output calculation, that is, the nodes between the hidden layers are no longer unconnected but connected, and the input of hidden layer includes not only the output of input layer, but also the output of previous hidden layer. The calculation of its output is shown in formula (5).

$$O_t = g(V \cdot S_t), S_t = f(U \cdot X_t + W \cdot S_{t-1}) \tag{5}$$

In formula (5), O_t represents the output at time t, S_t represents the value of hidden layer at time t, U represents the parameter matrix from the input layer to hidden layer, O represents the vector from the output layer, v represents the parameter matrix from the hidden layer to output layer, X_t represents the vector value at time t and W represents the weight parameter vector.

However, RNN suffers from the serious problems such as gradient vanishing, gradient explosion and long-term dependencies of features, and its computational efficiency is also relatively low. As a variant of RNN model, LSTM can effectively capture the temporal dependencies in RGB images through gating devices and unit states. It learns the dynamic changes and contextual information between different frames by inputting each RGB image, thereby demonstrating significant advantages in processing network traffic containing temporal variations.

On the basis of RNN, LSTM adds a new cell state and a gating device including an input gate to preserve the long-term state. The information from the input layer at each moment will first pass through input gate, and the switch of input gate will determine whether the information at this moment is input into the unit state and stored for long-term memory. Whether the information in unit state is output at each moment depends on the forget gate, which judges whether the information in unit state is forgotten at each moment. If it passes through this gate, the information will be forgotten. The specific process for the three gates of LSTM is shown in formula (6).

$$\begin{aligned} I_t &= \sigma(X_t W_{xi} + H_{t-l} W_{hi} + b_i) \\ F_t &= \sigma\left(X_t W_{xf} + H_{t-l} W_{hf} + b_f\right) \\ O_t &= \sigma(X_t W_{xo} + H_{t-l} W_{ho} + b_o) \end{aligned} \tag{6}$$

In formula (6), σ is the Sigmoid activation function; I_t, F_t and O_t are the input gate, forget gate and output gate, respectively; X_t is the input vector containing current time feature information, it is usually an observation value in time series; W is the weight matrix, b is the bias term of current gate control device, and H_{t-1} is the hidden state of previous time, the specific calculation is shown in formula (7).

$$H_t = O_t \odot tanh(C_t) \tag{7}$$

In formula (7), C_t is the memory state, and the specific calculation is shown in formula (8).

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{8}$$

In formula (8), $\tilde{C}_t$ is a candidate memory state, it is calculated using formula (9), where tanh represents the tanh activation function.

$$\tilde{C}_t = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{9}$$

Network traffic often has the temporal and dependency relationships, LSTM can effectively capture the long-term dependency through its cyclic structure, thereby identifying the potential attack patterns. In addition, LSTM also can handle the irregular and variable length input sequences, making it perform

better in analyzing dynamic network traffic. Through deeply learning the temporal features of RGB images, LSTM can not only improve the detection accuracy but also reduce false alarm rate. Therefore, LSTM is selected as a base model.

3.3.3. BiLSTM

Although LSTM can effectively capture the long-term dependencies in network traffic, it can only handle the unidirectional information from past to future in a sequence and cannot fully capture the information that may be important for current decisions in the future time steps, which limits its comprehensive understanding of specific contexts in network traffic; In addition, when modeling the complex temporal dependencies (especially when there are important correlations between certain parts of the sequence and subsequent data), the one-way processing capability of LSTM limits its ability to capture the long-term dependencies, resulting in a decrease detection performance. The network traffic represented by RGB images usually contains the rich spatial and temporal features, therefore, BiLSTM model is introduced to simultaneously process network traffic in both forward and reverse directions to effectively capture the contextual relationships between RGB images. This bidirectional processing capability enables BiLSTM to have a more comprehensive understanding of dynamic changes, thereby further improving the recognition of temporal features of network traffic. Specifically, BiLSTM first uses a forward LSTM model to obtain the input vectors [hl_1, hl_2, hl_3], which is similar to LSTM model; Compared with forward LSTM model, the backward LSTM model only reverses the processing order to obtain the input vectors [hr_1, hr_2, hr_3]; Next, formula (10) is used to concatenate the forward and backward output vectors, and map the merged output to the target space through a fully connected layer as shown in formula (11).

$$h_t^{bi} = [h_t^f, h_t^b] \tag{10}$$

$$y_t = W_y \cdot h_t^{bi} + b_y \tag{11}$$

In formulas (10) and (11), h_t^f is the output result of forward LSTM, h_t^b is the output result of backward LSTM, y_t is the probability distribution of the output at time step t, W_y is the weight, and b_y is the bias term of output layer, which is used to adjust the output value to improve the fitting ability of model.

In response to the rich spatiotemporal features of network traffic, BiLSTM can capture both the past and future information through its bidirectional structure, thus gaining a more comprehensive understanding of the behavior of network traffic; This bidirectional processing can effectively identify the complex patterns and features that may be overlooked in traditional unidirectional LSTM. In addition, BiLSTM can preserve the long-term dependencies and effectively identify the malicious behavior features accumulated over a longer period. Finally, BiLSTM can effectively filter out the network traffic caused by feature information loss due to network jitter, packet loss and other reasons through gating mechanisms, thereby improving the detection accuracy and reliability. Therefore, BiLSTM model is also being selected as a base model.

3.3.4. TCN

The complex and ever-changing network environment makes the temporal features of network traffic very prominent, while TCN is a model with temporal characteristics, it has higher stability and solving speed than traditional RNN and LSTM, as well as has the following significant differences: (1) The convolution in its architecture is causal, which means that there is no information "leakage" from the future to past; (2) Its architecture can take the sequences of any length and map them to the output sequences with same length; (3) It emphasizes how to use a combination of deep networks and dilated convolutions to construct long effective historical sizes. Assuming that the input sequence of network traffic is x_0, ..., x_T and it hopes to predict the corresponding output y_0, ..., y_T, then, only the previously observed input x_0, ..., x_t can be used to predict the output y_t. Formally, the sequence modeling network is any function f that generates mappings: $X_{T1} \rightarrow Y_{T1}$, as shown in formula (12).

$$y_0, ..., y_T = f(x_0, ..., x_T) \tag{12}$$

TCN uses the convolution operation to process sequential data, which not only avoids the common gradient explosion or disappearance problems in RNN and LSTM, but also reduces the number of parameters and improves the computational efficiency. TCN consists of expanded, causal one-dimensional convolutional layers with the same input and output length, it extracts the features by applying convolution kernels to the input sequence. Taking one-dimensional full convolution as an example, the input tensor is denoted as C_i and the output tensor is denoted as C_{full}; Firstly, a series of continuous elements with a length of kernel-size are examined, and the convolution kernel multiplies the values at each position of continuous element and then sums them up, and the results are then stored in the one-dimensional $tensorC_{full}$, which is the result of full convolution of input tensor and convolution kernel. The RGB images typically contain rich spatial and temporal information, while TCN effectively captures the long-range dependencies while maintaining efficient computational performance through the use of causal convolution and dilated convolution structures.

Regarding one-dimensional causal convolution, the value at time t in the previous layer is only relied on that at time t in the next layer and those before it, its principle is shown in formula (13). It can predict y_t based on x_1, ..., x_t and y_1, ..., y_{t-1}, making y_t closer to actual value. Unlike traditional CNN, the causal convolution is a unidirectional structure rather than a bidirectional structure, that is, it cannot see the future data, which means that only with the preceding cause can be the following effect. Therefore, TCN is a strict time constrained model.

$$p(x) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}) \tag{13}$$

Due to the limitation of modeling length of time in causal convolution on the size of convolution kernel, it is necessary to linearly stack many layers once wondering to capture the longer dependencies. Dilated convolution allows the filter to be applied to the areas larger than the length of filter itself by

skipping some inputs, i.e. by adding zeros to generate a larger filter from the original filter, as shown in formula (14).

$$RF_i = RF_{i-1} + (k-1) \times s \tag{14}$$

In summary, TCN not only effectively captures the temporal features through its convolution-based architecture, but also identifies the potential attack patterns; And it also utilizes the causal convolution to ensure the orderliness of information, enabling the processing of long sequence data without being affected by gradient vanishing or exploding. In addition, the parallel computing of TCN make its training speed faster and can adapt to the analysis needs of large-scale network traffic. Therefore, TCN model is being selected as a base model.

3.3.5. BiTCN

Although TCN utilizes the convolutional network to effectively capture the features in time series, it can only capture the past contextual information, which leads to the problem of ignoring important future information, therefore, BiTCN is introduced to address the shortcomings of TCN. BiTCN model uses two temporal convolutional networks, one of which encodes the future covariates and the other encodes the past covariates and historical values of the sequence. This combination not only enables the model to learn the temporal information from network traffic, but also maintains the efficiency of convolution computation.

The network traffic represented by RGB images usually contains rich dynamic changes and contextual information, the use of BiTCN model can effectively extract the forward and backward features from sequences, which is crucial for understanding the contextual information of each frame image. In addition, the bidirectional learning not only allows the model to capture the long-term dependencies, but also enables it to utilize the parallelism of convolution operations to improve the efficiency of training and inference. The specific operation of convolutional layer of BiTCN is shown in formula (15), where Y_t is the output at time step t, X_t is the input sequence, W is the convolution kernel, * represents the convolution operation, b is the bias term and f is the activation function (usually ReLU or Sigmoid).

$$Y_t = f(W * X_t + b) \tag{15}$$

In the BiTCN model, the commonly used activation function is ReLU, but the use of this activation function can lead to the problem of neuronal "death". To solve this problem, a GELU activation function as shown in formula (16) is introduced into the BiTCN model.

$$GELU(x) = 0.5x[1 + tanh(\sqrt{\frac{2}{\pi}}(x + 0.047715x^3))] \tag{16}$$

Compared with the ReLU, GELU allows non-zero values to be returned with negative inputs when the input is less than zero, thereby providing richer gradients for backpropagation while maintaining model integrity.

