Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.062094

ARTICLE

A Two-Layer Network Intrusion Detection Method Incorporating LSTM and Stacking Ensemble Learning

Jun Wang[1,2], Chaoren Ge[1,2], Yihong Li[1,2], Huimin Zhao[1,2], Qiang Fu[1,2,*], Kerang Cao[1,2] and Hoekyung Jung[3,*]

[1]College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang, 110142, China
[2]Key Laboratory of Intelligent Technology for Chemical Process Industry of Liaoning Province, Shenyang, 110142, China
[3]Computer Engineering Department, Paichai University, Daejeon, 35345, Republic of Korea

*Corresponding Authors: Hoekyung Jung. Email: hkjung@pcu.ac.kr; Qiang Fu. Email: qiang.fu@outlook.com

Received: 10 December 2024; Accepted: 10 March 2025; Published: 19 May 2025

ABSTRACT: Network Intrusion Detection System (NIDS) detection of minority class attacks is always a difficult task when dealing with attacks in complex network environments. To improve the detection capability of minority-class attacks, this study proposes an intrusion detection method based on a two-layer structure. The first layer employs a CNN-BiLSTM model incorporating an attention mechanism to classify network traffic into normal traffic, majority class attacks, and merged minority class attacks. The second layer further segments the minority class attacks through Stacking ensemble learning. The datasets are selected from the generic network dataset CIC-IDS2017, NSL-KDD, and the industrial network dataset Mississippi Gas Pipeline dataset to enhance the generalization and practical applicability of the model. Experimental results show that the proposed model achieves an overall detection accuracy of 99%, 99%, and 95% on the CIC-IDS2017, NSL-KDD, and industrial network datasets, respectively. It also significantly outperforms traditional methods in terms of detection accuracy and recall rate for minority class attacks. Compared with the single-layer deep learning model, the two-layer structure effectively reduces the false alarm rate while improving the minority-class attack detection performance. The research in this paper not only improves the adaptability of NIDS to complex network environments but also provides a new solution for minority-class attack detection in industrial network security.

KEYWORDS: Two-layer architecture; minority class attack; stacking ensemble learning; network intrusion detection

1 Introduction

Network Intrusion Detection System is one of the important tools to ensure network security. It can effectively prevent attacks by analyzing network traffic characteristics to identify abnormal activities. With the rapid development of network technology, the forms of cyberattacks have become increasingly complex and diverse. Cybersecurity incidents can cause significant damage, especially in the industrial Internet of Things (IIoT) and industrial control systems (ICS). Consequently, developing NIDS models capable of efficiently detecting complex, particularly minority class attacks, has become a key challenge in current research.

In recent years, deep learning-based network traffic intrusion detection systems have achieved remarkable results on several common network datasets (e.g., CICIDS 2017). Common technical methods include combining convolutional neural networks (CNN) and long short-term memory networks (LSTM) to

simultaneously extract spatial and temporal features from traffic data. However, traditional deep learning models often face challenges in effectively recognizing minority-class attacks. This is because minority-class attack samples are scarce, making models less sensitive to detecting them. In addition, many existing methods still use a single structure and cannot deeply analyze minority class attacks. In industrial network environments, the attack characteristics are more hidden, and generalized models are difficult to migrate and apply directly.

The issue of data imbalance remains challenging to address in many existing studies. Some studies begin by addressing the issue of unbalanced samples. Cui et al. [1] proposed a method to use GMM to generate rare samples for the minority class based on WGAN, and the redundant samples for the majority class were under-sampled based on the clustering algorithm. Ding et al. [2] proposed a TACGAN model that added two loss functions to the generator to measure the loss of information between the real data and the generated data, and designed a data filtering module to eliminate possible noise in the generated data, thus increasing the realism of the generated data. Some studies also start with modeling, for example, Bedi et al. [3] proposed an I-SiamIDS model that constructed a fusion model integrating a gradient-boosted tree and a double neural network, aiming to filter abnormal traffic through hierarchical filtering. However, it is still insufficient to identify samples of minority categories. Ullah et al. [4] proposed an IDS-INT model using transformer-based transfer learning to learn network feature representations and feature interactions in unbalanced data.

In this paper, a two-layer network intrusion detection system model is proposed. The first layer combines a CNN, a bidirectional Long Short-Term Memory Network (BiLSTM), and an attention mechanism to extract global and key features of network traffic and initially distinguish between normal traffic, majority-class attacks, and minority-class attacks. The second layer employs a stacking ensemble learning method to further classify the minority-class attack samples output by the first layer, thereby improving the accuracy of minority-class attack detection. Besides the general network datasets CIC-IDS2017 and NSL-KDD, the Mississippi Gas Pipeline dataset is also introduced. This improves the model's adaptability and generalization across different network environments. The main contributions of this paper are:

1. An NIDS model with a two-layer structure is proposed, which combines CNN-BiLSTM, attention mechanism, and Stacking technology for multi-stage feature extraction and classification.
2. The industrial network dataset is added to the model validation to verify its recognition ability in an industrial network environment.
3. Aiming at the scarcity of minority class attack samples, a hierarchical classification architecture is adopted to significantly improve the detection capability of minority class attacks.

The rest of this paper is organized as follows. Section 2 describes related work on network intrusion detection systems in recent years, and also discusses the application and improvement of the PSO algorithm in the field of deep learning models. Section 3 describes the proposed model in detail. Section 4 analyses the experimental results. Section 5 summarizes the work in this paper.

2 Related Works

As cyberattacks continue to evolve, traditional rule-based intrusion detection methods are no longer sufficient to meet current security needs. As a result, researchers have turned to more effective solutions, such as machine learning and deep learning techniques, which are gradually becoming mainstream.

These emerging methods can automatically extract features from massive amounts of network data to more accurately identify potentially malicious activities. Compared with traditional rule-based approaches, they are more flexible and adaptable to deal with more complex and stealthy attack patterns. The advantages of these methods are particularly obvious when dealing with large-scale network traffic.

Several machine learning algorithms have been applied to network intrusion detection systems. Ambusaidi et al. [5] proposed an improved random forest algorithm and compared its performance with nine well-known ML algorithms to obtain good results. Training data for NIDS must be explicitly labeled as normal or malicious traffic, but relevant data is scarce. To address this phenomenon, Apruzzese et al. [6] proposed the XeNIDS framework, which can be used for cross-validation evaluation for machine learning.

In addition, due to the increase in network data traffic and the increase in the dimensionality of data characteristics, it has become very difficult to improve the prediction accuracy of individual classifiers. Therefore, the development of more complex and deeper network structures has become a trend. Saheed et al. [7] proposed an integrated model using a Gray Wolf Optimizer (GWO). The model employed a voting technique to combine the probability averages of the basic learners and used GWO to optimize the parameters of the comprehensive model. Good detection results were achieved on the BoT-IoT industrial network dataset. Okey et al. [8] proposed a BoostedEnsML model. They first integrated six different machine learning models using Stacking and majority voting and then selected the two models with the best results to form the final model. Experiments demonstrate that BoostedEnsML exhibits performance advantages over existing integrated models.

The field of deep learning network intrusion detection provides a promising approach. Bakhsh et al. [9] constructed a deep learning-based intrusion detection system using a feedforward neural network (FFNN), LSTM, and a random neural network (RandNN), which demonstrated excellent performance in the CIC-IoT22 test. Sai Chaitanya Kumar et al. [10] proposed a Deep Residual Convolutional Neural Network (DCRNN) and fine-tuned it using the Improved Gazelle Optimization Algorithm (IGOA). Serrano et al. [11] proposed a CyberAIBot model that assigned a set of deep learning models to normal traffic and each type of attack traffic and learned specific traces from previous network attacks. The model detected and classified current attacks in real time, and assigned a new set of deep learning models to new attack categories as they were discovered. Detecting a new attack category does not affect the clusters of existing attack categories.

Meanwhile, stratifying intrusion traffic is also a major research hotspot. Luo et al. [12] proposed a two-layer intrusion detection model, the first layer used the traditional rule-based intrusion detection module, and the traffic that passed the rule then entered the second layer of the AI detection module (consisting of a two-layer GRU) for detection. The traffic is categorized as normal only if both detection modules judge it as normal. Harini et al. [13] proposed a three-layer detection model. The first layer used a weighted deep neural network to filter out attack traffic, the second layer employed CNN + LSTM for secondary classification, and a small number of classified attacks were merged into the third layer for tertiary classification using Boosting. This provides an idea of combining deep learning with machine learning, but the model is complex and the minority classes are likely to be filtered out in the first two layers. Wisanwanichthan et al. [14] proposed a DLHA model that tried to solve the problem that it is difficult for a single machine learning classifier to accurately identify all types of attacks. The first layer of the DLHA model employed a plain Bayesian classifier to detect DoS and Probe, and the second layer deployed Support Vector Machines to achieve the detection of R2L and U2R. The model can detect U2R (a minority class of attacks that can be easily misclassified) very well, but the model is not sufficiently scalable and is limited.

With the increase of high dimensional data, feature selection becomes an important step in data processing. Optimization algorithms, such as PSO and genetic algorithms, are effective in feature selection of high-dimensional data [15]. In network intrusion detection, PSO is mainly used for feature selection and model parameter optimization. Traditional PSO suffers from the problem that it is easy to fall into the local optimal solution. Common improvement directions for PSO algorithms are: population initialization, population diversity, and correlation information between features. The following section introduces some variants of PSO.

