Hindawi
Security and Communication Networks
Volume 2021, Article ID 6610675, 15 pages
https://doi.org/10.1155/2021/6610675

Research Article

A Novel Framework Design of Network Intrusion Detection Based on Machine Learning Techniques

Chongzhen Zhang,[1] Yanli Chen,[1] Yang Meng,[2] Fangming Ruan ,[1] Runze Chen,[1] Yidan Li,[1] and Yaru Yang[1]

[1] *School of Big Data and Computer Science, Guizhou Normal University, Guiyang 550025, China*
[2] *Department of Biomedical Engineering, School of Medicine, Tsinghua University, Beijing 100084, China*

Correspondence should be addressed to Fangming Ruan; rfm@gznu.edu.cn

Received 27 October 2020; Revised 20 December 2020; Accepted 9 January 2021; Published 25 January 2021

Academic Editor: Savio Sciancalepore

Traditional machine learning-based intrusion detection often only considers a single algorithm to identify intrusion data, lack of the flexibility method, low detection rate, no handing high-dimensional data, and cannot solve these problems well. In order to improve the performance of intrusion detection system, a novel general intrusion detection framework was proposed in this paper, which consists of five parts: preprocessing module, autoencoder module, database module, classification module, and feedback module. The data processed by the preprocessing module are compressed by the autoencoder module to obtain a lower-dimensional reconstruction feature, and the classification result is obtained through the classification module. Compressed features of each traffic are stored in the database module which can both provide retraining and testing for the classification module and restore these features to the original traffic for postevent analysis and forensics. For evaluation of the framework performance proposed, simulation was conducted with the CICIDS2017 dataset to the real traffic of the network. As the experimental results, the accuracy of binary classification and multiclass classification is better than previous work, and high-level accuracy was reached for the restored traffic. At the last, the possibility was discussed on applying the proposed framework to edge/fog networks.

1. Introduction

1.1. Background. In recent years, the widespread use of computers and networks and the emergence of new technologies such as big data, internet of things, and cloud computing have prompted new threats in this modern complex environment; there has been a significant increase in the number of malicious activities. The need to protect network resources from cyber threats has been growing, and the intrusion detection system (IDS) is critical in the cybersecurity to achieve robust protection against cyber attackers; IDS enables us to detect, identify, and recognize anomalous behaviors caused by intruders in networks and computer systems. Machine learning (ML) methods can be used for prediction and classification by learning features in advance. Based on training approach of the classifier, IDS can be divided into supervised and unsupervised; supervised learning learns labeled training samples as much as possible in order to be able to predict data outside the training sample set; unsupervised learning is learning training samples unlabel to discover the structural knowledge in the training sample set. Some techniques have been applied to intrusion detection by researchers [1], for instance, random forest (RF) [2], support vector machine (SVM) [3], *k*-nearest neighbor (KNN) [4], or artificial neural network (ANN) [5].

Application of the deep neural network (DNN) in the field of intrusion detection has become a hot-point, and many researchers paid much attention for the application, convolutional neural network (CNN) [6], for example, deep reinforcement learning (DRL) [7] and hybrid DNN structure [8, 9]. DNN as an effective method is evolved from the shallow neural network (SNN), a branch of ANN, and also the focus of deep learning research. Differing from traditional SNN with more network hierarchies, DNN has stronger ability to model or abstract representations and can simulate more complex models. DNN has hence great potential in realizing effective data representation to construct useful methods.

Autoencoder (AE) as a model for feature extraction will be the focus in this study. AE is an unsupervised DNN that learns features in an unsupervised learning manner from unlabeled data. Two basic structures are included in AE, i.e., encoder and decoder. The encoding layer reduces the dimension by compressing the input data, which can reduce the number of nodes in the output layer and use it for feature reduction. The decoding layer attempts to obtain the original data by decoding the encoded data. Typically, the number of nodes in the input layer is equal to the number of nodes in the output layer. Since the unlabeled data are used for training, the reconstruction error can be obtained directly by comparing the reconstruction output with the original input, and the optimal AE model is obtained by minimizing the loss of the reconstruction output through layer-by-layer training.

If we add the L1 norm to the AE, constraining the number of nodes in each layer (most of which should be 0 and only a few of which should not be 0), following that the Sparse Autoencoder (SAE) model can be obtained. The expressions obtained each time are as sparse as possible because sparse expressions tend to be more efficient than others.

1.2. Problem Statement. Traditional ML-based intrusion detection often only considers using a single algorithm to identify intrusion data, which has problems such as rigid methods, low detection rates, and high-dimensional data. How to design a good intrusion detection framework to adapt to the modern Internet environment and how to flexibly and extensively combine different machine learning technologies to solve problems are problems worth considering. This paper focuses on the design of such a general intrusion detection framework to improve the performance of various aspects of intrusion detection.