In addition, the lagged value in BiTCN is combined with all past covariates before passing through the dense layer and time block stack, and the categorical covariate is first embedded and then combined with other covariates to make both past and future covariates are combined together, thereby outputting the combination of lagged value and covariate information. In the fully connected layer, BiTCN maps the high-dimensional features extracted by convolutional layer to the output space, thereby effectively transforming the complex patterns in the time series into specific decision information. The specific process is shown in formula (17), where Z is the fully connected layer used for final prediction, and its shape depends on the type of task; W_f is the weight matrix of a fully connected layer, usually represented as ([$noputput$, $ninput$]), where $noputput$ is the number of output neurons and $ninput$ is the number of input features (here is the dimension of Y).

$$Z = W_f \cdot Y + b_f \tag{17}$$

3.4. Multi-stacking ensemble learning strategy

After building five efficient base models, it is necessary to integrate the classification results through appropriate ensemble strategies. In recent years, many scholars have attempted to use the idea of ensemble learning for malicious traffic detection tasks [37][38], but most existing researches are based on traditional ensemble learning strategies such as voting, bagging and boosting. Among them, voting is a simple ensemble strategy that obtains the final prediction result by voting on the prediction results of multiple models; For the malicious traffic detection tasks, the voting ensemble strategy typically uses the hard voting or soft voting to integrate the detection results of base models, it then weights or counts the predicted values of base model to obtain the final detection result. Bagging resamples the original network traffic and generates multiple different training sets; And then, multiple base models are independently trained on these training sets and combined using voting or averaging to obtain the final detection result. Boosting trains multiple base models iteratively, where each new model focuses on the incorrectly predicted samples by the previous model and uses the weighted voting to combine the prediction results of different models.

Unlike voting, bagging and boosting, stacking ensemble can leverage the complementary information between different models while fully utilizing their advantages, thereby minimizing the potential bias and overfitting risks that each base model may bring. Specifically, stacking ensemble combines the prediction results of multiple different base models and uses cross validation to generate a new feature set to train a meta learner to optimize these prediction results, thereby improving the overall performance and generalization ability. Therefore, the stacking strategy is introduced into ensemble learning to construct ensemble models with better performance.

In order to reduce the time consumption of ensemble learning, this paper adopts a two-layer stacking structure as shown in Fig. 2. Firstly, in the first layer stacking, five malicious traffic detection models including CNN, LSTM, BiLSTM, TCN and

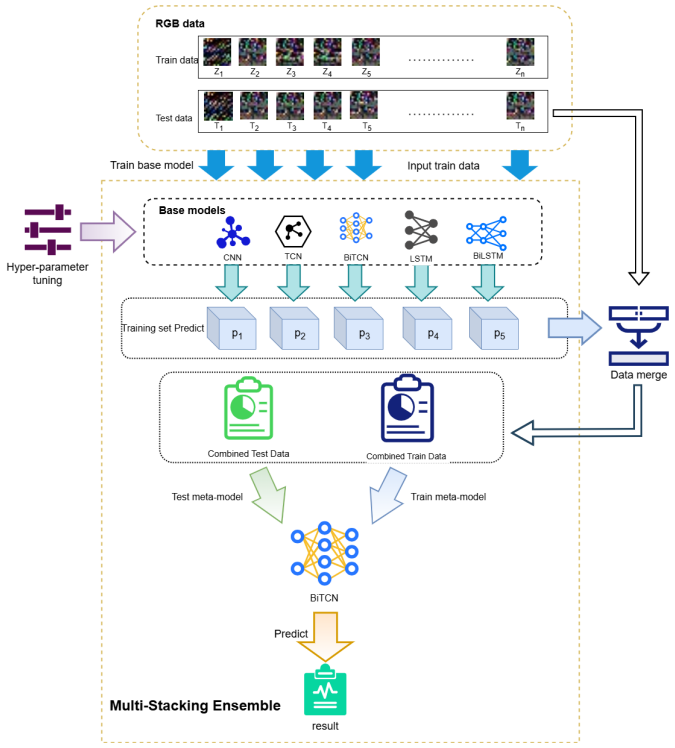


Fig. 2. The structure of two-layer stacking

BiTCN are selected as the base models, which train on the same network traffic dataset to calculate a prediction label value for each testing dataset to maximize the generalization and overall prediction ability; And then, these predicted values are concatenated and used as input data for the second layer meta learner. Due to the fact that BiTCN model not only combines the advantages of bidirectional convolution to simultaneously capture the past and future information in feature sequences, but also effectively captures the local patterns in time series, it is chosen as the meta learner in the second layer to obtain the best detection result. The specific process is shown as follows:

(1) First layer prediction: The network traffic datasets (containing n training samples and m testing samples, i.e. $Z = (z_1, z_2, z_3, ..., z_n)$ and $T = (t_1, t_2, ..., t_m)$) are input into the base models in first layer for training. For each base model (i = 1, 2, ..., 5), Z is used as the training set, and the prediction output for each base model is obtained as $q_i = (q_1, q_2, ..., q_5)$, where q_i is the prediction result vector for the i^{th} base model. The prediction results of all base models $(q_1, q_2, ..., q_5)$ are then used as the new feature vectors to input into the second layer.

(2) Second layer training: The prediction outputs of base models $(q_1, q_2, ..., q_5)$ are concatenated to form a new feature vector $Y = (q_1, q_2, q_3, q_4, q_5)$. And then, Y is used as the input to train the meta-learner BiTCN in the second layer. At this phase, Y serves as the training set for the meta-learner, and the model learns how to combine the predictions from base models to produce more accurate predictions.

(3) Second layer prediction: The testing set $T = (t_1, t_2, ..., t_m)$ is used as input to the meta model BiTCN to produce the prediction output, and the final output is the detection result of malicious traffic. The prediction from the second-layer model generates the final classification result of malicious traffic.

The execution process of malicious traffic detection model based on deep multi-stacking ensemble learning is shown in Algorithm 2. Firstly, the input network traffic dataset is converted into the feature representation of RGB images, and the RGB images are split into the training set Z and testing set T (see lines 1-2). And then, a list P is initialized to store the final results, and each base model is trained using the training set Z (see lines 3-4). In the subsequent loop, the testing set T is input into five base models (including CNN, TCN, BiTCN, LSTM and BiLSTM) to obtain the predictions of q_1, q_2, q_3, q_4, and q_5 (see lines 5-10). These predictions are concatenated to form $Y = (q_1, q_2, q_3, q_4, q_5)$ (see line 11). Finally, Y is used to train the meta-model BiTCN, which is then tested on T to produce the prediction *result* of meta model (see lines 12-13). The prediction *result* is added to P, and after the loop ends, P is returned as the final prediction result.

Algorithm 2 DMSE

Input: Network Traffic F
Output: Malicious traffic detection result P
1: $R = RGB\_generate(F)$ // Represent network traffic with RGB images
2: Z, T = random_split(R) //Split the RGB images to get training and testing sets
3: P = [] // store the final results
4: Train each base model with training set Z
5: **for** i in range(5) **do**
6: q_1 = CNN(T) // CNN prediction
7: q_2 = TCN(T) // TCN prediction
8: q_3 = BiTCN(T) //BiTCN prediction
9: q_4 = LSTM(T) // LSTM prediction
10: q_5 = BiLSTM(T) // BiLSTM prediction
11: Y = Integrate q_1, q_2, q_3, q_4, q_5 as $[q_1, q_2, q_3, q_4, q_5]$
12: Training meta model BiTCN with Y
13: *result* = BiTCN(Y) // meta-model prediction
14: Add *result* to P
15: **end for**
16: **return** P

4. Experiments and analysis

To validate the effectiveness of proposed DMSE model, we conduct extensive experiments on three widely used network traffic datasets of USTC-TFC2016, CTU and ISAC.

4.1. Description of network traffic datasets

To verify the efficiency of the proposed DMSE model, three network traffic datasets are used in the experiment. The number of different categories of network traffic and the distribution of train sets, validation sets and test sets are shown in Table 2.

USTC-TFC2016: This dataset is provided by the researchers from the University of Science and Technology of China for classification, it contains the network traffic from different applications, such as HTTP, FTP, SMTP, etc., including

Table 2
Specific information of used network traffic datasets

Dataset	Category	Train Set	Validate Set	Test Set	Dataset	Category	Train Set	Validate Set	Test Set	Dataset	Category	Train Set	Validate Set	Test Set
USTC-TFC2016	Cridex	14748	1638	1638	CTU	DownloadGuide	253	28	28	ISAC	Artemis	7825	869	869
	Geodo	36853	4094	4094		Dyreza	17	1	1		Coinminer	286	31	31
	Htbot	5731	636	636		Emotet	54000	6000	6000		Dridex	2820	313	313
	Miuref	12133	1348	1348		Ghost RAT	27	3	3		Htbot	8442	938	938
	Neris	30412	3379	3379		Ncurse	7228	803	803		Miuref	12133	1348	1348
	Nsisay	5463	606	606		OpenCandy	9180	1019	1019		Tinba	54000	6000	6000
	Shifu	8671	963	963		Ramnit	743	82	82		Trickbot	20315	2257	2257
	Tinba	7654	850	850		Sennoma	54000	6000	6000		Ursnif	34641	3849	3849
	Virut	29793	3310	3310		Shifu	23290	2587	2587		Normal	36645	4071	4071
	Zeus	9873	1097	1097		Normal	57689	6407	6407		/	/	/	/
	Normal	190127	21121	21121		/	/	/	/		/	/	/	/

10 categories of malicious traffic (Cridex, Geodo, Htbot, Miuref, Neris, Nsisay, Shifu, Tinba, Virut, Zeus) and 1 category of normal traffic.

CTU: This dataset is a mixed dataset of real botnet traffic and normal traffic captured at CTU University in Czech Republic. Among them, the CTU-13 dataset contains 13 different scenarios of network traffic samples generated by botnets, and the network traffic samples of each scenario are recorded in a PCAP file, while processing these PCAP files can obtain the type information of network traffic such as NetFlows, WebLogs, etc.

ISAC: This dataset consists of malicious traffic and normal traffic captured over a long period, sourced from the Malware Capture Facility Project. In this dataset, we select 9 categories of malicious traffic (including DownloadGuide, Dyreza, Enotet, Ghost-RAT, Ncurse, OpenCandy, Ramnit, Sennoma, and Shifu) and 1 category of normal traffic.