Jin et al. [16] proposed an adaptive pyramid particle swarm optimization algorithm, which divided the population into pyramid structures with different levels. The individuals at higher levels instructed those at lower levels to learn, thus improving the global search ability. Particles at different levels competed with each other, and the particle that lost the competition learned from the winning particle. In addition, a feature dynamic flipping strategy was added to enhance the population development. However, these steps led to a significant increase in algorithmic time while providing better performance. Pramanik et al. [17] proposed an AAPSO algorithm for feature extraction from a pneumonia chest X-ray dataset. The algorithm introduced a memory-based adaptive parameter and incorporated an altruistic behavior, where particles with good performance actively help those with poor performance by transferring a portion of their data. Although the algorithm was originally designed for medical image analysis, it is considered equally applicable to high-dimensional feature selection in network intrusion detection. The APSO algorithm applied in this paper incorporates improvements based on this algorithm. Meanwhile, due to the high feature complexity of the industrial network dataset, the process of passing on altruistic behaviors was simplified, and the fitness evaluation function was reduced to classification accuracy to improve optimization efficiency. Song et al. [18] proposed a feature selection algorithm based on BBPSO (Bare Bones Particle Swarm Optimization). The method uses label correlation to initialize the swarm, incorporates operators to refine feature subsets, and applies adaptive mutation to avoid local optima. This approach demonstrates strong performance in high-dimensional feature selection tasks. Due to space limitations, it is not possible to list all relevant literature, and only a brief introduction to PSO improvements is given.

Hierarchical traffic processing is regarded as an effective strategy; however, existing approaches still face several challenges. Some rely on a single machine learning or deep learning method, limiting their ability to fully utilize the advantages of both. Others combine machine learning and deep learning techniques, but suffer from complex models, limited datasets, and poor generalization, making them unsuitable for complex industrial network environments.

3 Proposed Model

The dataset contains a small number of minority attack samples, leading to significant class imbalance relative to other attack categories and normal traffic. Techniques like over-sampling and under-sampling cannot fully mitigate this imbalance. Furthermore, the limited number of samples results in suboptimal deep learning performance. Traditional machine learning methods often outperform deep learning with small sample sizes.

This section introduces the proposed two-layer intrusion detection model, which integrates a bidirectional long short-term memory network, an attention mechanism, and Stacking ensemble learning. In addition, an altruism-based Adaptive Particle Swarm Optimization (APSO) algorithm is introduced for feature selection and optimization of the model.

3.1 Two-Layer Intrusion Detection Model

The proposed model is a two-layer architecture. The main task of the first layer is to filter out normal traffic and majority attack categories and merge majority attack categories into one; the second layer focuses on the detailed classification of the merged minority class attack.

Algorithm 1 shows the steps of the proposed two-layer intrusion detection model.

Algorithm 1: Proposed two-layer intrusion detection model

```
Input: Dataset,
Process:
Data preprocessing (e.g., feature selection, normalization, one-hot encoding);
Split the dataset as Training_set, test_set
Training first layer Model(model1):
For train_index, valid_index in StratifiedKFold(Training_set)
    train_X, valid_X ← Split the feature data based on the index
    train_y, valid_y ← Split the target data based on the index
    train_X_over, valid_X_over ← Use SMOTE-ENN for oversampling and cleaning
model1.train(train_X_over, train_y, validation_data = (valid_X_over, valid_y))
End
Training second layer model(model2)
Evaluate the first layer model:
    y_pred_layer1 ← model1.predict(Test_set)
    y_non_minority_pred, y_minority_pred ← split the y_pred_layer1
Evaluate the second layer model:
    y_pred_layer2 ← model2.predict(y_minority_pred)
    y_pred_total ← merge y_pred_layer1 and y_pred_layer2
Output: The prediction results of the entire model
```

The structural design of these two layers is described in Fig. 1.

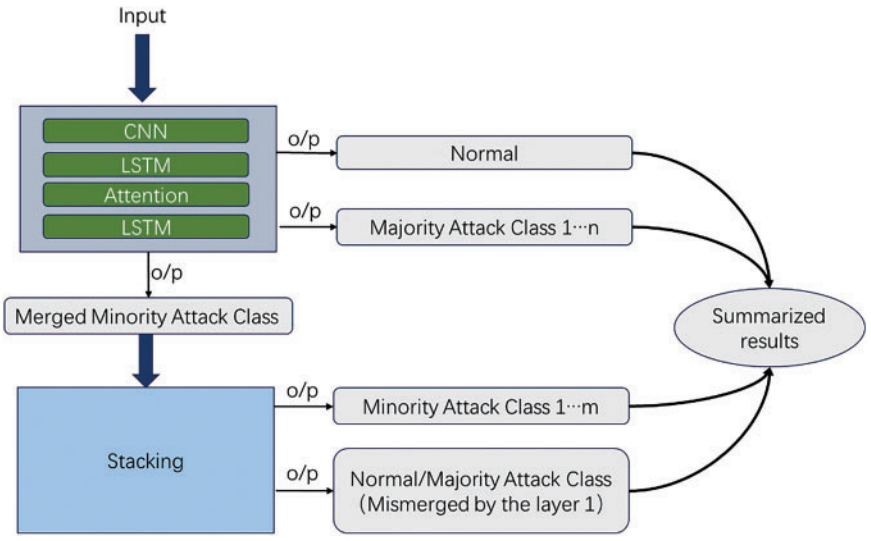


Figure 1: Two-layer model architecture diagram

3.1.1 Layer1: CNN + BiLSTM + Attention

The first layer combines CNN, BiLSTM, and Attention Mechanisms. The CNN part is responsible for extracting spatial features from the input data to capture local patterns in the network traffic. Then, the

BiLSTM part processes the data and learns the temporal dependencies to capture both forward and reverse timing information for a more comprehensive understanding of the timing characteristics in the data.

BiLSTM is an improved recurrent neural network based on the traditional LSTM, which is capable of capturing both forward and backward dependencies in sequential data, thus improving the understanding of contextual information. Unlike unidirectional LSTM, BiLSTM performs information extraction from the forward and backward directions of the input sequence by introducing two independent LSTM layers and combining the outputs of the two parts, which are used to capture more comprehensive features.

An attention mechanism was inserted between the BiLSTM layers, as formulated below:

Given an input sequence $X = \{x_1, x_2, \ldots, x_T\}$, where $x_t \in R^d$ denotes the feature vector at the t-th time step. The computational process of the attention mechanism can be represented as the following steps:

1. Compute the hidden layer representation

Hidden layer representation u_t for each time step:

$$u_t = tanh\ (Wx_t + b) \tag{1}$$

where $W \in R^{d \times d}$ is the learnable weight matrix, $b \in R^d$ is the bias vector, and *tanh* is the activation function to introduce nonlinearity.

2. Calculate the attention weights

Calculate the attentional weight α_t using a context vector $u \in R^d$ to measure the importance of each time step:

$$e_t = u^\top u_t \tag{2}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{3}$$

where e_t is the attention score and α_t is the normalized attention weight that satisfies the $\sum_{t=1}^{T} \alpha_t = 1$.

3. Compute the weighted representation of the attention mechanism

The input sequence is weighted and summed using the attention weights to obtain the output vector v

$$v = \sum_{t=1}^{T} \alpha_t x_t \tag{4}$$

where $v \in R^d$ is the weighted representation of the input sequence

The attention mechanism, when integrated into the BiLSTM layer, weights the time series of the first layer LSTM output to focus on critical time steps. The first layer LSTM output is $H = \{h_1, h_2, \ldots, h_T\}$, where $h_t \in R^d$ is the hidden state of the first layer LSTM at the $t-$th time step. The weighted representation is then generated through the attention mechanism: $v = \sum_{t=1}^{T} \alpha_t h_t$. Finally, the weighted representation is passed to the second LSTM layer. This approach can help the second LSTM layer to be able to focus more on the key features and improve the model's ability to recognize the minority class attacks. Moreover, the attention mechanism reduces the time steps of the first LSTM layer to a single feature vector, which also reduces the computational complexity of the subsequent layers.

In this way, the first layer can extract effective features from the network traffic data and perform an initial categorization to classify the traffic into normal traffic, majority class attacks, and a merged minority class of attacks.

3.1.2 Layer2: Stacking Ensemble Learning

The second layer uses a stacking ensemble learning approach, where the Stacking model contains multiple base classifiers that can efficiently handle complex and high-dimensional data. The structure of the stacking ensemble learning is shown in Fig. 2. Combining the outputs of multiple base learners, the model can effectively perform minority class attack classification. In this layer, the model can effectively utilize the advantage of machine learning in the case of a small sample size to segment the merged minority class attacks delivered by the first layer, further improving the detection of minority class attacks.

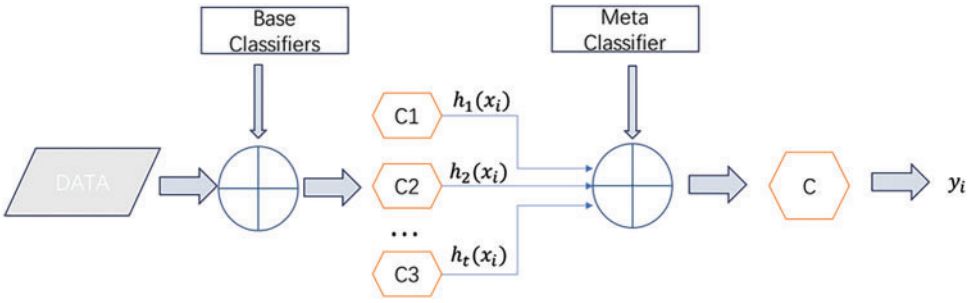


Figure 2: Structure diagram of stacking ensemble learning

The output of the second layer contains not only subdivided minority class attacks but also some normal or majority attack classes. Ideally, the first layer should accurately classify all normal and most attack traffic, leaving only minority attacks in the merged outputs passed to the second layer. However, in practice, the classification process at the first layer may produce some errors and mistakenly pass some non-minority attacks to the second layer. Therefore, in the second layer, a secondary classification is performed on the misclassified traffic from the first layer, further enhancing the identification of suspicious traffic.