DNN can achieve effective representations that improve the classification results of traditional supervised machine learning algorithms. Although several approaches that rely on deep learning techniques are effective, they are still limited by time complexity.

Inspired by the autoencoder (AE) model, we have conducted some studies using the AE model in the actual application scenarios of IDS. AE helps to reconstruct the input feature and transform it into a hyperspace representation associated with the input data, reduce the impact of high-dimensional redundant features, and reduce training complexity. We combined AE with the supervised machine learning algorithm to significantly improve the performance of the classification task.

In this research, we propose a novel intrusion detection framework that can solve the above problems, which consists of five parts: preprocessing module, autoencoder module, database module, classification module, and feedback module. The preprocessed data are compressed by the SAE model of the AE module to obtain a lower-dimensional reconstruction feature, and the classification result is obtained through the classification module. The compressed features of each traffic are stored in the database of the database module, which we call it feature library. This library can provide retraining and testing for the classification module and can also restore these features to the original traffic for postevent analysis and forensics.

1.3. Key Contributions and Paper Organization. In summary, the paper's main contributions are as follows:

(1) We propose a novel intrusion detection framework to improve classification capabilities. Simultaneously, the retraining of the classifier in the classification module is realized through the database module and the feedback module so as to ensure the high accuracy rate of the classification module continuously.

(2) We developed a novel classification method by combining SAE and RF. Our approach realizes the potential of effective representation and dimensionality reduction to improve the classification results for traditional ML algorithms in binary and multiclass classification.

(3) We take full use of the characteristics of the SAE model, combined with the feature library in the database module, to restore the flow before compression, which can be used for postevent analysis and forensics.

(4) We evaluate our proposed framework using the CICIDS2017 dataset and give the training and testing times. Compared with different methods in the related work using the same dataset, we have achieved the best value in the binary and multiclass classification.

The remainder of the paper is organized as follows. Section 2 states the selection criteria for the relevant work and introduces these research studies. Section 3 briefly describes the dataset used to evaluate our proposed framework and some of the preprocessing required for the subsequent experiments. Section 3 focuses on our proposed framework and the principles involved in it. In Section 4, we evaluate the performance of our approach based on the experimental results and compare it with some previous work. The application of IDS to edge/fog networks is discussed in Section 5. Finally, Section 6 summarizes and discusses future directions.

2. Related Work

The purpose of this section is to review recent work related to the design of intrusion detection systems. The selection process was based on certain criteria as follows:

(1) Novel framework

(2) Relevant to the CICIDS2017 dataset

(3) Recent and relevant to the autoencoder

2.1. Improved Algorithm. Many improved algorithms have been applied to intrusion detection systems to improve the detection capabilities, and these approaches show better performance than single learning methods. Elbasiony et al. [10] presented a hybrid detection framework based on data mining classification and clustering techniques that used a random forest algorithm together with a weighted k-means algorithm to construct a hybrid framework that chose anomalous clusters by injecting known attacks into uncertain connected data, which is evaluated on the KDD'99 dataset to achieve a 98.3% detection rate. Authors in [11] presented a framework for automatic intrusion detection that employed the affinity propagation (AP) algorithm to understand the behavior of objects by dynamically clustering stream data, automatically label data and adapting to normal changes in behavior, and identifying anomalies. The test results show that better results were achieved compared with the adaptive sequential Karhunen–Loeve method and static AP as well as three other static anomaly detection methods. Yao et al. [12] proposed a hybrid multilevel data mining-based intrusion detection framework (HMLD), which the authors claimed can detect both known and unknown attacks. The KDDCUP99 dataset was used to evaluate the performance of HMLD, and experimental results showed that HMLD could achieve 96.70% accuracy. Although [10–12] have all achieved good results with the KDD99 dataset, these datasets are outdated and contain not only normal data but also overly simple attack data, making it difficult to use these datasets to simulate today's highly complex network environment. Using these algorithms to analyze malicious traffic in relatively new datasets is also difficult to achieve the desired effect.

In [13], a network attack detection method that combined stream computing with deep learning was proposed. The method used sliding windows to mine frequent patterns to form pattern library quickly detects whether there are malicious behaviors in the data stream and used classification algorithms based on deep belief networks and support vector machines (DBN-SVMs) to improve accuracy. The authors only used the data on Tuesday in the CICIDS2017 dataset for 5 classifications, which cannot reflect the real situation, whereas we use all the data in this dataset and made 15 classifications.