4.2. Baselines

To test the effectiveness of proposed DMSE model, five advanced ensemble learning-based malicious traffic detection models are selected as baselines in the experiment. The detailed descriptions of these models are as follows:

(1) Hossain and Islam [21]: It integrates the random forest, decision tree, logistic regression, SVM and AdaBoost through Gradient Boosting strategy, and then introduces the SMOTE technique for balancing the category of network traffic, thereby reducing the poor detection performance caused by data imbalance.

(2) RepuTE [25]: It is an ensemble learning-based malicious traffic detection model for classifying DoS/DDoS and Sybil attacks, it integrates the KNN, Extra Tree Classifier, Quadratic Discriminant Analysis Classifier, etc. as basic models with the use of soft voting strategy. In addition, a new feature selection technique is used to calculate the feature importance score and mutual information gain to filter relevant features, thereby improving the detection accuracy.

(3) Alghamdi and Bellaiche [26]: It is a Lambda architecture-based deep ensemble intrusion detection method, which combines LSTM, CNN and ANN models to identify different malicious traffic and adopts the majority voting strategy to obtain the better detection accuracy of malicious traffic.

(4) ENIDS [37]: It is a deep learning ensemble framework used for network intrusion detection, it first employs the SMOTE technology to sample the network traffic and constructs a stacking ensemble framework that utilizes the CNN, LSTM and GRU as base models, while uses the DNN as meta learner. The use of SMOTE alleviates the data imbalance issue, while the stacking strategy achieves better performance by leveraging multiple base models.

Table 3
Confusion matrix

Confusion matrix		Predict	
		P	N
Actual	P	TP (*True Positive*)	FN (*False Negative*)
	N	FP (*False Positive*)	TN (*True Negative*)

(5) DIS-IoT [38]: It is a new IoT intrusion detection method based on stacking ensemble, it adopts four deep learning models (including MLP, DNN, CNN, and LSTM) as base models and uses a fully CNN as meta learner to integrate the predictions of base models. The stacking ensemble strategy improves the ability of model to analyze diverse network traffic, improving both the generalizability and accuracy.

4.3. Evaluation metrics

To test the detection efficiency of DMSE model, this paper evaluates its effectiveness through four metrics: Accuracy, TPR, FPR and F1-score. These four metrics are calculated using the confusion matrix shown in Table 3.

In Table 3, TP indicates that the category is malicious traffic and the classification result is also malicious traffic; FP indicates that the category is normal traffic but the classification result is malicious traffic; FN indicates that the category is malicious traffic but the classification result is normal traffic; TN indicates that the category is normal traffic and the classification result is also normal traffic.

(1) Accuracy: It represents the proportion of correctly classified network traffic samples (including correctly identified malicious traffic samples TP and correctly identified normal traffic samples TN) to the total number of network traffic samples, it is calculated as shown in formula (18).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{18}$$

In the process of malicious traffic detection, Accuracy reflects the overall classification accuracy of the model on the entire network traffic samples. The higher accuracy indicates the model can more accurately distinguish the malicious traffic from normal network traffic.

(2) TPR: It represents the proportion of malicious traffic samples correctly identified by the model to the actual total number of malicious traffic samples, it is calculated as shown in formula (19).

$$TPR = \frac{TP}{TP + FN} \tag{19}$$

In the process of malicious traffic detection, TPR reflects the ability of the model to detect malicious traffic. A higher TPR means that the model can accurately capture more malicious traffic, reducing the possibility of missed detections.

(3) FPR: It represents the proportion of normal network traffic samples misclassified as malicious traffic by the model to the total number of network traffic samples, it is calculated as shown in formula (20).

$$FPR = \frac{FP}{FP + TN} \tag{20}$$

In the process of malicious traffic detection, FPR reflects the frequency of the model incorrectly identifying normal network traffic as malicious traffic. The lower FPR indicates the model less likely incorrectly marking normal traffic as malicious, which helps reduce false positive.

(4) F1-score: It represents the harmonic mean of precision and recall, where precision is the number of correctly identified malicious traffic samples divided by the total number of malicious traffic samples, and recall is the number of correctly identified malicious traffic samples divided by the total number of actual malicious traffic samples, it is calculated as shown in formula (21).

$$F1 - score = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{21}$$

In the process of malicious traffic detection, F1-score provides a performance metric that balances the precision and recall, and it is a comprehensive indicator of the model's ability to detect malicious traffic.

4.4. Experimental results

In order to better evaluate the detection performance of DMSE model, we set the following five research questions (RQs):

RQ 1. Whether the use of improved feature representation in RGB images can better represent the features of network traffic, thereby improving the detection accuracy of malicious traffic?

RQ 2. Whether the use of multi-stacking ensemble strategy can achieve better detection results compared to other ensemble strategies?

RQ 3. Does the choose of BiTCN model as meta learner achieve better detection performance than other models?

RQ 4. Does the proposed DMSE model achieve better detection performance compared to the state-of-the-art ensemble learning-based malicious traffic detection models?

RQ 5. Whether the proposed DMSE model obtain better stability compared to the state-of-the-art ensemble learning-based malicious traffic detection models?

To eliminate the randomness of experiment, 30 experiments are conducted each time, and the average result is calculated as the final experimental result. The value before the "±" symbol in the experimental result represents the average of 30 repeated experimental results, and the value after the "±" symbol represents the standard deviation of 30 repeated experimental results.

4.4.1. Answer to RQ1

To answer RQ1, we convert three network traffic datasets into the representation of gray-scale images, traditional RGB images and the improved RGB images, and then conduct experimental verification using deep learning models of CNN, RNN, TCN, LSTM, BiTCN, BiLSTM, ResNet and EfficientNet, respectively. The experimental results are shown in Table 4.

It can ben seen from Table 4 that the use of improved RGB images can improve the detection accuracy of malicious traffic for all eight deep learning models. Taking the BiTCN model as an example, on the USTC-TFC2016 dataset, the accuracy of improved RGB images is higher than that of traditional RGB images and gray-scale images by 2.98% and 7.18%, with the FPR is lower by 0.13% and 0.62%, with the TPR higher by 0.50% and 7.36%, with the F1-score higher by 3.38% and 7.59%; In addition, the standard deviation of accuracy of improved RGB images is lower than that of traditional RGB images and gray-scale images by 0.50% and 0.69%, with the standard deviation of FPR lower by 0.01% and 0.04%, with the standard deviation of TPR lower by 0.48% and 0.73%, and with the standard deviation of F1-score lower by 0.59% and 0.85%. On the CTU and ISAC datasets, the average accuracy of improved RGB images is also higher than that of traditional RGB images and gray-scale images, and the standard deviations of four evaluation metrics are also lower than that of other two representations. In the eight malicious traffic detection models, BiTCN model always obtain the best detection performance, and the BiLSTM model can obtain the second best detection performance, while the ResNet and CNN usually achieve the worse detection accuracy.

In summary, the proposed feature representation method is benefit for the detection models achieving higher accuracy, F1-score and TPR while maintaining the lowest FPR and standard deviations. This can be attributed to the fact that the improved feature representation stores the information from every layer and application layer separately in the R, G and B channels of RGB image, which allows RGB images to encapsulate a substantial amount of features of network traffic, permitting the malicious traffic detection models to comprehend both the global information and local features, thereby facilitating a more comprehensive and effective learning of the spatiotemporal relationships in network traffic.

Answer to RQ1:

Extensive experimental results show that the improved feature representation in RGB images can retain a greater and more effective features of network traffic, thereby enabling the detection model to accurately and efficiently detect malicious traffic.

4.4.2. Answer to RQ2

To answer RQ2, we compare the stacking ensemble strategy used in the DMSE model with five ensemble strategies (including Hard-voting, Soft-voting, Bagging, Adaboost and Gradient_boosting), where the format of network traffic used