Algorithm 2 shows the steps of stacking ensemble learning.

Algorithm 2: Stacking ensemble learning

Input: Data set $\mathcal{D} = \{x_i, y_i\}_{i=1}^{m}$;
Base Classifier $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_T$;
Meta Classifier $\mathcal{C}$.
Process:
For $t = 1, \ldots, T$
$\boldsymbol{h_t} = \mathcal{C}_t(\mathcal{D})$ %Trained using Base Classifier $\mathcal{C}_t$
End;
Construct a new dataset from $\mathcal{D}$ which is named as $\mathcal{D}'$
For $i = 1,\ldots,m$:
For $t = 1, \ldots, T$:
$z_{it} = h_t(x_i)$ %Classify the sample x_i using Base Classifier $\mathcal{C}_t$
End;
$\mathcal{D}' = D'\mathrm{U}((\mathrm{z_{i1}}, \mathrm{z_{i2}}, \ldots, \mathrm{z_{it}}), \mathrm{y_i})$
End;
$h' = \mathcal{C}(\mathcal{D}')$ %Trained using Meta Classifier $\mathcal{C}$
Output: $\mathbf{H(x)} = h'(\boldsymbol{h_1}(\mathbf{x}), \boldsymbol{h_2}(\mathbf{x}), .., \boldsymbol{h_t}(\mathbf{x}))$

3.2 APSO

In this paper, APSO is mainly used for feature selection. Through feature selection, APSO can help eliminate redundant and irrelevant features, reduce computational overhead, and thus improve the training speed of the model. In addition, APSO's feature selection process can effectively improve the model's sensitivity to important features and ensure that the model focuses on the most informative features, thus optimizing the model's classification performance.

3.2.1 Motivation

In the traditional PSO algorithm, the particle swarm often converges too quickly in the search space, which increases the risk of falling into a local optimum. In nature, certain individuals make sacrifices for the benefit of the group. This altruistic behavior is introduced into PSO. The particles with better performance help the particles with poorer performance, thereby improving the searchability of the entire group.

3.2.2 The Concrete Realization

Altruistic behavior is implemented in the following two steps. First, retaining the optimal agents. In each round of iteration, the fitness of each particle is calculated and ranked. The best-performing group of elite particles is retained first. These particles will not participate in the altruistic process, they will retain their positions and continue to converge to the optimal solution. The second step is pairing, the remaining non-elite particles are divided into two halves according to their performance, i.e., "better-performing particles" and "worse-performing particles". Each "better-performing particle" is paired with a "worse-performing particle". During the pairing process, the better particles randomly pass some of their features and speed to the worse particles to help them improve their search efficiency. The flowchart of the APSO algorithm is shown in Fig. 3.

During the pairing process, not all information is simply transferred. The velocities and properties of the 'better particles' are determined based on Eq. (5) to decide whether they should be transferred to the 'worse particles'. Where $rand(x)$ is a random number function that generates a random number in the interval $[0, 1]$. a is an editable parameter that can be adjusted to regulate the efficiency of information transfer in the pairing process by adjusting the size of a. In addition, the fitness is set to be the classification accuracy.

$$f(x) = \begin{cases} 1, & rand(x) < \alpha \\ 0, & rand(x) \geq \alpha \end{cases} \tag{5}$$

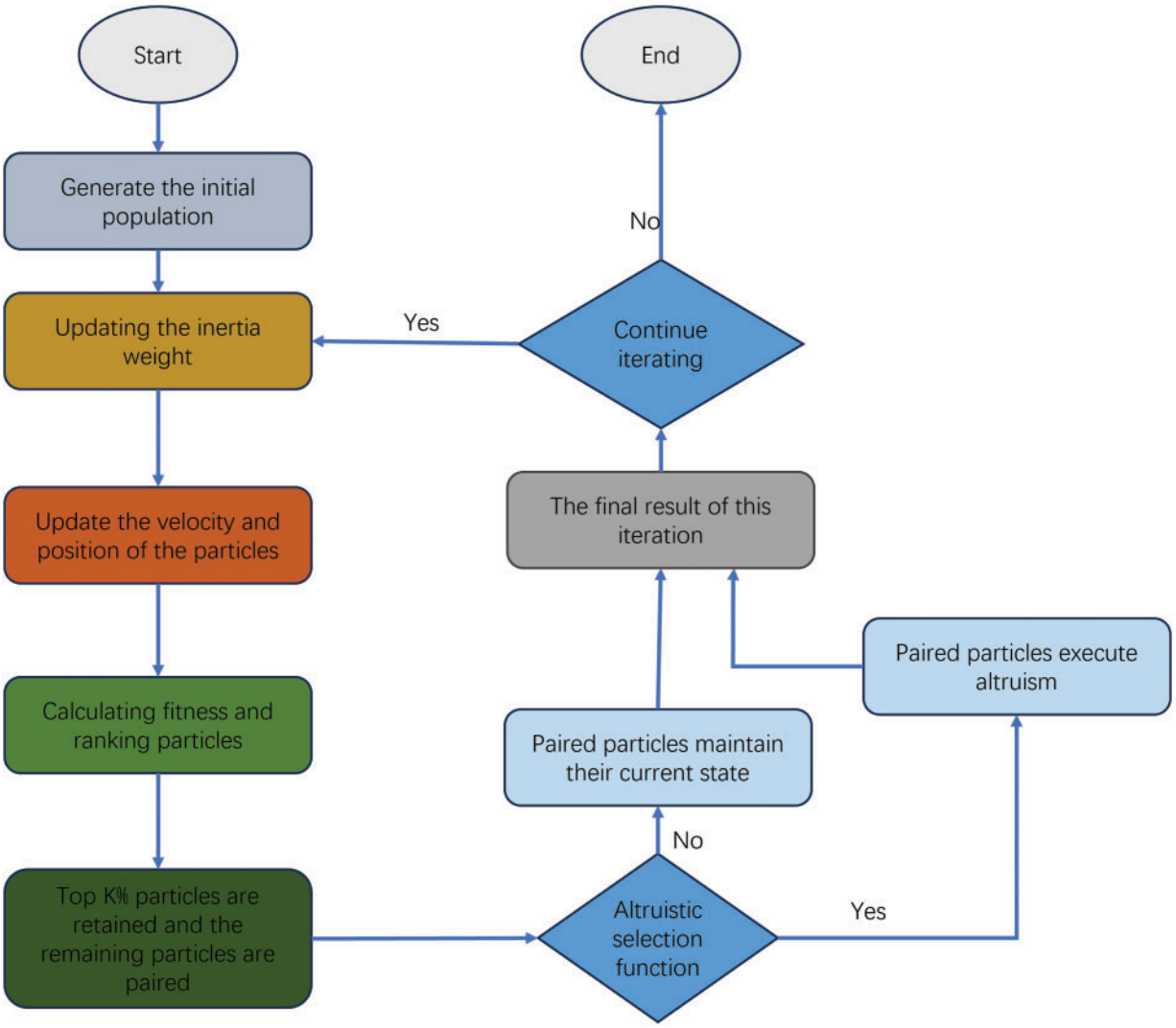


Figure 3: Flowchart of the APSO algorithm

Algorithm 3 shows the pseudocode description of the APSO:

Algorithm 3: APSO

Input: Dataset: N particles with having a solution set S, each with the dimension d, having corresponding velocities v. k fraction of solutions to be considered as elite, Maximum number of iterations T,

Process:

Initialize the population:

For $t = 1$ to T:

    Update the inertia weight, velocity and position of the particles;

    Perform PSO;

    Sort particles by fitness value in descending order;

    start_particles $\leftarrow k\% \times N$

    stop_particles $\leftarrow N - k\% \times N$

(Continued)

Algorithm 3 (continued)

```
        For i in start_particles to stop_particles:
            m ← N − i
                For j ← 1 1 to d:
                    if random (0, 1) < a then
                        S_mj ← S_ij
                        v_mj ← v_ij
                End
        End
    End
    Output: The optimal feature subset
```

In addition, the inertia weight coefficients are updated in a linearly decreasing manner with the Eq. (6)

$$w_k = w_{\max} - \frac{k}{K} \cdot (w_{\max} - w_{\min}) \tag{6}$$

where w_k is the inertia weight at the k-th iteration, $w_{\max}$ is the initial inertia weight, $w_{\min}$ is the minimum inertia weight, k is the number of current iterations (from k to K), and K is the maximum number of iterations. In general, the algorithm performs best when $w_{\max} = 0.9$, $w_{\min} = 0.4$.

4 Experiment and Result Analysis

For hyperparameter selection in the first layer of deep learning, experiments were conducted within the following ranges:

Number of Convolutional Layer Filters (Filters): We tested 32, 64, and 128 filters to balance the model's feature extraction capability and computational complexity.

LSTM Units: We experimented with 32, 64, and 128 units and selected 64 and 128 to balance capturing temporal dependencies and avoiding overfitting.

Dropout Rate: A dropout rate of 0.6 was selected to prevent overfitting while maintaining sufficient model capacity for learning.

Some key parameters are given in Table 1.

Table 1: Explanation of some important parameters of the model

Layer	Parameters
Conv1D	Filters = 64, kernel_size = 3, activation = "relu"
MaxPooling1D	pool_size = 2
BatchNormalization	
LSTM	Units = 64
Attention	
Reshape	target_shape = (128,1)
MaxPooling1D	pool_size = 2
BatchNormalization	
LSTM	units = 128
Dropout	rate = 0.6

In the second layer of the stacked ensemble model, we chose Random Forest and ExtraTrees as base classifiers, with Random Forest as the meta-classifier. All three models are tree-based. Through stacking, their robustness is effectively enhanced. This approach does not significantly increase computational complexity, thus striking a balance between efficiency and effectiveness.