Min et al. [14] proposed an autoencoder-based framework for network intrusion detection. The framework used clustering loss (or classification loss) and reconstruction loss jointly trained by the unsupervised clustering module (or classification module) and autoencoder to obtain a more cluster-friendly feature representation for better clustering results.

2.2. Focus on Data Quality. Moreover, the researchers found that the performance of intrusion detection was also dependent on the quality of the training data. The authors in [15] eliminated irrelevant features using genetic algorithm feature selection techniques and using Bayesian networks as a base classifier for predicting attack types, and the method has 98.2653% accuracy on the NSL-KDD dataset. Gu et al. [16] obtained new transformed training data by transforming the original feature into a logarithm marginal density rate, SVM ensemble was then used to build the intrusion detection model, and the model was evaluated on the NSL-KDD dataset to achieve 99.41% accuracy. The training complexity of SVM is highly dependent on the size of the dataset, requiring large amounts of memory and a lot of training time, and the method improves detection rates without taking time into account.

Ahmim et al. [17] proposed three different classifier-based hierarchical intrusion detection systems based on reduced error pruning (REP) tree, JRip algorithm, and Forest PA. The first and second methods took the features of the dataset as input and classified the network traffic as attack/benign; the third classifier took the output of the first classifier, second classifier, and the features of the initial dataset as input. Evaluation using the CICIDS2017 dataset showed that the model obtained 96.665% accuracy. Kumar et al. [18] used random forest regression technique to select a subset of attributes from the original set, and then the selected set of important features were used in the trained classifier to solve the problem of low detection rate. Compared with [17, 18], our method has higher accuracy.

Authors in [19] used the concept of self-learning to train deep neural networks to perform feature extraction by pretraining the network, combining raw and extracted features to train SAE. Nathan et al. [20] designed a stacked nonsymmetric deep autoencoder for unsupervised feature learning, using the RF as a classifier, while also reducing the training time required for training. The method was evaluated using the benchmark KDD Cup'99 and NSL-KDD datasets, which showed better performance in terms of accuracy for binary classification. The main drawback of this method being that the dimensionality reduction mechanism does not considered.

In [9], CNN and long short-term memory (LSTM) network extracted, respectively, the spatial features of single packets and the temporal features of the data stream, and the hybrid network extracted features from the network data to analyze the network traffic. Experiment was conducted on the CICIDS2017 dataset, and the results showed that this approach achieved an overall accuracy of 98.67%. As far as the results are concerned, their 7 classification results are lower than our 15 classifications, and they also face the same problem as the literature [8] mentioned in the next subsection.

2.3. Features Dimensionality Reduction Approaches. Faced with high-dimensional and large-volume data, it has always been the focus of network intrusion detection research. Although many unsupervised learning-based approaches for network intrusion detection have been proposed in recent years, some of them still have limitations and problems. Predictions from high-dimensional learning have redundant features, thus reducing classification accuracy in the raw dataset. In [21], using SAE for feature learning and soft-max regression for classification evaluated models of classes 2, 5, and 23 using the NSL-KDD benchmark dataset, respectively. But, this result seems to be somewhat poor even under multiclassification with the accuracy of less than 80%.

The model proposed in [22] uses SAE to obtain reconstructed new feature representations, using SVM for classification and reducing the training and testing time, and the method was evaluated on the NSL-KDD dataset with 99.416% accuracy and 99.396% accuracy for binary and multiclass classification, respectively. The method improved the SVM accuracy and reduced the training and testing time required; however, both the time provided in their work and the results obtained from simulations in our experimental environment are unacceptable. We focus on overall performance, not just the improvement of classification results.

Abdulhammed et al. [23] used AE and principal component analysis (PCA) to reduce the feature dimensionality of the CICIDS2017 dataset. The low-dimensional features generated by the two techniques were then used to construct various classifiers to detect malicious attacks. The low-dimensional features were then used to construct various classifiers, such as RF, Bayesian networks, linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA) to design IDS. The authors converted the source IP address (source IP) and destination IP address (destination IP) into integer representations to train the model; however, some of the features of the dataset (e.g., Source IP and Timestamp) mentioned in the literature above will make the model adapt to a specific dataset when training the model using the CICIDS2017 dataset; in fact, in separate experiments in [24] on whether to use IPs as features for training, the final detection rate differed by 3%.

In [8], a multimodal sequential intrusion detection method with a special structure for progressive networks was proposed, which composed of a multimodal deep autoencoder (MDAE) and an LSTM. By designing the special structure of a hierarchical progressive network, the method can integrate feature information at different levels in network connections and simultaneously automatically learn temporal information between adjacent network connections. LSTM is used to learn the temporal features of the dataset so that specific types of attacks are correlated with time (e.g., on a Friday morning, the attacks are all port scans), which is not true law. Models can exhibit high performance for test datasets, but the results always seem to be reduced when analyzing actual network traffic.