Table 4
Detection performance of different base models

Feature representation	dataset	Metric	CNN	RNN	TCN	LSTM	BiTCN	BiLSTM	ResNet	EfficientNet
Improved RGB feature representation	USTC-TFC2016	Accuracy	94.40±0.68	92.26±1.22	95.64±0.64	93.58±0.63	97.09±0.52	96.05±0.53	89.92±1.30	90.60±1.24
		FPR	0.47±0.09	1.12±0.13	0.41±0.07	0.51±0.09	0.32±0.05	0.36±0.06	1.31±0.40	1.36±0.15
		TPR	94.66±0.73	92.55±1.03	95.64±0.60	93.66±0.82	97.11±0.53	95.99±0.59	86.65±1.49	90.77±1.32
		F1-score	94.82±0.88	92.68±1.03	95.55±0.50	94.04±0.88	97.15±0.43	95.63±0.52	82.60±1.11	90.50±1.25
	CTU	Accuracy	92.83±0.88	90.86±1.03	94.33±0.74	92.11±0.78	97.71±0.28	96.95±0.53	90.29±1.17	90.86±1.07
		FPR	0.77±0.08	1.34±0.17	0.71±0.07	0.95±0.09	0.16±0.03	0.41±0.04	1.37±0.28	1.38±0.18
		TPR	92.55±0.91	90.33±1.18	94.36±0.69	92.17±0.77	97.66±0.48	97.26±0.54	90.04±1.20	90.71±1.23
		F1-score	92.89±0.66	90.37±0.78	94.54±0.64	92.40±0.72	97.60±0.49	97.34±0.57	89.71±0.96	90.88±0.80
	ISAC	Accuracy	94.22±0.94	92.53±1.15	95.86±0.60	93.73±0.92	98.02±0.57	96.75±0.58	88.61±1.52	90.72±1.18
		FPR	0.39±0.09	1.17±0.21	0.38±0.06	0.40±0.07	0.19±0.05	0.35±0.06	1.20±0.23	1.23±0.25
		TPR	94.40±0.81	92.73±1.13	95.81±0.60	94.56±1.05	97.84±0.59	97.03±0.37	88.43±1.85	91.21±1.32
		F1-score	94.36±0.75	93.42±0.91	96.04±0.73	94.70±0.88	97.95±0.67	97.24±0.21	88.57±1.31	92.28±1.30
Traditional RGB feature representation	USTC-TFC2016	Accuracy	89.45±1.31	86.41±1.52	92.62±1.20	91.73±1.36	94.11±1.02	92.79±1.11	87.93±1.58	88.04±1.67
		FPR	1.18±0.15	1.30±0.27	1.08±0.12	1.21±0.14	0.45±0.06	0.88±0.07	1.59±0.50	1.62±0.24
		TPR	90.34±1.34	86.30±1.52	94.53±1.25	90.75±1.30	94.61±1.01	92.21±1.13	81.84±1.58	83.61±1.54
		F1-score	86.34±1.33	85.48±1.52	87.33±1.21	86.63±1.32	93.77±1.02	92.14±1.17	80.86±1.56	83.30±1.55
	CTU	Accuracy	87.39±1.54	87.48±2.03	91.40±1.43	87.92±1.52	93.35±1.00	92.30±1.30	86.07±1.76	86.05±1.81
		FPR	1.54±0.15	2.59±0.21	1.34±0.14	1.51±0.15	0.94±0.08	1.18±0.13	2.24±0.39	2.21±0.24
		TPR	87.16±1.66	85.53±1.79	88.81±1.44	84.76±1.51	89.12±1.12	88.38±1.32	83.49±1.78	83.61±1.78
		F1-score	86.42±1.51	85.78±1.72	86.94±1.44	86.66±1.50	94.80±1.18	89.63±1.36	85.91±1.67	85.51±1.79
	ISAC	Accuracy	90.67±1.68	86.23±1.84	91.50±1.50	90.76±1.63	93.19±1.00	92.43±1.26	86.16±1.91	87.43±1.93
		FPR	1.31±0.20	1.81±0.29	1.36±0.15	1.42±0.21	0.39±0.08	0.58±0.08	2.84±0.35	1.51±0.35
		TPR	88.08±1.64	84.13±1.80	90.37±1.59	88.35±1.65	92.03±1.09	89.83±1.29	84.87±1.90	87.35±1.97
		F1-score	88.73±1.66	85.21±1.90	90.13±1.52	88.30±1.64	92.26±1.00	91.61±1.26	87.99±2.27	87.35±1.92
Gray-scale feature representation	USTC-TFC2016	Accuracy	85.62±1.62	85.09±1.70	86.41±1.54	85.65±1.63	89.91±1.21	89.77±1.17	84.22±1.71	84.63±1.74
		FPR	1.44±0.17	1.57±0.31	1.26±0.13	1.34±0.16	0.94±0.09	0.98±0.09	1.77±0.55	1.65±0.25
		TPR	85.62±1.36	84.98±1.68	81.40±1.29	81.58±1.37	89.75±1.26	89.73±1.32	78.27±1.82	79.42±1.68
		F1-score	85.35±1.54	84.78±1.62	86.78±1.44	81.72±1.58	89.56±1.28	89.44±1.35	75.37±1.63	81.39±1.74
	CTU	Accuracy	85.42±1.62	85.38±2.09	85.62±1.53	85.57±1.61	88.62±1.28	86.94±1.33	85.12±2.23	84.52±2.37
		FPR	1.68±0.19	2.61±0.23	1.54±0.17	1.67±0.42	1.38±0.11	1.43±0.16	2.63±0.42	2.71±0.25
		TPR	84.47±1.74	83.63±1.83	86.76±1.49	83.69±1.64	87.83±1.35	85.62±1.57	83.26±1.83	81.74±1.82
		F1-score	85.73±1.64	85.64±1.67	85.93±1.58	84.73±1.72	88.47±1.46	86.75±1.52	85.37±1.74	85.25±1.88
	ISAC	Accuracy	85.72±1.77	85.49±1.87	86.37±1.69	86.34±1.84	89.35±1.13	88.47±1.62	84.63±2.18	85.27±2.37
		FPR	1.62±0.25	1.92±0.32	1.58±0.23	1.87±0.26	0.74±0.09	0.86±0.09	2.87±0.39	1.96±0.38
		TPR	85.65±1.68	85.17±1.98	85.74±1.64	84.73±1.76	87.96±1.28	87.54±1.52	84.26±2.36	84.24±2.26
		F1-score	85.49±1.75	84.76±2.04	86.22±1.57	86.12±1.68	88.72±1.33	87.83±1.39	84.73±2.48	84.36±2.03

Table 5
Detection performance of different ensemble strategies (%)

Dataset	Metrics	Hard voting	Soft voting	Adaboost	Gradient Boosting	Bagging	Multi-stacking
USTC-TFC2016	Accuracy	97.22±0.43	97.73±0.33	97.93±0.69	97.84±0.42	95.11±1.15	**98.65±0.17**
	FPR	0.26±0.03	0.24±0.04	0.18±0.04	0.17±0.03	1.65±0.13	**0.13±0.02**
	TPR	95.96±0.49	96.84±0.73	97.83±0.86	97.63±0.57	96.46±0.75	**98.59±0.18**
	F1-score	96.12±0.37	97.16±0.48	97.39±0.58	96.23±0.62	94.85±1.48	**98.65±0.13**
CTU	Accuracy	97.13±0.45	97.42±0.61	97.86±0.42	97.94±0.55	95.60±0.37	**99.07±0.29**
	FPR	0.59±0.08	0.54±0.06	0.36±0.05	0.42±0.04	2.53±0.19	**0.14±0.03**
	TPR	94.76±0.94	95.72±0.49	97.41±0.58	96.83±0.44	93.50±3.60	**99.16±0.24**
	F1-score	96.27±0.45	96.38±0.53	96.84±0.69	96.62±0.56	95.53±2.00	**99.08±0.28**
ISAC	Accuracy	97.58±0.88	97.96±0.52	98.32±0.48	98.57±0.64	92.27±0.13	**99.57±0.04**
	FPR	0.18±0.07	0.16±0.08	0.14±0.06	0.11±0.05	4.14±0.39	**0.05±0.01**
	TPR	96.82±0.54	96.48±0.37	98.17±0.29	97.83±0.34	93.35±0.18	**99.62±0.04**
	F1-score	97.41±0.79	97.74±0.64	97.46±0.36	98.36±0.42	91.11±0.17	**99.56±0.03**

in this experiment is the proposed improved RGB images, the base models used in the stacking ensemble strategy is the best performing CNN, TCN, BiTCN, LSTM and BiLSTM, and the meta learner is BiTCN. The experimental results are shown in Table 5, where the best detection results are highlighted in bold.

Table 5 indicates that the stacking strategy outperforms other five ensemble learning strategies of hard voting, soft voting, AdaBoost, gradient boosting and bagging on the three publicly available network traffic datasets. On the USTC-TFC2016 dataset, the stacking strategy achieves a detection accuracy of 98.65% with a standard deviation of 0.17%, a TPR of 98.59% with a standard deviation of 0.18%, and a F1-score of 98.55% with a standard deviation of 0.18%, these three metrics are the highest among six ensemble strategies, while the FPR reaches at 0.13% with a standard deviation of 0.02%, it is the lowest among six strategies. On the CTU dataset, the stacking strategy outperforms the second-best gradient boosting by 1.13% in accuracy, 2.30% in TPR and 2.46% in F1-score; Compared to the

Table 6
Detection performance of different meta learners in multi-stacking ensemble framework (%)

Dataset	Metrics	CNN	RNN	TCN	LSTM	BiLSTM	ResNet	EfficientNet	BiTCN
USTC-TFC2016	Accuracy	95.89±0.95	94.82±1.09	97.78±0.51	95.73±0.88	97.19±0.66	95.70±1.35	94.19±1.39	**98.65±0.17**
	FPR	0.48±0.07	1.13±0.20	0.18±0.05	0.46±0.09	0.31±0.08	0.56±0.09	1.04±0.28	**0.13±0.02**
	TPR	95.98±0.96	94.77±1.10	97.81±0.85	95.82±0.86	97.04±0.73	95.79±1.14	94.23±1.37	**98.59±0.18**
	F1-score	95.16±1.11	93.38±1.25	97.79±0.31	95.16±1.00	96.80±0.70	95.68±1.31	92.58±2.42	**98.65±0.13**
CTU	Accuracy	96.57±0.73	93.98±2.79	97.73±0.29	95.95±0.76	97.65±0.15	94.06±0.60	94.32±1.76	**99.07±0.29**
	FPR	0.52±0.09	94.05±2.73	0.41±0.08	0.69±0.09	0.41±0.04	0.87±0.04	1.06±0.63	**0.14±0.03**
	TPR	96.65±0.69	1.08±0.73	97.81±0.30	96.01±0.73	97.76±0.27	94.41±0.79	94.86±1.79	**99.16±0.24**
	F1-score	96.18±0.99	93.57±2.32	97.67±0.29	95.46±1.01	97.58±0.15	94.04±0.60	93.76±1.77	**99.08±0.28**
ISAC	Accuracy	95.59±0.96	93.68±1.35	97.88±0.90	95.81±1.20	98.28±0.45	96.02±1.48	96.82±1.88	**99.57±0.04**
	FPR	1.29±0.48	2.35±0.53	0.31±0.07	0.50±0.09	0.25±0.03	0.49±0.09	1.01±0.15	**0.05±0.01**
	TPR	95.66±1.10	94.38±1.41	98.41±0.69	95.91±1.57	98.47±0.38	96.35±1.46	96.93±1.82	**99.62±0.04**
	F1-score	94.71±1.21	91.21±1.76	98.28±0.47	95.48±1.62	98.50±0.47	96.19±1.37	96.18±2.45	**99.56±0.03**

worst-performing bagging ensemble, it improves the accuracy by 3.47%, TPR by 5.63%, and F1-score by 3.55%, with FPR reduces by 2.39%. On the ISAC dataset, the stacking strategy achieves an accuracy of 99.57% with a standard deviation of 0.04%, which is outperforming hard voting by 1.99%, soft voting by 1.61%, AdaBoost by 1.25%, and gradient boosting by 1.00%. Additionally, the stacking strategy records a F1-score of 99.56% with a standard deviation of 0.03%, and a TPR of 99.62% with a standard deviation of 0.04%, while the performance of other five ensemble strategies are slightly inferior.