We applied the EarlyStopping technique during the training process. To enhance model stability during training, we incorporated parameters for recovering the model's optimal weights into EarlyStopping.

4.1 Datasets and Data Preprocessing

Three network traffic datasets were used for this experiment: the CIC-IDS2017, NSL-KDD, and the Mississippi Gas Pipeline datasets.

The NSL-KDD dataset is an improved version of the KDD CUP 1999 dataset, which solves some problems in the original dataset by removing redundant data and adjusting the number of records, making the experimental results more reliable. Its training set is KDDTrain+ and the test set is KDDTest+. The dataset includes normal traffic and four major types of attack traffic, which are: denial-of-service attacks (DOS), illegal access by normal users to local super-spam user privileges (U2R), illegal access from remote machines (R2L), and port monitoring or scanning (Probe). The sample distribution of the dataset is shown in Table 2.

Table 2: The description of the NSL-KDD dataset

Class	KDDTrain+ Num	KDDTrain+ Proportion	KDDTest+ Num	KDDTrain+ Proportion
Normal	67,343	53.45	9711	43.07
Dos	45,927	36.45	7460	33.09
Probe	11,656	9.25	2421	10.73
R2L	995	0.007	2885	12.80
U2R	52	0.0004	67	0.0002

Since its development, the CICIDS-2017 dataset [19] has attracted a great deal of attention from many researchers and students. Based on the analysis and study of this dataset, many new intrusion detection models have been proposed. The dataset, from the Canadian Institute for Cyber Security Research, consists of eight different files containing five days of normal and attack traffic data.

This dataset is extensive, with a large amount of data. However, it contains noise, default values, and overly detailed segmentation of attacks within the same class, which leads the model to spend most of its training time learning to detect additional classes.

To address these issues, we refer to methods proposed by Goryunov et al. [20], Kurniabudi et al. [21], Salo et al. [22], and others. These methods form a new attack class by merging the minority classes of the same broad class and removing some of the majority classes. This is the commonly used processing method. The preprocessed dataset features are called CICIDS2017_sample, and detailed information about the features and content of the new dataset is provided in Table 3.

The Mississippi Gas Pipeline dataset [23] belongs to the industrial control system (ICS) domain. It contains simulated attack samples, such as disrupting control signals, forging packets, and other behaviors, providing a realistic validation basis for applying the model in industrial network scenarios. The specific attack categories are given in Table 4.

Table 3: The description of CICIDS-2017 dataset

Serial number	Labels	Number of instances
0	BENIGN	22,767
1	DoS	19,035
2	PortScan	7946
3	BruteForce	2767
4	WebAttack	2180
5	Bot	1966

Table 4: The description of Gas Pipeline dataset

Serial number	Class
0	Normal
1	NMRI
2	CMRI
3	MSCI
4	MPCI
5	MFCI
6	DOS

SMOTE-ENN (Synthetic Minority Over-sampling Technique-Edited Nearest Neighbors) is an algorithm that combines over-sampling and under-sampling techniques to address the class imbalance issue. In the first stage, the SMOTE technique is applied to generate synthetic samples of the minority classes, thereby increasing their number. In the second stage, the ENN technique is applied to the dataset after SMOTE oversampling to remove noise and redundant samples. Samples lacking a clear class in their neighborhood are removed by editing the neighbors of each sample.

SMOTE (Synthetic Minority Over-sampling Technique) is an oversampling method used to address class imbalance issues. Specifically, the SMOTE algorithm randomly selects a sample from the minority class and computes its K nearest neighbors. It then interpolates a point between the original sample and its neighbors to generate a new sample. This process smooths the decision boundary, enabling the model to better recognize the minority class.

ENN (Edited Nearest Neighbors) is an under-sampling technique that enhances dataset quality by removing noisy and redundant samples. It works by examining each sample's K nearest neighbors; if the sample's category differs from most of its neighbors, it is deemed noisy and removed. ENN removes samples with unclear boundaries or misclassifications, thereby cleaning the dataset and reducing noisy interference during training. This improves the model's accuracy and generalization ability. ENN is especially effective when the dataset contains many mislabeled or borderline samples. However, it may also remove useful samples, particularly when class boundaries are ambiguous.

SMOTE-ENN is effective for both binary and multiclass classification problems with class imbalance, as it improves minority class recognition and reduces noise influence.

4.2 Evaluation Metrics

To comprehensively assess the model's classification performance, this paper employs a variety of performance evaluation metrics. *Accuracy* reflects the overall classification correctness of the model across all samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

The *precision* measures the model's prediction success, i.e., the proportion of samples that the model successfully predicts out of all the samples it predicts.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall reflects the model's ability to cover the target class, i.e., the proportion of samples that the model can correctly detect from the actual sample. A higher recall indicates that the model misses fewer samples:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

The $F1$ value balances both *precision* and *recall* and is suitable for the classification evaluation of unbalanced data. In minority class detection, a higher $F1$ value indicates that the model achieves a better trade-off between *precision* and *recall*:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{10}$$

The ROC curve is a graphical tool for evaluating classification model performance. It shows the model's ability to discriminate between categories by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at different thresholds. The AUC value indicates the overall ability of the classifier to distinguish between positive and negative classes.

$$TPR = \frac{TP}{TP + FN} \tag{11}$$

$$FPR = \frac{FP}{FP + TN} \tag{12}$$

4.3 Evaluation Using the CIC-IDS2017 Dataset

To enhance the generalization ability of the model, cross-validation techniques were applied during the experiment. A preliminary experiment was conducted to determine the optimal number of folds for cross-validation. Additionally, the effectiveness of combination sampling techniques was tested. The experimental results are shown in Fig. 4.

As shown in Fig. 4, when the SMOTE-ENN technique is not applied, increasing the number of folds helps the model better recognize minority class samples, thereby improving performance. When the SMOTE-ENN technique is applied, the model's performance with SMOTE-ENN is higher than without it for the same number of folds, as the class imbalance issue in the training samples is partially addressed.

The model's performance at different epochs was then evaluated, and the results are shown in Figs. 5 and 6.

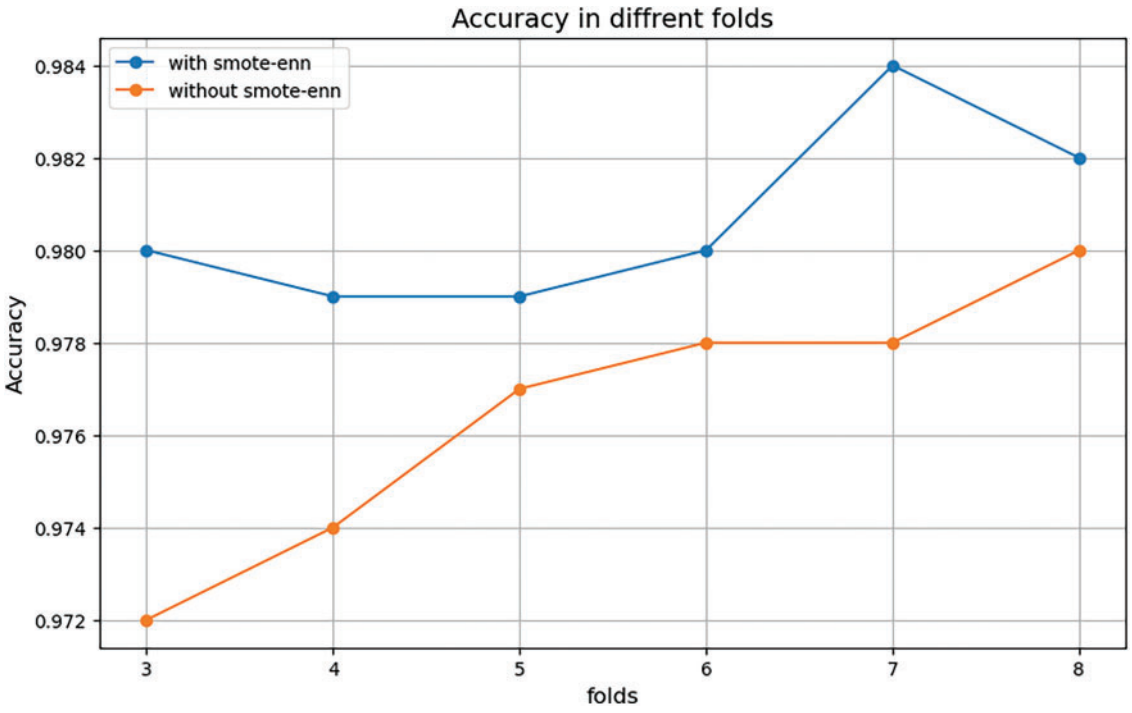


Figure 4: Accuracy in different folds

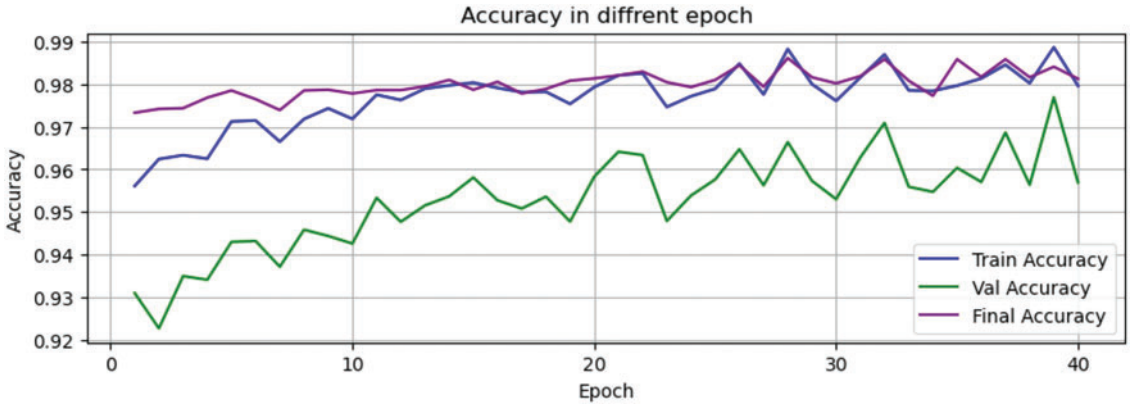


Figure 5: Accuracy in a different epoch

The Final Accuracy in Fig. 5 is explained here: Because the model uses a hierarchical structure, whether the first layer can correctly identify the merged minority attacks will largely affect the overall performance of the final model. The first-layer deep learning model will yield a Validation Accuracy, while the overall performance of the total model is reflected in Final Accuracy.