Overall, the use of multi-stacking ensemble strategy can promote the malicious traffic detection models obtain better detection efficiency than other ensemble strategies in the metrics of accuracy, TPR, FPR and F1-score, this is mainly because the multi-stacking ensemble strategy allows for the use of more complex relational models as meta-learners, which can improve the generalization capability of model and further improve the detection performance.

Answer to RQ2:

Extensive experimental results show that the use of multi-stacking ensemble strategy can more effectively leverage the advantages of each base model, thereby achieving more precise detection of malicious traffic.

4.4.3. Answer to RQ3

To answer RQ3, we use different meta learners (including CNN, RNN, TCN, LSTM, BiTCN, BiLSTM, ResNet and EfficientNet) in the stacking ensemble strategy, and the experimental results are shown in Table 6, where the best detection results are highlighted in bold.

As is shown in Table 6 that on the USTC-TFC2016 dataset, the use of BiTCN as meta learner can achieve an accuracy of 98.65%, TPR of 98.59%, FPR of 0.13%, and F1-score of 98.59%; Compared to other meta learners, the accuracy of BiTCN is 2.76% higher than CNN, 3.83% higher than RNN, 0.87% higher than TCN, 2.92% higher than LSTM, 1.46% higher than BiLSTM, 2.95% higher than ResNet and 4.46% higher than EfficientNet; In addition, the FPR is lower than CNN by 0.35%, and the TPR and F1-score also show the similar advantages. On the CTU dataset, the accuracy of BiTCN is 2.50% higher than CNN, 5.09% higher than RNN and 1.34% higher than TCN; The FPR is lower than CNN by 0.38%; It also performs well in the metrics of TPR and F1-score. On the ISAC dataset, the accuracy, TPR and F1-score of BiTCN model are also the highest, and its FRP metric is the lowest.

The experimental results show that compared with other seven models, the use of BiTCN as meta learner can achieve better performance in malicious traffic detection. The reason for appearing this situation is that the unique bidirectional structure of BiTCN can simultaneously capture both forward and backward information in time series, which means that this model is no longer limited to unidirectional information flow, allowing it to uncover more complex patterns and associations hidden in the network traffic. Additionally, through efficient convolutional kernel configurations, BiTCN can accurately and quickly extract the local key features, enabling the detection model grasp the information across different scales of both short-term and long-term contexts, thereby sensitively detecting both subtle changes and trending directions.

Answer to RQ3:

Extensive experimental results show that the use of BiTCN as meta learner achieves better detection results in comparison to other seven meta learners in the metrics of accuracy, TPR, FPR and F1-score.

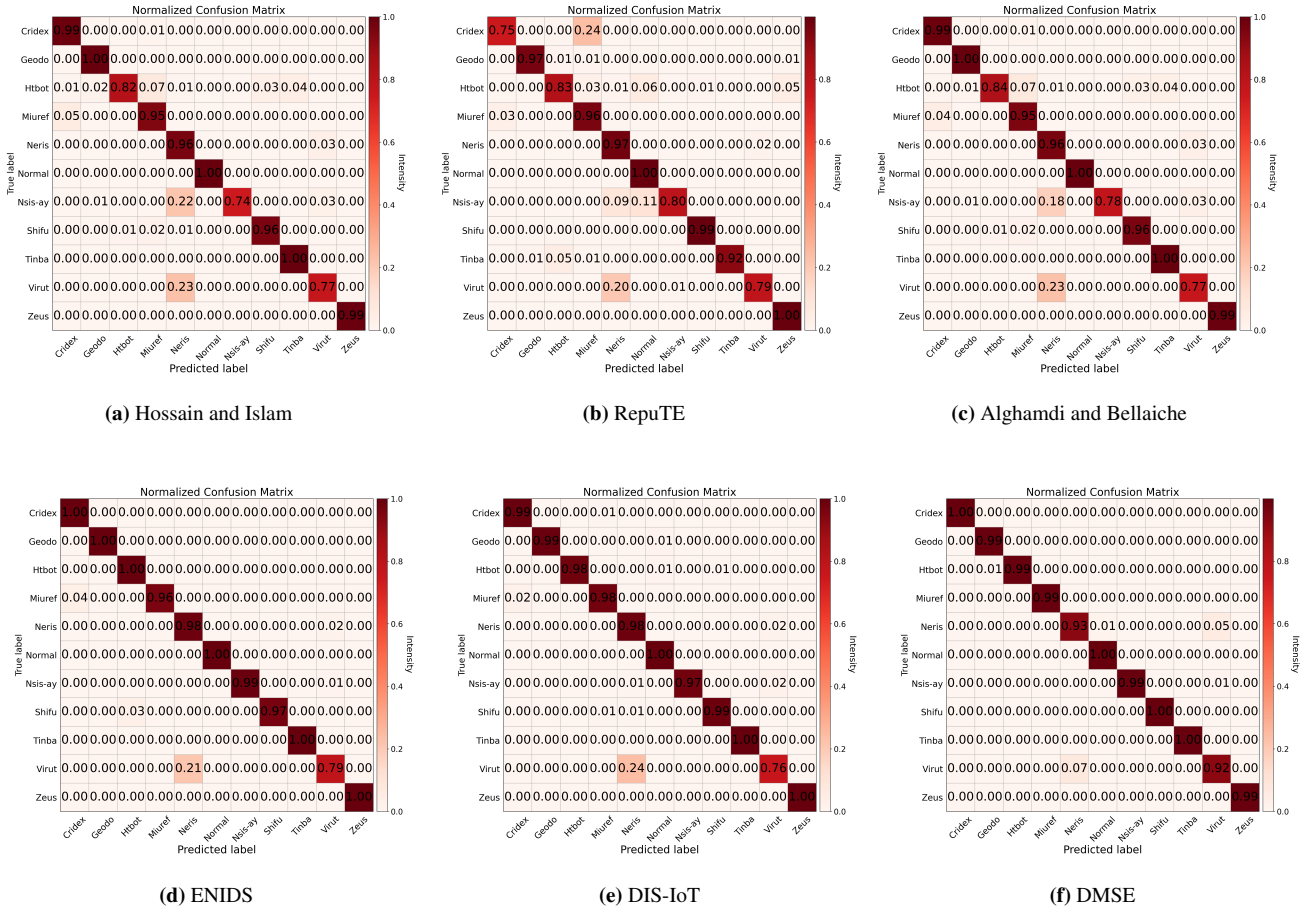
4.4.4. Answer to RQ4

To answer RQ4, we conduct experimental comparisons between the proposed DMSE model and five state-of-the-art ensemble learning based malicious traffic detection models (including RepuTE [25], ENIDS [37], DIS-IoT [38] and the models proposed by Hossain and Islam [21], Alghamdi and Bellaiche [26]). The experimental results are shown in Table 7, Fig. 3 to Fig. 6, where the best detection results are highlighted in bold.

It can be seen from Table 7 that on the USTC-TFC2016 dataset, the accuracy of DMSE is 98.65% (standard deviation:

Table 7
Detection performance of different ensemble learning-based malicious traffic detection models (%)

Dataset	Metrics	Hossain and Islam	RepuTE	Alghamdi and Bellaiche	ENIDS	DIS-IoT	DMSE(proposed)
USTC-TFC2016	Accuracy	96.56 ± 0.35	95.38 ± 0.59	96.55 ± 0.32	97.87 ± 0.40	97.11 ± 0.30	**98.65±0.17**
	FPR	0.31 ± 0.04	0.40 ± 0.09	0.29 ± 0.04	0.18 ± 0.08	0.23 ± 0.04	**0.13±0.02**
	TPR	96.63 ± 0.39	95.38 ± 0.51	96.28 ± 0.52	97.83 ± 0.21	97.37 ± 0.46	**98.59±0.18**
	F1 score	96.39 ± 0.35	95.15 ± 0.58	96.57 ± 0.14	98.21 ± 0.42	97.17 ± 0.20	**98.65±0.13**
CTU	Accuracy	96.09 ± 0.90	95.23 ± 1.01	96.38 ± 0.45	98.31 ± 0.58	97.47 ± 0.36	**99.07±0.29**
	FPR	0.36 ± 0.05	0.46 ± 0.07	0.31 ± 0.05	0.18 ± 0.06	0.36 ± 0.08	**0.14±0.03**
	TPR	92.51 ± 1.06	91.12 ± 0.72	96.74 ± 0.98	96.20 ± 0.32	97.24 ± 0.45	**99.16±0.24**
	F1 score	93.30 ± 0.67	90.76 ± 0.57	96.42 ± 0.42	96.45 ± 0.88	97.12 ± 0.30	**99.08±0.28**
ISAC	Accuracy	96.00 ± 1.00	95.14 ± 0.78	95.77 ± 0.44	97.90 ± 0.49	97.22 ± 0.21	**99.57±0.04**
	FPR	0.35 ± 0.08	0.42 ± 0.05	0.42 ± 0.04	0.25 ± 0.05	0.32 ± 0.05	**0.05±0.01**
	TPR	93.02 ± 0.22	91.43 ± 0.36	91.87 ± 0.37	97.84 ± 0.65	97.33 ± 0.49	**99.62±0.04**
	F1 score	94.15 ± 0.65	90.90 ± 0.58	92.13 ± 0.84	97.82 ± 0.40	97.10 ± 0.29	**99.56±0.03**



(a) Hossain and Islam **(b)** RepuTE **(c)** Alghamdi and Bellaiche

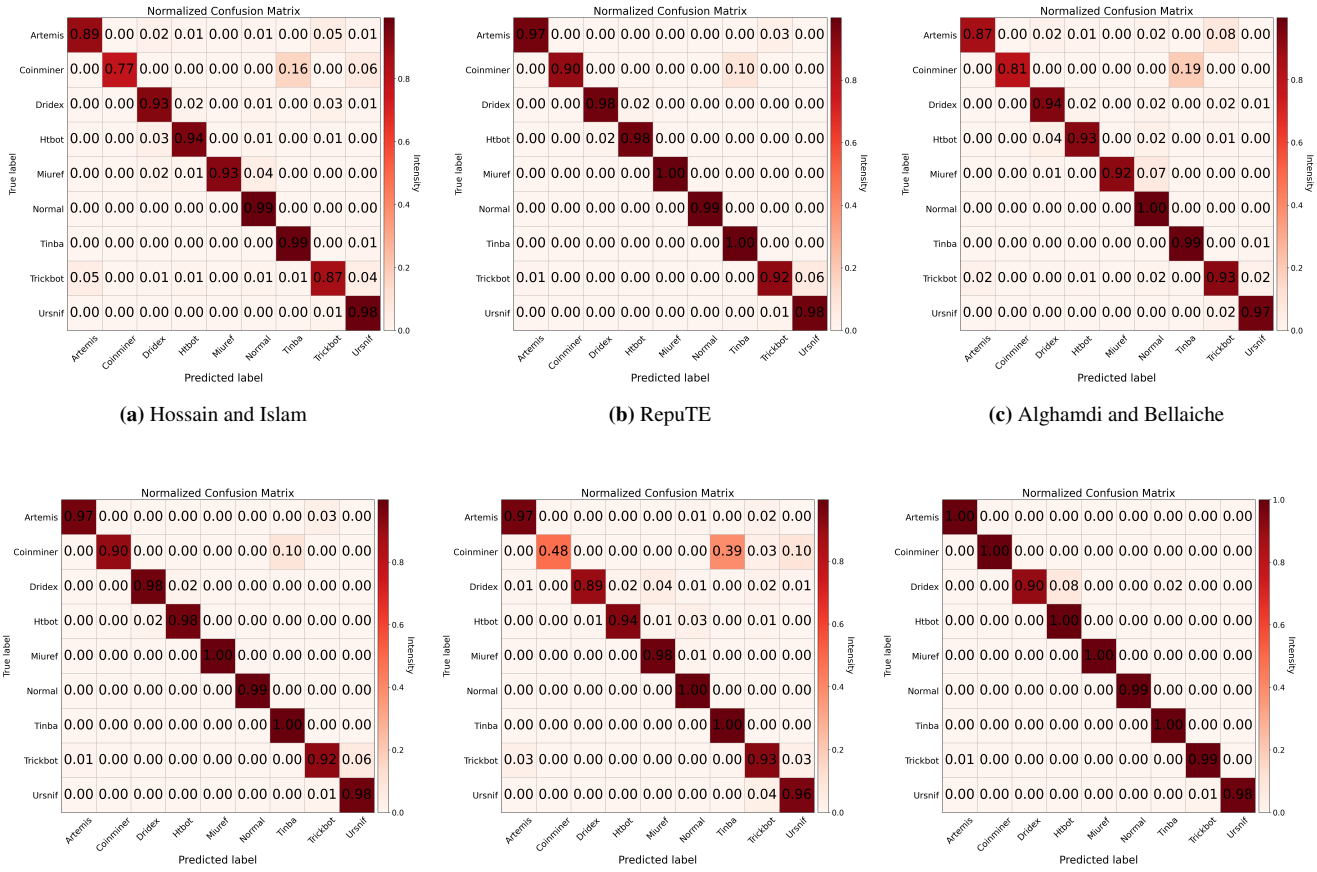
(d) ENIDS **(e)** DIS-IoT **(f)** DMSE

Fig. 3. The confusion matrix of six ensemble learning-based malicious detection models on USTC-TFC2016 dataset

0.17%), the TPR is 98.59% (standard deviation: 0.18%), the FPR is 0.13% (standard deviation: 0.02%), and the F1-score is 98.55% (standard deviation: 0.18%). The accuracy of DMSE is 2.09%, 3.27%, 2.1%, 0.78%, and 1.54% higher than Hossain and Islam, RepuTE, Alghamdi and Bellaiche, ENIDS and DIS-IoT, respectively; The FPR is 0.18%, 0.27%, 0.16%, 0.05%, and 0.10% lower, respectively. The TPR is 1.96%, 3.21%, 2.31%, 0.76%, and 1.22% higher, respectively; And the F1-score is 2.26%, 3.50%, 2.08%, 0.44%, and 1.48% higher than those models, respectively.

In addition, the confusion matrices of the experimental results in Fig. 3 to Fig. 5 also show that the proposed DMSE model performs well in malicious traffic detection, it can achieve better detection accuracy across all categories compared to other five advanced ensemble learning-based malicious traffic detection models.

The reason for the better detection performance of DMSE model is that it adopts the deep learning models as base models, which can automatically extract the features and possess the stronger learning capabilities. Moreover, the two-layer stacking ensemble strategy can leverage the advantages of each base model more effectively compared to simpler voting, boosting or bagging strategies, thereby improving the model's generalization ability and detection accuracy. Additionally, the used



(a) Hossain and Islam **(b)** RepuTE **(c)** Alghamdi and Bellaiche

(d) ENIDS **(e)** DIS-IoT **(f)** DMSE

Fig. 4. The confusion matrix of six ensemble learning-based malicious detection models on CTU dataset

base models of CNN, TCN and BiTCN are good at extracting the image information, as well as LSTM and BiLSTM can effectively process the temporal information; The advantages of these used base models allow the proposed DMSE model to utilize the information from network traffic more comprehensively.

Answer to RQ4:

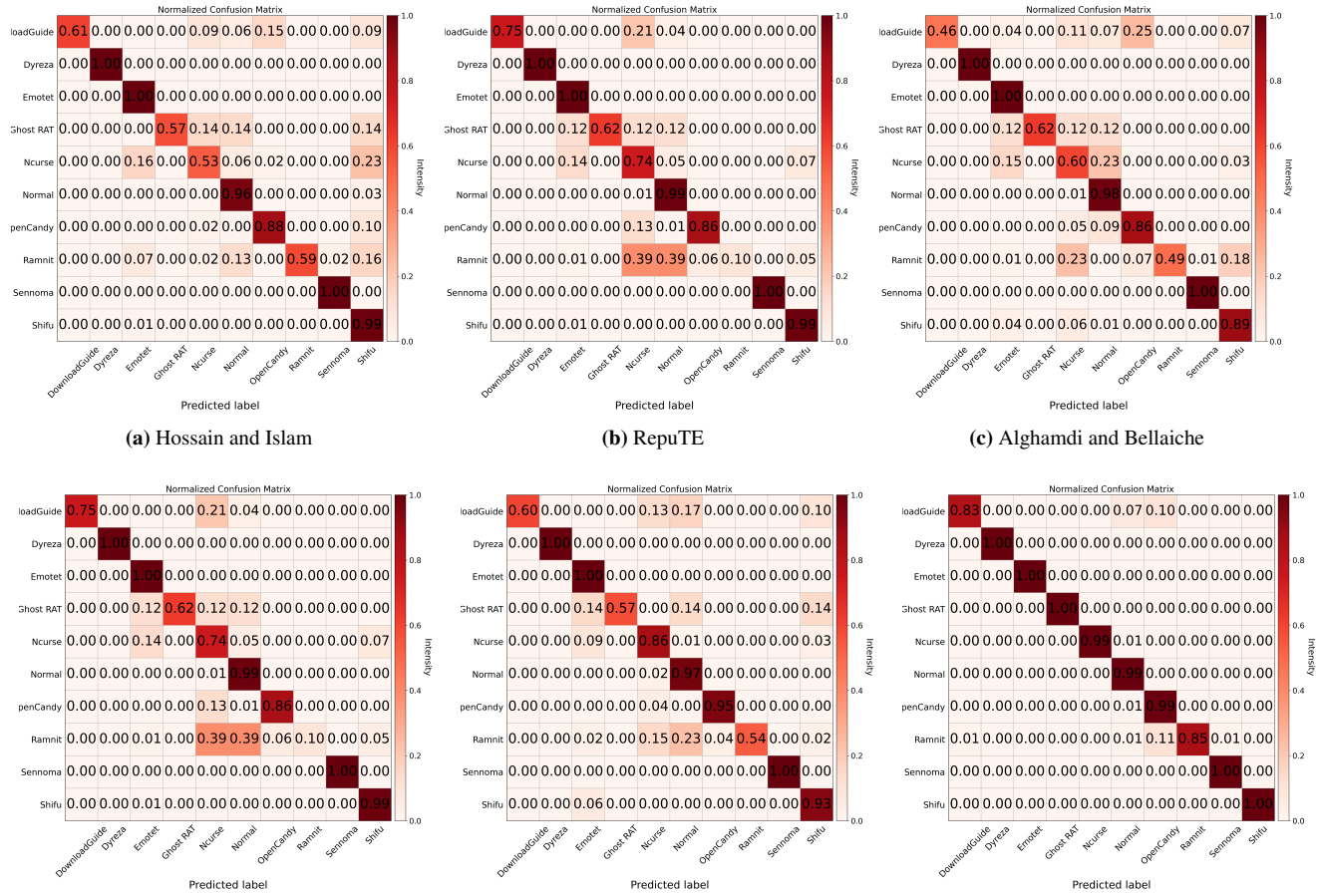
Extensive experimental results show that compared with the state-of-the-art ensemble learning-based malicious traffic detection models, the proposed DMSE model performs better in detection accuracy on three public network traffic, which indicates that it can be effectively used to maintain cyber security.

4.4.5. Answer to RQ5

To answer RQ5, we also conduct extensive experiments to verify the stability of six compared ensemble learning-based malicious traffic detection models, and the experimental results are shown in Fig. 6.

As is shown in Fig. 6 that on these three network traffic datasets, the proposed DMSE model exhibits the smallest interquartile range (box length) among four indicators compared to five advanced ensemble learning-based malicious traffic detection models, and it has no outliers, indicating that the proposed DMSE model has better detection stability. As is shown in Fig. 6a that DMSE outperforms the models proposed by Hossain and Islam, Alghamdi and Bellaiche, RepuTE, ENIDS and DIS-IoT, in terms of accuracy; As is illustrated in Fig. 6c and Fig. 6d that the DMSE model maintains the highest positions in TPR and F1-score, indicating that DMSE exhibits superior performance in malicious traffic detection tasks. Furthermore, as shown in Fig. 6b that the FPR of DMSE is the lowest on all three network traffic datasets, which also indicates that DMSE model can reduce the probability of misclassifying the normal network activities as malicious traffic attacks.