The overall performance of the model on the CIC-IDS2017 dataset is shown in Table 5, with an overall accuracy of 99% on the test dataset, in addition to a recall above 97% and an F1 value above 98% for each category. This indicates that the model proposed in this paper demonstrates excellent overall performance while maintaining strong recognition performance across all categories.

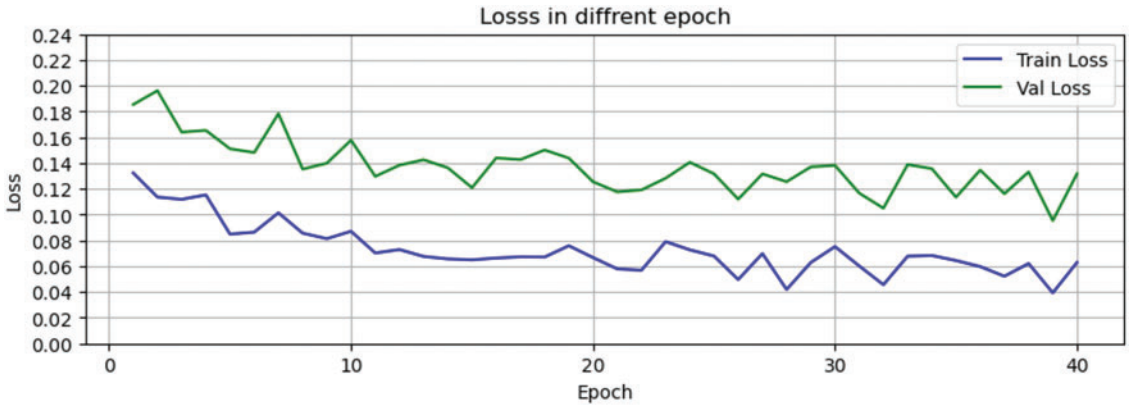


Figure 6: Loss in different epochs

Table 5: The precision, recall, F1-score of every kind of data in CIC-IDS2017 dataset

Class	Precision	Recall	F1
BENIGN	0.99	0.98	0.99
Bot	1.00	0.97	0.98
BruteForce	1.00	0.99	1.00
DoS	0.99	1.00	0.99
PortScan	0.96	1.00	0.98
WebAttack	1.00	0.98	0.99

To further assess the model's predictive performance across different categories, a visualization of the confusion matrix is provided below:

As shown in Fig. 7, the vast majority of the samples were correctly categorized. However, some normal traffic was misclassified as malicious traffic. This may be because some normal traffic shares characteristics similar to attack traffic, especially DoS and PortScan, which exhibit high traffic or repeated connection patterns. Overall, the model demonstrates a high level of reliability, as shown in Fig. 8.

To better verify the validity of the proposed model, the experimental results were compared with those of the models in other research papers.

As can be seen from Table 6, the proposed model achieves a score of 0.99 in Precision, Recall, and F1-Score, outperforming other compared methods or matching the optimal method:

In terms of Precision and F1 metrics, it is substantially ahead of the traditional MLP method. Compared with the CNN-BiLSTM with a similar structure, the proposed model is slightly improved in all the metrics through the enhancement of the Attention mechanism and the optimization of the two-layer structure. Although OC-SVM/RF performs similarly to the proposed model on the CICIDS2017 dataset, its three-layer structure makes the model highly dependent on the thresholds set at each stage. Small changes in these thresholds can significantly affect detection results. Determining the optimal thresholds typically requires numerous experiments and validations. Additionally, the OC-SVM/RF model has not been tested on diverse datasets and lacks sufficient validation of its generalizability. The characteristics of different datasets may also significantly influence the selection of optimal thresholds.

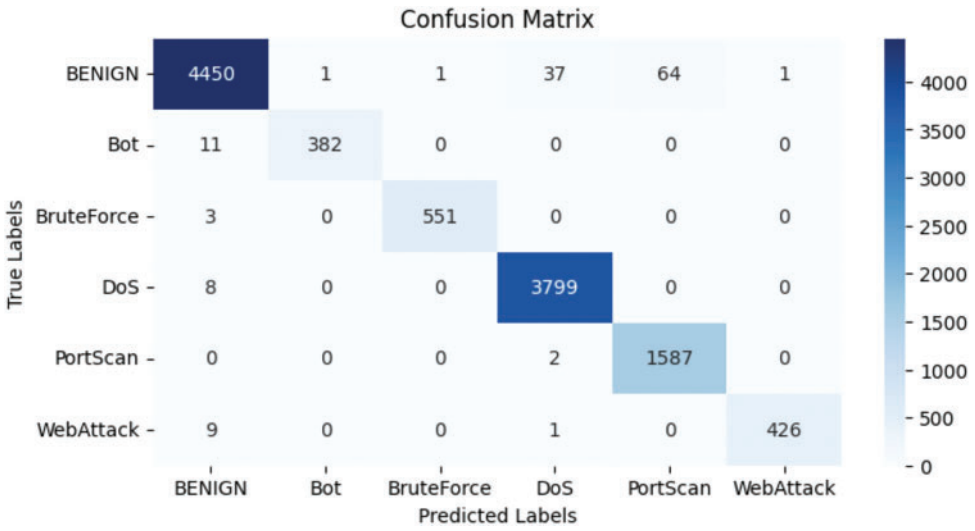


Figure 7: Confusion matrix for multi-class classification on the CICIDS-2017 dataset

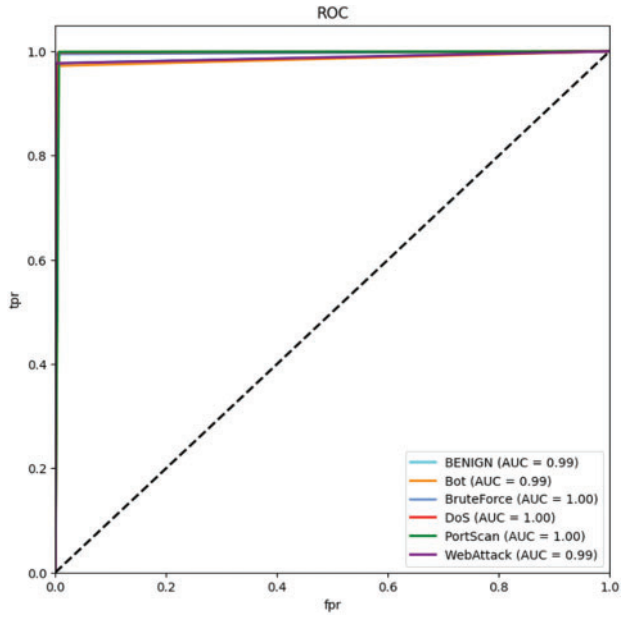


Figure 8: ROC curve for multi-class classification on natural CICIDS-2017 dataset

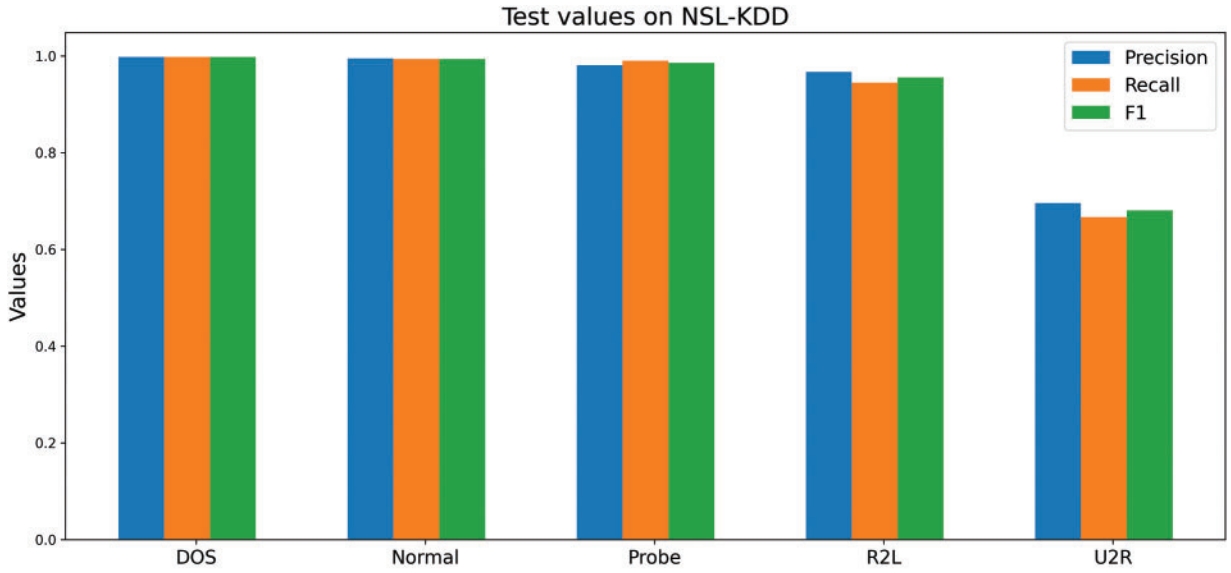
Table 6: The comparison with other method on CICIDS2017

Model	Precision	Recall	F1
Proposed method	0.99	0.99	0.99
MLP [24]	0.88	0.99	0.87
HMCD-Model [25]	0.87	0.95	0.90
CNN-BiLSTM [26]	0.98	0.98	0.98
OC-SVM/RF [27]	0.99	0.98	0.99

In summary, the proposed model not only effectively improves the overall detection performance of network attacks through the combination of Attention mechanism and stacking ensemble learning, but also shows stronger robustness and generalization ability in the detection of a few types of attacks. The effectiveness and practicality of the proposed model have been successfully verified.

4.4 Evaluation Using the NSL-KDD Dataset

To evaluate performance in a generalized network environment, the NSL-KDD dataset was also tested. As can be seen from Fig. 9, excellent classification results were obtained for all categories except for the U2R category where the results were not satisfactory.



Class	Precision	Recall	F1
DOS	0.998	0.998	0.998
Normal	0.995	0.994	0.994
Probe	0.981	0.99	0.986
R2L	0.967	0.945	0.956
U2R	0.696	0.667	0.681

Figure 9: Histogram for multi-class classification on natural NSL-KDD dataset

To further explore the low classification performance of U2R, the dataset and training process were analyzed, and it was found that U2R belongs to very few categories. The distribution of data within the NSL-KDD dataset is illustrated in Fig. 10. In the NSL-KDD dataset, the total number of U2R samples is only 119, accounting for only 0.08% of the entire data set. Both the sample size and the distribution of traffic types suggest that U2R is difficult to classify. Referring to other literature, the classification effect of U2R is not ideal. Excluding U2R, the overall classification performance of the model remains excellent, with accuracy, recall, and F1 scores for the entire dataset reaching 99.4%. This is also supported by the confusion matrix presented in Fig. 11 and the ROC plot presented in Fig. 12.

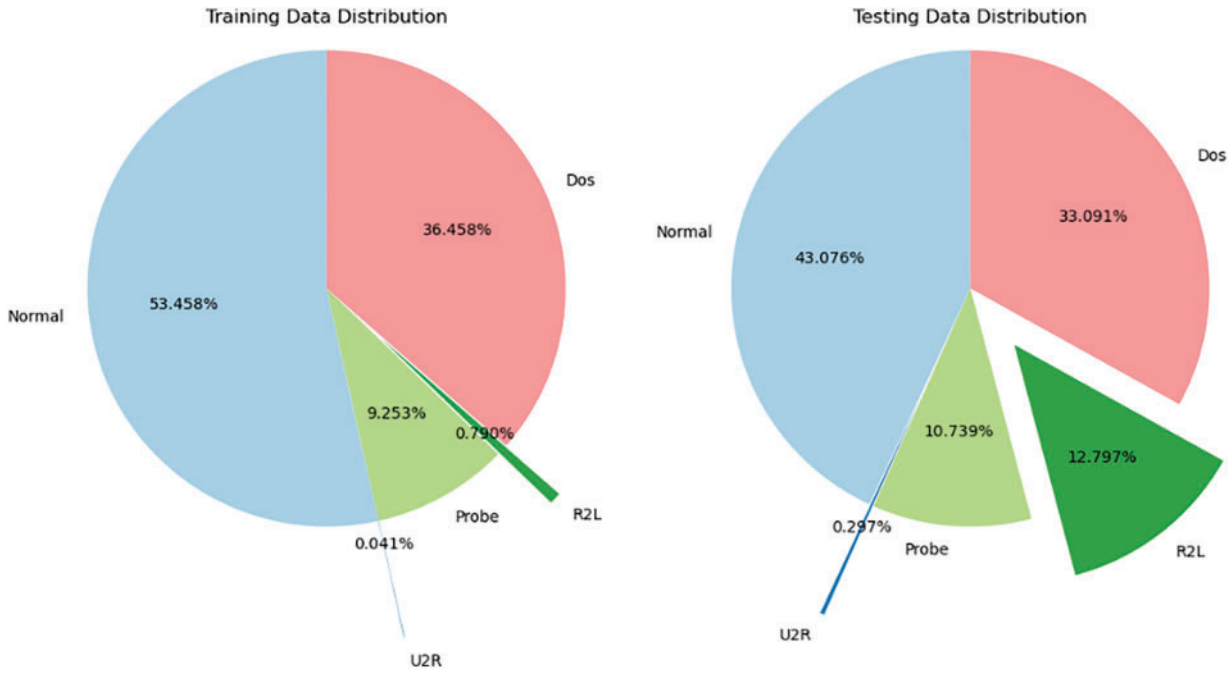


Figure 10: Data distribution on the NSL-KDD dataset

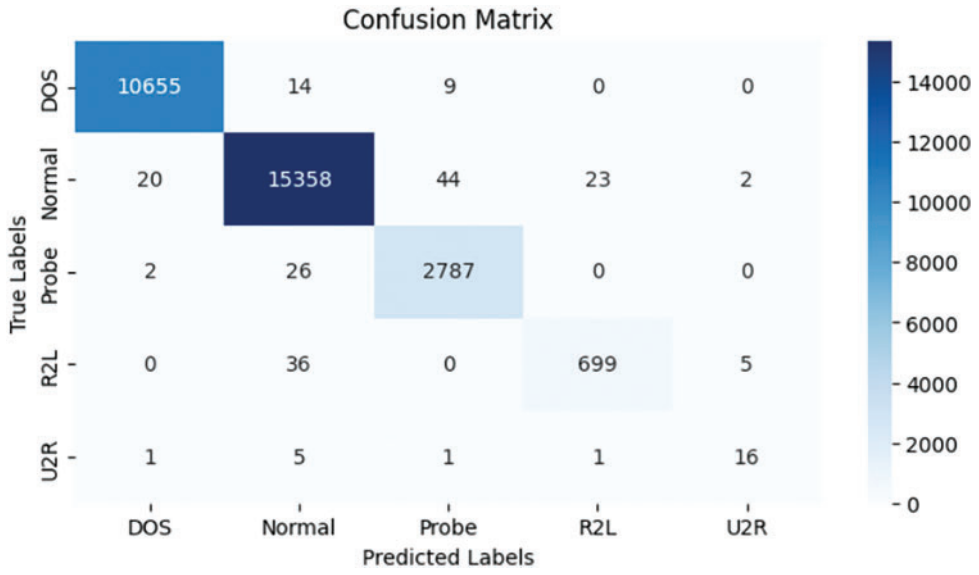


Figure 11: Confusion matrix for multi-class classification on NSL-KDD dataset

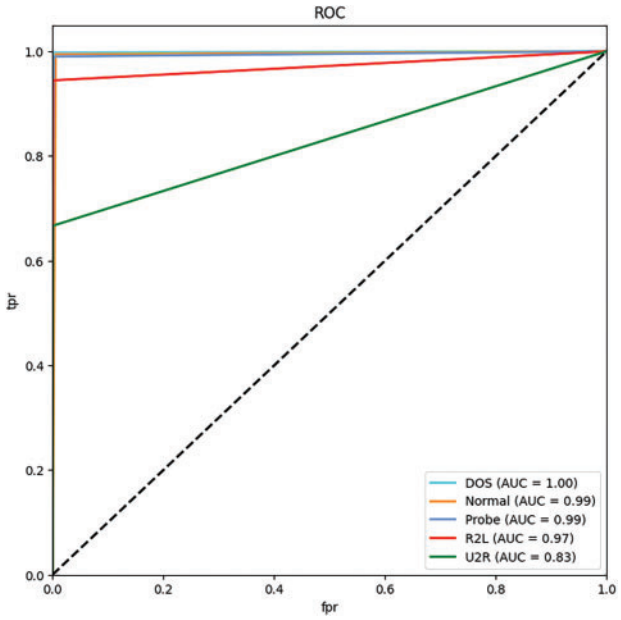


Figure 12: ROC curve for multi-class classification on natural NSL-KDD dataset

A comparison was made with the models from other papers, and the results are shown in Table 7.

Table 7: The comparison with other method on NSLKDD

Model	Precision	Recall	F1
Proposed method	0.994	0.994	0.994
CNN-LSTM [1]	0.886	0.875	0.851
CNN-GRU [28]	0.992	0.992	0.992
LSTMCNN-B [29]	0.936	0.812	–
ANN [30]	0.99	0.967	0.957
CNN-LSTM [31]	–	0.89	0.94

The proposed method achieves 0.994 in Precision, Recall, and F1 values. Compared with CNN-LSTM (0.886/0.875/0.851) and LSTMCNN-B (–/0.812/0.936), the proposed method has a significant improvement in comprehensive performance. The proposed model demonstrates strong overall performance. It achieves high classification effectiveness while maintaining accuracy. It also improves recall and shows excellent generalization ability. Some models, such as ANN, achieve similar precision to the proposed model, but they fall short in the other two metrics and fail to achieve the balance.

4.5 Evaluation Using the Gas Pipeline Dataset

This paper employs the CIC-IDS2017 and NSL-KDD datasets, which are widely recognized for general network intrusion detection. To further assess the model's performance in industrial network intrusion detection, it is also tested on the Mississippi Gas Pipeline dataset. The experimental results are shown in the following Table 8.