Compared with five ensemble learning-based malicious traffic detection models, the proposed DMSE model demonstrates better stability for the following reasons: (1) It employs a multi-stacking ensemble strategy that combines the advantages of multiple base learners, which can reduce the overfitting risk of individual models; (2) Traditional ReLU activation function is replaced by ELU to address the neuron death problem, which helps the DMSE model better capture the features of network traffic; (3) The BiTCN is adopted as meta-learner in second layer stacking, its rapid adaptation and generalization abilities enable the efficient feature extraction and anomaly detection of DMSE in different network environment.



(a) Hossain and Islam **(b)** RepuTE **(c)** Alghamdi and Bellaiche

(d) ENIDS **(e)** DIS-IoT **(f)** DMSE

Fig. 5. The confusion matrix of six ensemble learning-based malicious detection models on ISAC dataset

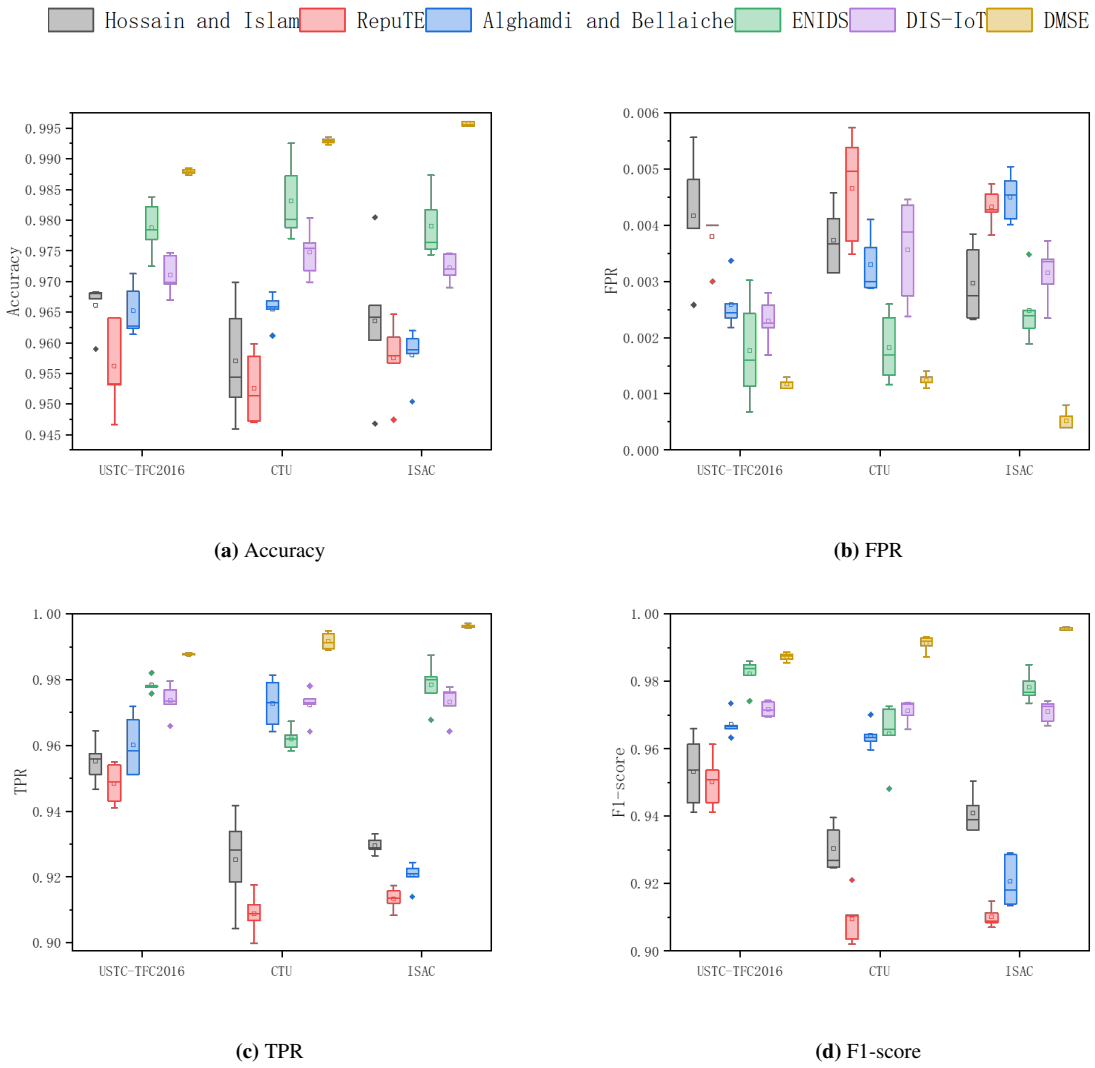
Answer to RQ5:

Extensive experimental results show that compared with the advanced ensemble learning-based malicious traffic detection models, the proposed DMSE model has better stability, which makes it suitable for the use of intrusion detection.

4.5. Discussion

This paper presents a novel malicious traffic detection model called DMSE, it is constructed around three aspects: feature representation of network traffic, base model detection and multi-stacking ensemble learning. This model first extracts the features of original network traffic from the PCAP files by obtaining all layers and application layer data packets of different types, and the extracted features are represented as RGB images, which assist the model better learning the global and local information of network traffic. The generated RGB images are then fed into the base models to obtain initial predictions from the first layer, and the predictions from base models are collected and concatenated to serve as training data for the meta learner in second layer. Finally, the trained meta learner is used to re-predict the testing dataset, thereby obtaining the final prediction results. The feature representation of network traffic in RGB images and the combination of two-layer multi-stacking architecture allow the model more effectively detect malicious traffic.

Although the proposed DMSE model achieves better detection results, there are still some issues to solve: **(1) Model Practicality:** We validate the detection performance of proposed DMSE model on three public network traffic datasets. The experimental results show that DMSE demonstrates superior performance in accuracy, TPR, FPR, F1-score and stability; Consequently, DMSE possesses strong practicality in malicious traffic detection. However, it requires to convert the network traffic into RGB images, combines with ensemble deep learning models and relatively complex two-layer multi-stacking architecture, makes it challenging to meet the scenarios that demand real-time performance. In the future, we plan to improve the time efficiency of DMSE through the following manners: (I) Employ the lightweight base models to minimize the time requirements for the first-layer stacking; (II) Appropriately reduce the number of network layers in the deep learning models to improve the efficiency of malicious traffic detection; (III) Utilize the parallel computing methods, leverage the multi-core processing or distributed computing framework to simultaneously train multiple base models, thereby decreasing time consumption. **(2) Limitations of the used datasets:** Due to the dynamic nature of network traffic, the features in these three network traffic datasets may not cover all features of malicious



(a) Accuracy

(b) FPR

(c) TPR

(d) F1-score

Fig. 6. Stability of different data augmentation-based malicious traffic detection models

traffic, leading to difficulties in detecting novel malicious traffic based on existing network traffic for DMSE model. In the future, we intend to address the dataset limitations through the following measures: (I) Adopt the data augmentation techniques by transforming the RGB images of network traffic to improve the generalization capability of model under limited network traffic; (II) Generate the synthetic network traffic using GAN or other generative models to fill the gaps within the datasets; (III) Use the SMOTE sampling techniques to generate synthetic samples for malicious traffic (minority class) to increase its volume, thereby addressing the data imbalance issues. **(3) Adaptability in different network environments:** The category of malicious traffic is continually updated alongside the developments of network environment, causing the trained DMSE model cannot effectively detect novel malicious traffic in real life. To improve the adaptability of DMSE model in varying circumstances, we would like to: (I) Introduce an automated update mechanism for model, which utilizes the real-time monitoring and feedback systems to automatically detect and trigger updates when the performance of DMSE model declines; (II) Compress and optimize the model, which employs the model distillation and pruning methods to solve the issues with real-time transmission or execution in unstable network environments.

4.6. Threats to validity

Although the experimental results in subsections 4.1-4.5 confirm that the proposed DMSE model makes the breakthroughs in the detection of malicious traffic with better detection performance, this study still presents a series of potential validity issues: (1) We employ three widely used network traffic datasets for extensive experiments to verify the efficiency of DMSE model, but these three network traffic datasets may not cover all types of network traffic, indicating a need to verify its capability to handle diverse types of network traffic when deploying it into real-world scene; (2) Given that some existing efficient network malicious traffic detection methods (such as RepuTE, ENIDS and DIS-IoT) do not release their source codes, we can only reproduce these methods based on our understanding, which might deviate from the original intentions of authors; (3) In the selection of evaluation metrics, we choose the metrics (including accuracy, TPR, FPR and F1-score) that can better reflect the performance of model, but the time efficiency of DMSE model may be challenged in real-time scenarios during actual deployment; (4) The proposed DMSE model employs the BiTCN model as a meta learner for its outstanding detection performance, but it is unclear whether hackers might target this model with malicious attacks, posing a threat to system security; (5) We determine the base model combinations

and parameter settings in the DMSE model through comparison and achieves expected results on testing network traffic datasets; Nonetheless, whether DMSE is the optimal model in complex and dynamic network traffic detection scenarios needs further deliberation; (6) In the feature representation phase of DMSE model, we transform the PCAP packets of network traffic into RGB images through cutting and reshaping operations, this process involves adjusting color channel to present richer information, but it is yet to be confirmed if all features of network traffic can be well-preserved in various detection scenarios; In addition, this conversion process demands high computational resources, potentially leading to efficiency decreases during actual deployment.

5. Conclusion

With the rapid development of internet, the number of network traffic generated by various devices is very large, while the growing presence of malicious traffic poses increasing threats to cyber security, therefore, there is an urgent need for an efficient malicious traffic detection model to accurately identify abnormal activities, thereby improving the security of cyberspace. Despite the recent advances in malicious traffic detection models, most existing methods are based on single machine learning model or deep learning model, resulting in difficult effectively dealing with multiple types of network traffic. This paper proposes an efficient malicious traffic detection model called DMSE based on deep multi-stacking ensemble learning, which is composed of feature representation module, base model detection module and multi-stacking ensemble learning module, these three modules collectively perform malicious traffic detection.