Table 8: The precision, recall, and F1-score of every kind of data in the Gas Pipeline dataset

Class	Precision	Recall	F1
Normal	0.95	0.98	0.96
NMRI	1.00	0.02	0.03
CMRI	0.94	0.98	0.96
MSCI	0.97	0.94	0.96
MPCI	0.97	0.98	0.97
MFCI	1.00	0.96	0.98
DoS	0.99	0.98	0.98
Reconnaissance	1.00	1.00	1.00

The experimental results show that the model's performance is overall excellent, with Precision, Recall, and F1 values maintained at high levels for all categories. Especially in the Reconnaissance category, the model achieves a perfect performance of precision, recall, and F1 value of 1.00, indicating that the model can detect this category of attacks completely and correctly. Among the main attack categories, the F1 values of CMRI and MSCI both reached 0.96, while the F1 values of MPCI and MFCI were 0.97 and 0.98, respectively. These results show that the model can detect the main attack types in industrial control systems with a low false alarm rate.

However, for the NMRI category, although the accuracy reached 1.00, the recall was only 0.02, and the F1 value was only 0.03. This indicates that the model cannot correctly identify any real attack samples in this category. This is further evidenced by the lower AUC value of NMRI in Fig. 13. This may be because the number of samples in this category is too small, or the feature distribution is too different from other categories, making it difficult for the model to learn an effective classification boundary.

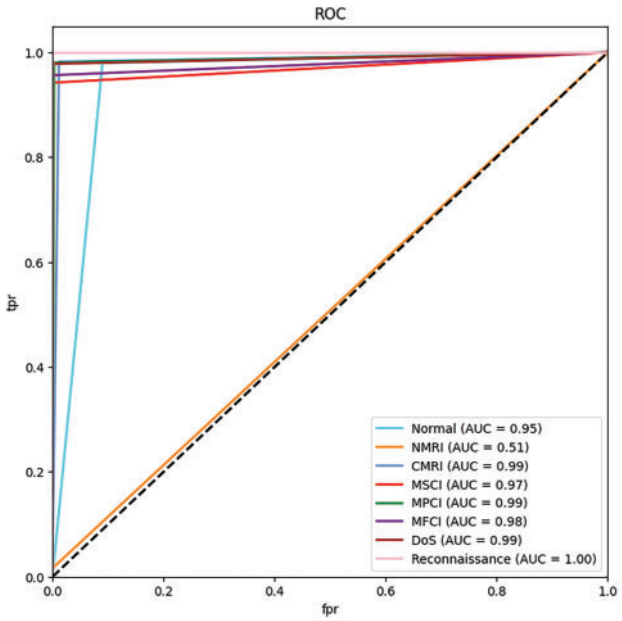


Figure 13: ROC curve for multi-class classification on natural Gas pipeline dataset

Overall, the experimental results confirm the model's applicability in industrial network environments. However, there is still potential for improvement in detecting specific minority classes, such as NMRI.

The above conclusion can also be further verified by the ROC plot. The classification confidence of the NMRI category is relatively low, but the confidence of the other categories is very high.

To further explore the effectiveness of the hierarchical structure proposed in this paper, ablation experiments were conducted to test the single-layer CNN-BiLSTM-Attention and the two-layer model proposed in this paper. The results are shown in Figs. 14 and 15.

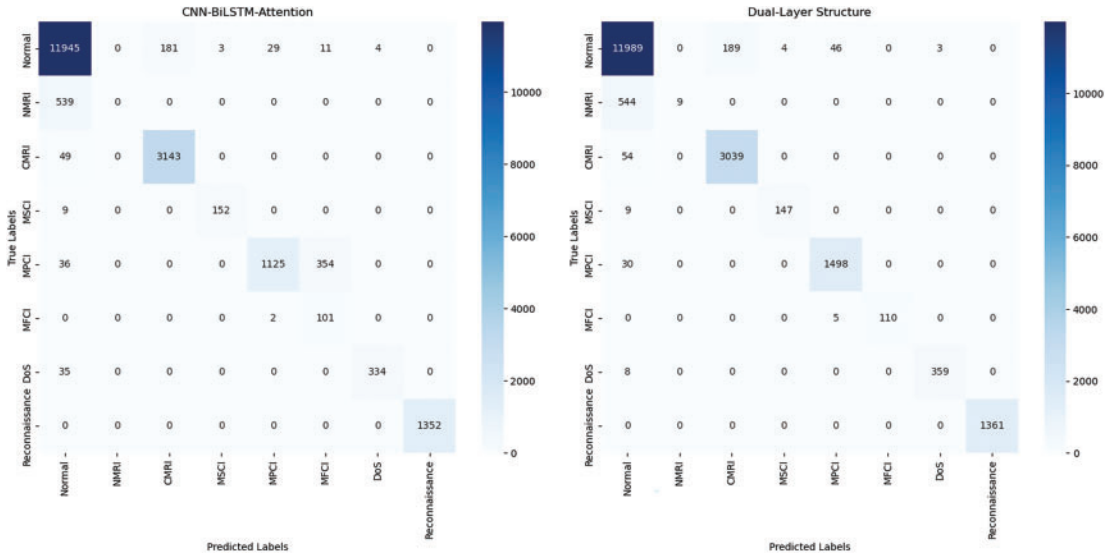


Figure 14: Confusion matrix comparison on Gas pipeline dataset

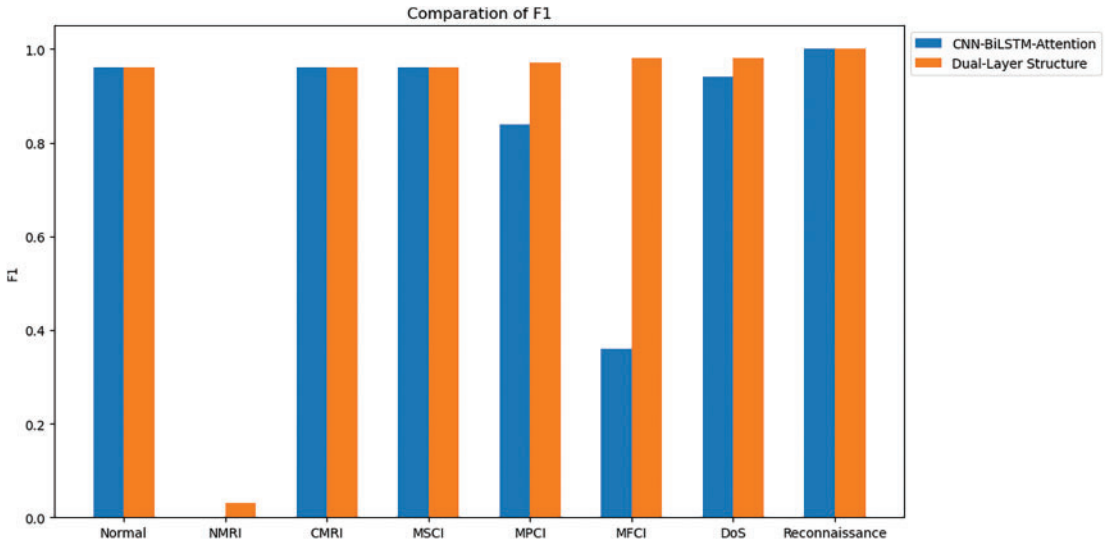


Figure 15: Multi-class classification comparison of F1 on Gas pipeline dataset

Further comparison of the F1 value shows that the proposed two-layer model greatly improves the recognition of the two minority categories of MPCI and MFCI, and to some extent improves Dos. This shows that the proposed two-layer model is indeed feasible and effective in detecting minority categories.

Overall, the model shows excellent classification performance in both general network environments (CIC-IDS2017 dataset and NSL-KDD dataset) and industrial network environments (Mississippi Gas Pipeline dataset). It can not only accurately detect the majority class of attacks, but also identify the minority class of attacks. This verifies the applicability and robustness of the model in different network scenarios, reflects its good generalization ability, and provides reliable technical support for multi-scenario applications of network traffic intrusion detection.

According to Table 9, the model proposed in this paper performs best in Precision, Recall, and F1, which are 0.96, 0.95, and 0.94, respectively, outperforming other benchmark methods. In comparison, the FS-IDS model performs slightly worse in Precision, Recall, and F1, with values of 0.90, 0.89, and 0.89, respectively. Traditional models, such as SVM and Decision Trees, show lower Precision and F1 scores. Although they perform well in Recall, their poor Precision could lead to higher false alarm rates. In contrast, the proposed model not only achieves a better balance between accuracy and recall but also demonstrates high overall performance. Overall, the model in this paper significantly outperforms other methods, showing stronger classification ability and better generalization in intrusion detection tasks.

Table 9: The comparison with other method on gas pipeline

Model	Precision	Recall	F1
Proposed method	0.96	0.95	0.94
FS-IDS [32]	0.90	0.89	0.89
SVM [33]	0.78	0.94	0.85
Decision Tree [34]	0.86	0.85	0.87
CNN-GRU [35]	0.79	0.79	0.76

5 Conclusion

This paper proposes a two-layer network intrusion detection model. The first layer integrates a convolutional neural network (CNN), a bidirectional long short-term memory network (BiLSTM), and an attention mechanism, while the second layer employs a stacking ensemble learning approach. The model is designed to effectively detect network intrusions, especially for the challenging problem of identifying minority class attacks.

In addition to the CICIDS2017 and NSL-KDD datasets, industrial network datasets were incorporated into the learning process. This enhances the adaptability of the model in various network environments such as general networks and industrial network traffic. Experimental results show that the model proposed in this paper outperforms other existing methods in terms of accuracy, precision, and recall, especially in minority-class attack detection.

In conclusion, this paper proposes a powerful and adaptive network intrusion detection model that can effectively deal with mainstream and minority-class attacks, especially with strong cross-domain adaptability for performance in different types of network environments.