Firstly, in the feature representation module, DMSE stratifies and transforms the network traffic and places the transformed features of network traffic into three channels of RGB images, which can improve the hierarchical representation of network traffic as well as allow the model to learn both global and local features. Subsequently, in the base model detection module, each base model (including CNN, TCN, LSTM, BiLSTM, BiTCN) processes the network traffic represented by RGB images to generate prediction outputs, which serve as the input for training the meta learner. In the multi-stacking ensemble learning module, these outputs are combined and used as training set, thereby obtaining the final detection results through utilizing the RGB-transformed testing set. Through employing the multi-stacking ensemble strategy, the strengths of each base model are fully utilized to improve the accuracy and generalization capability of malicious traffic detection. Extensive experimental results on three public network traffic datasets demonstrate that the proposed DMSE model outperforms existing ensemble learning-based malicious traffic detection methods in terms of accuracy, TPR, FPR and F1-score, and exhibits superior stability.

Although the proposed DMSE model achieves better malicious traffic detection performance, it also encounters some challenges such as difficult processing network traffic in real-time. We would like to improve the DMSE model through following aspects: (1) Employ the lightweight base models to improve the time efficiency in the first layer of multi-stacking framework, thereby reducing overall detection time of the model. (2) Implement automatic update mechanisms to retrain models or adjust model parameters based on the latest features of network traffic, thereby better accommodating the dynamic network environment.

Declaration of Competing Interest

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

Credit authorship contribution statement

Saihua Cai: Investigation, Methodology, Validation, Writing - original draft, Writing - review & editing, Funding acquisition. **Yang Zhang:** Investigation, Methodology, Validation, Writing - original draft, Writing - review & editing. **Yanghang Li:** Data curation, Validation, Writing - review & editing. **Yupeng Wang:** Data curation, Writing - review & editing. **Jiayao Li:** Data curation, Writing - review & editing. **Xiang Zhou:** Methodology, Writing - original draft, Writing - review & editing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) (Grant no. 62202206), the China Postdoctoral Science Foundation (Grant no. 2023T160275), the Natural Science Foundation of Jiangsu Province (Grant no. BK20220515), and the College Student Innovation and Entrepreneurship Training Program (Grant no. 202410299191Y).

References

[1] M. Saied, S. Guirguis, and M. Madbouly, "Review of artificial intelligence for enhancing intrusion detection in the internet of things," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107231, 2024.

[2] S. Cai, H. Xu, M. Liu, Z. Chen, and G. Zhang, "A malicious network traffic detection model based on bidirectional temporal convolutional network with multi-head self-attention mechanism," *Computers & Security*, vol. 136, p. 103580, 2024.

[3] U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, "Unknown, Atypical and Polymorphic Network Intrusion Detection: A Systematic Survey," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 1190–1212, 2024.

[4] J. Chen, T. Lv, S. Cai, L. Song, and S. Yin, "A novel detection model for abnormal network traffic based on bidirectional temporal convolutional network," *Information and Software Technology*, vol. 157, p. 107166, 2023.

[5] B. Lampe and W. Meng, "A survey of deep learning-based intrusion detection in automotive applications," *Expert Systems with Applications*, vol. 221, p. 119771, 2023.

[6] S. Cai, H. Tang, J. Chen, Y. Hu, and W. Guo, "CDDA-MD: An efficient malicious traffic detection method based on concept drift detection and adaptation technique," *Computers & Security*, vol. 148, p. 104121, 2025.

[7] J. Chen, Y. Chen, S. Cai, S. Yin, L. Zhao, and Z. Zhang, "An optimized feature extraction algorithm for abnormal network traffic detection," *Future Generation Computer Systems*, vol. 149, pp. 330–342, 2023.

[8] E. D. Vugrin, J. Cruz, C. Reedy, T. Tarman, and A. Pinar, "Cyber threat modeling and validation: port scanning and detection," in *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, 2020.

[9] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Detection of zero-day attacks: An unsupervised port-based approach," *Computer Networks*, vol. 180, p. 107391, 2020.

[10] H. Wu, Z. Shao, F. Yang, G. Cheng, X. Hu, J. Ren, and W. Wang, "PD-CPS: A practical scheme for detecting covert port scans in high-speed networks," *Computer Networks*, vol. 231, p. 109825, 2023.

[11] M. Ying, "Slow Port Scanning Attack Detection Algorithm Based on Dynamic Time Window Mechanism," in *APWeb-WAIM 2022 International Workshops on Web and Big Data*, pp. 188–201, 2023.

[12] M. Guendouz and A. Amine, "A New Feature Selection Method Based on Dragonfly Algorithm for Android Malware Detection Using Machine Learning Techniques," *International Journal of Information Security and Privacy*, vol. 17, pp. 1–18, 2023.

[13] Z. Chen, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier," *Computer Networks*, vol. 250, p. 110576, 2024.

[14] M. A. Talukder, M. M. Islam, M. A. Uddin, K. F. Hasan, S. Sharmin, S. A. Alyami, and M. A. Moni, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, pp. 1–44, 2024.

[15] D. Suja Mary, L. Jaya Singh Dhas, A. Deepa, M. A. Chaurasia, and C. Jaspin Jeba Sheela, "Network intrusion detection: An optimized deep learning approach using big data analytics," *Expert Systems with Applications*, vol. 251, p. 123919, 2024.

[16] F. U. Islam, G. Liu, W. Liu, and Q. M. Haq, "A deep learning-based framework to identify and characterise heterogeneous secure network traffic," *IET Information Security*, vol. 17, no. 2, p. 294–308, 2022.

[17] R. Devendiran and A. V. Turukmane, "Dugat-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy," *Expert Systems with Applications*, vol. 245, p. 123027, 2024.

[18] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in IoT networks," *Information Sciences*, vol. 639, p. 119000, 2023.

[19] H. Nandanwar and R. Katarya, "Deep learning enabled intrusion detection system for Industrial IOT environment," *Expert Systems with Applications*, vol. 249, p. 123808, 2024.

[20] U. B. Clinton, N. Hoque, and K. Robindro Singh, "Classification of DDoS attack traffic on SDN network environment using deep learning," *Cybersecurity*, vol. 7, no. 23, pp. 1–28, 2024.

[21] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 16, pp. 1–23, 2024.

[22] H. Ren, Y. Tang, W. Dong, S. Ren, and L. Jiang, "DUEN: Dynamic ensemble handling class imbalance in network intrusion detection," *Expert Systems with Applications*, vol. 229, p. 120420, 2023.

[23] Z. Lin, T. D. Pike, M. M. Bailey, and N. D. Bastian, "A Hypergraph-Based Machine Learning Ensemble Network Intrusion Detection System," *IEEE Transactions on Systems Man Cybernetics-Systems*, vol. 54, no. 11, pp. 6911–6923, 2024.

[24] I. S. Crespo-Martínez, A. Campazas-Vega, Ángel Manuel Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez Aparicio, and C. Fernández-Llamas, "SQL injection attack detection in network flow data," *Computers & Security*, vol. 127, p. 103093, 2023.

[25] R. Verma and S. Chandra, "RepuTE: A soft voting ensemble learning framework for reputation-based attack detection in fog-IoT milieu," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105670, 2023.

[26] R. Alghamdi and M. Bellaiche, "An ensemble deep learning based IDS for IoT using Lambda architecture," *Cybersecurity*, vol. 6, no. 5, pp. 1–17, 2023.

[27] M. Al-Sharif and A. Bushnag, "Enhancing cloud security: A study on ensemble learning-based intrusion detection systems," *IET Communications*, vol. 18, no. 16, pp. 950–965, 2024.

[28] S. Cai, Y. Zhao, J. Lyu, S. Wang, Y. Hu, M. Cheng, and G. Zhang, "DDP-DAR: Network intrusion detection based on denoising diffusion probabilistic model and dual-attention residual network," *Neural Networks*, vol. 184, p. 107064, 2025.

[29] O. A. Fernando, H. Xiao, and J. Spring, "New Algorithms for the Detection of Malicious Traffic in 5G-MEC," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2023.

[30] C. Wei, G. Xie, and Z. Diao, "A lightweight deep learning framework for botnet detecting at the iot edge," *Computers & Security*, vol. 129, p. 103195, 2023.

[31] H. Yan, X. Li, W. Zhang, R. Wang, H. Li, X. Zhao, F. Li, and X. Lin, "Automatic Evasion of Machine Learning-Based Network Intrusion Detection Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 1, pp. 153–167, 2024.

[32] W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Information Security*, vol. 13, no. 2, pp. 109–116, 2019.

[33] H. M. Gowda, M. Adithya, S. G. Prasad, and S. Vinay, "Development of anti-phishing browser based on random forest and rule of extraction framework," *Cybersecurity*, vol. 20, no. 3, pp. 1–14, 2020.

[34] O. M. Almorabea, T. J. S. Khanzada, M. A. Aslam, F. A. Hendi, and A. M. Almorabea, "IoT Network-Based Intrusion Detection Framework: A Solution to Process Ping Floods Originating From Embedded Devices," *IEEE Access*, vol. 11, pp. 119118–119145, 2023.

[35] P. Zhao, Z. Fan, Z.-W. Cao, and X. Li, "Intrusion Detection Model Using Temporal Convolutional Network Blend Into Attention Mechanism," *International Journal of Information Security and Privacy*, vol. 16, pp. 1–20, 2022.

[36] X. Wang, J. Liu, and C. Zhang, "Network intrusion detection based on multi-domain data and ensemble-bidirectional LSTM," *EURASIP Journal on Information Security*, vol. 2023, no. 5, pp. 1–14, 2023.

[37] I. Mohammed Sayem, M. Islam Sayed, S. Saha, and A. Haque, "ENIDS: A Deep Learning-Based Ensemble Framework for Network Intrusion Detection Systems," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, pp. 5809–5825, 2024.

[38] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowledge-Based Systems*, vol. 279, p. 110941, 2023.

[39] A. M. Alsaffar, M. Nouri-Baygi, and H. M. Zolbanin, "Shielding networks: enhancing intrusion detection with hybrid feature selection and stack ensemble learning," *Journal of Big Data*, vol. 11, no. 1, p. 133, 2024.

[40] A. H. Janabi, T. Kanakis, and M. Johnson, "Convolutional Neural Network Based Algorithm for Early Warning Proactive System Security in Software Defined Networks," *IEEE Access*, vol. 10, pp. 14301–14310, 2022.