Although this study has achieved some results, there is still room for improvement in minority class attack recognition. Future research can further optimize the model by, for example, improving the attention mechanism to reduce the possible misjudgment behaviors in the first layer, as well as exploring the

application of data enhancement techniques (e.g., generative adversarial networks) and migration learning to improve the model's ability to recognize minority classes. In addition, extending more datasets for training to improve the robustness and real-time performance of the model, which in turn enhances its performance in real-world deployments. These improvements will help enhance the overall performance of the model and drive its application in complex environments.

Acknowledgement: The authors sincerely appreciate the support from the Liaoning Province Nature Fund Project; and Scientific Research Project of Liaoning Province Education Department.

Funding Statement: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)—Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government (MSIT) (IITP-2025-RS-2022-00156334); in part by Liaoning Province Nature Fund Project (2024-BSLH-214).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Jun Wang, Chaoren Ge, Qiang Fu, Hoekyung Jung; data collection: Jun Wang, Qiang Fu, Yihong Li, Huimin Zhao, Kerang Cao; experiment analysis and interpretation of results: Jun Wang, Chaoren Ge, Yihong Li, Huimin Zhao; writing—original draft preparation: Jun Wang, Chaoren Ge; writing—review and editing: Jun Wang, Chaoren Ge, Qiang Fu, Hoekyung Jung. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Cui J, Zong L, Xie J, Tang M. A novel multi-module integrated intrusion detection system for high-dimensional imbalanced data. Appl Intell. 2023;53(1):272–88. doi:10.1007/s10489-022-03361-2.
2. Ding H, Chen L, Dong L, Fu Z, Cui X. Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener Comput Syst. 2022;131(7):240–54. doi:10.1016/j.future.2022.01.026.
3. Bedi P, Gupta N, Jindal V. I-SiamIDS: an improved Siam-IDS for handling class imbalance in network-based intrusion detection systems. Appl Intell. 2021;51(2):1133–51. doi:10.1007/s10489-020-01886-y.
4. Ullah F, Ullah S, Srivastava G, Lin JC. IDS-INT: intrusion detection system using transformer-based transfer learning for imbalanced network traffic. Digit Commun Netw. 2024;10(1):190–204. doi:10.1016/j.dcan.2023.03.008.
5. Al-Ambusaidi M, Zhang Y, Muhammad Y, Yahya A. ML-IDS: an efficient ML-enabled intrusion detection system for securing IoT networks and applications. Soft Comput. 2024;28(2):1765–84. doi:10.1007/s00500-023-09452-7.
6. Apruzzese G, Pajola L, Conti M. The cross-evaluation of machine learning-based network intrusion detection systems. IEEE Trans Netw Serv Manag. 2022;19(4):5152–69. doi:10.1109/TNSM.2022.3157344.
7. Saheed YK, Misra S. A voting gray wolf optimizer-based ensemble learning models for intrusion detection in the Internet of Things. Int J Inf Secur. 2024;23(3):1557–81. doi:10.1007/s10207-023-00803-x.
8. Okey OD, Maidin SS, Adasme P, Lopes Rosa R, Saadi M, Carrillo Melgarejo D, et al. BoostedEnML: efficient technique for detecting cyberattacks in IoT systems using boosted ensemble machine learning. Sensors. 2022;22(19):7409. doi:10.3390/s22197409.
9. Bakhsh SA, Khan MA, Ahmed F, Alshehri MS, Ali H, Ahmad J. Enhancing IoT network security through deep learning-powered intrusion detection system. Internet Things. 2023;24(10):100936. doi:10.1016/j.iot.2023.100936.

10. Sai Chaitanya Kumar G, Kiran Kumar R, Parish Venkata Kumar K, Raghavendra Sai N, Brahmaiah M. Deep residual convolutional neural network: an efficient technique for intrusion detection system. Expert Syst Appl. 2024;238(4):121912. doi:10.1016/j.eswa.2023.121912.
11. Serrano W. CyberAIBot: artificial intelligence in an intrusion detection system for CyberSecurity in the IoT. Future Gener Comput Syst. 2025;166:107543. doi:10.1016/j.future.2024.107543.
12. Luo F, Yang Z, Zhang Z, Wang Z, Wang B, Wu M. A multi-layer intrusion detection system for SOME/IP-based in-vehicle network. Sensors. 2023;23(9):4376. doi:10.3390/s23094376.
13. Harini R, Maheswari N, Ganapathy S, Sivagami M. An effective technique for detecting minority attacks in NIDS using deep learning and sampling approach. Alex Eng J. 2023;78(1):469–82. doi:10.1016/j.aej.2023.07.063.
14. Wisanwanichthan T, Thammawichai M. A double-layered hybrid approach for network intrusion detection system using combined naive Bayes and SVM. IEEE Access. 2021;9(4):138432–50. doi:10.1109/ACCESS.2021.3118573.
15. Chen K, Xue B, Zhang M, Zhou F. Evolutionary multitasking for feature selection in high-dimensional classification via particle swarm optimization. IEEE Trans Evol Comput. 2022;26(3):446–60. doi:10.1109/TEVC.2021.3100056.
16. Jin X, Wei B, Deng L, Yang S, Zheng J, Wang F. An adaptive pyramid PSO for high-dimensional feature selection. Expert Syst Appl. 2024;257(12):125084. doi:10.1016/j.eswa.2024.125084.
17. Pramanik R, Sarkar S, Sarkar R. An adaptive and altruistic PSO-based deep feature selection method for Pneumonia detection from chest X-rays. Appl Soft Comput. 2022;128(6):109464. doi:10.1016/j.asoc.2022.109464.
18. Song XF, Zhang Y, Gong DW, Sun XY. Feature selection using bare-bones particle swarm optimization with mutual information. Pattern Recognit. 2021;112(4):107804. doi:10.1016/j.patcog.2020.107804.
19. Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy; 2018 Jan 22–24; Funchal, Portugal. doi:10.5220/0006639801080116.
20. Goryunov MN, Matskevich AG, Rybolovlev DA. Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset. Proc ISP RAS. 2020;32(5):81–94. doi:10.15514/ISPRAS-2020-32(5)-6.
21. Kurniabudi, Stiawan D, Darmawijoyo, Bin Idris MY, Bamhdi AM, Budiarto R. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access. 2020;8:132911–21. doi:10.1109/ACCESS.2020.3009843.
22. Salo F, Injadat M, Nassif AB, Shami A, Essex A. Data mining techniques in intrusion detection systems: a systematic literature review. IEEE Access. 2018;6:56046–58. doi:10.1109/ACCESS.2018.2872784.
23. Morris T, Gao W. Industrial control system network traffic datasets to facilitate intrusion detection system research. In: Shenoi S, Butts J, editors. Critical infrastructure protection VIII. Berlin/Heidelberg, Germany: Springer; 2014. doi: 10.1007/978-3-662-45355-1.
24. Belarbi O, Khan A, Carnelli P, Spyridopoulos T. An intrusion detection system based on deep belief networks. In: International Conference on Science of Cyber Security; 2022; Cham, The Switzerland: Springer International Publishing. doi:10.1007/978-3-031-17551-0_25.
25. Yun X, Xie J, Li S, Zhang Y, Sun P. Detecting unknown HTTP-based malicious communication behavior *via* generated adversarial flows and hierarchical traffic features. Comput Secur. 2022;121(3):102834. doi:10.1016/j.cose.2022.102834.
26. Wang J, Si C, Wang Z, Fu Q. A new industrial intrusion detection method based on CNN-BiLSTM. Comput Mater Contin. 2024;79(3):4297–318. doi:10.32604/cmc.2024.050223.
27. Verkerken M, D'hooge L, Sudyana D, Lin YD, Wauters T, Volckaert B, et al. A novel multi-stage approach for hierarchical intrusion detection. IEEE Trans Netw Serv Manag. 2023;20(3):3915–29. doi:10.1109/TNSM.2023.3259474.
28. Cao B, Li C, Song Y, Fan X. Network intrusion detection technology based on convolutional neural network and BiGRU. Comput Intell Neurosci. 2022;2022(7):1942847. doi:10.1155/2022/1942847.
29. Aldarwbi MY, Lashkari AH, Ghorbani AA. The sound of intrusion: a novel network intrusion detection system. Comput Electr Eng. 2022;104(1):108455. doi:10.1016/j.compeleceng.2022.108455.

30. Zakariah M, AlQahtani SA, Alawwad AM, Alotaibi AA. Intrusion detection system with customized machine learning techniques for NSL-KDD dataset. Comput Mater Contin. 2023;77(3):4025–54. doi:10.32604/cmc.2023.043752.
31. Bamber SS, Katkuri AVR, Sharma S, Angurala M. A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system. Comput Secur. 2025;148:104146. doi:10.1016/j.cose.2024.104146.
32. Ouyang Y, Li B, Kong Q, Song H, Li T. FS-IDS: a novel few-shot learning based intrusion detection system for SCADA networks. In: ICC 2021—IEEE International Conference on Communications; 2021 June 14–23; Montreal, QC, Canada.
33. Anton SDD, Sinha S, Dieter Schotten H. Anomaly-based intrusion detection in industrial data with SVM and random forests. In: 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM); 2019 Sep 19–21; Split, Croatia. p. 1–6.
34. Al-Asiri M, El-Alfy EM. On using physical based intrusion detection in SCADA systems. Procedia Comput Sci. 2020;170:34–42. doi:10.1016/j.procs.2020.03.007.
35. Cao B, Li C, Song Y, Qin Y, Chen C. Network intrusion detection model based on CNN and GRU. Appl Sci. 2022;12(9):4184. doi:10.3390/app12094184.