2.4. Novel Perspective. We are committed to developing an approach with high classification performance and generalization, and some researchers are gradually starting to make some attempts. Yin et al. [25] proposed a GAN-based intrusion detection framework to improve the performance and generalization of classifiers, which used adversarial training to enhance the classifiers and generated false label samples continuously using a generative model to assist the classifiers in improving their detection performance. Madani and Vlajic [26] studied the contamination based on anomaly detection in the adaptive detection intrusion system, using autoencoder reconstruction error as a metric for anomaly detection, using training time and test temporal metrics to evaluate the performance of the model, using NSL-KDD datasets for performance evaluation, and maintaining a stable detection situation while reducing the contamination level of the training dataset to less than 2% compared with a PCA-based IDS. The two work analyze the problem from a novel perspective but are still a bit far from the practical phase.

3. Proposed Framework

In this section, we will introduce the proposed framework and its workflow. The proposed intrusion detection framework is mainly composed of five parts: preprocessing module, autoencoder module, database module, classification module, and feedback module; the various functional modules are maintained to build a powerful intrusion detection framework, which has high accuracy and low training complexity. The proposed framework is shown in Figure 1, in which different functions are represented using different colored lines; the black line represents the main process of the detection function, the orange line represents the process involved in retraining, and the green line represents the process of restore function, where the blue two-way arrows represents the processes that crosses with other functions.

3.1. Workflow. Before that, we first describe each module in the framework separately and then present the three main functions that the entire framework is based on these modules.

(1) Preprocessing module: the raw traffic collected from the network is processed in a predetermined way (Section 4.2) to obtain the initial data

(2) Autoencoder module: the module is mainly composed of an SAE model, which processes the data, removes the unimportant features in the data, and obtains the reconstructed low-dimensional features

(3) Database module: the module consists of a database, which serves as a data storage and dumps center for the entire framework; the database is continuously updated and can be used for other subsequent operations

(4) Classification module: this module mainly uses supervised algorithms to classify traffic, determine whether to an attack, and then whether a warning is given based on the results

(5) Feedback module: this module adjusts its functions through the machine's output status and alarm information

3.1.1. Detection Function. The data collector collects the raw traffic from the network, processes it through the

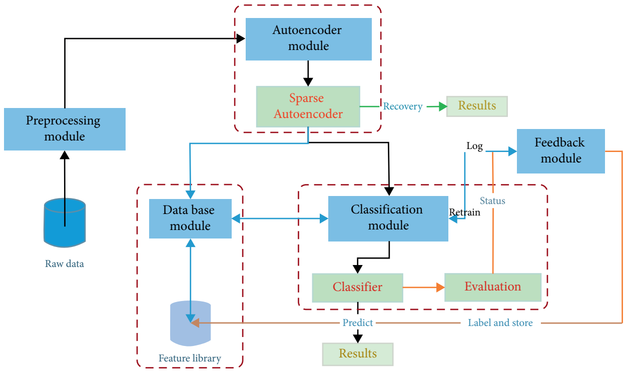


Figure 1: Proposed framework.

preprocessing module, and then extracts the features through autoencoder module. This reconstructed low-dimensional feature is used as input to the classification module. It is also stored in the database module, where the trained classifier in the classification module predicts the input and finally outputs the results.

3.1.2. Retraining Function. Obviously, the training data may not cover all possible normal execution and abnormal behavior patterns which may change over time. We used the results of the evaluation metrics in the classification module, the administrator's analysis of the machine output for status, and alarm information to determine whether the classifier needs to be updated. If it is judged to need updating, the administrator gives a feedback to the classification module and relabels the false alarm entries and stores it in the database; the classification module retrieves the data from the database module to retrain the classifier and then re-evaluates the classifier until the training is complete.

3.1.3. Restore Function. For a successful intrusion detection system, it can not only accurately identify possible intrusions but also provide a basis for the formulation of network security strategies. We make full use of the features of the autoencoder model and use the features stored in the database module to restore the original traffic, which is convenient for subsequent analysis and forensics by the administrator.

3.2. Sparse Autoencoder. Figure 2 gives a simple SAE training diagram; suppose we have N input and output nodes and M hidden layer nodes, first input $x = (x_1, x_2, \ldots, x_n)$ tries to get an equal output $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$, i.e., $x = \hat{x}$. Compress x into a lower-dimensional hidden layer representation that consists of one or more hidden layers $a = (a_1, a_2, \ldots, a_m)$ and then map the hidden representation a to the reconstructed output $\hat{x}$. The output of the neurons in each hidden layer can be calculated by

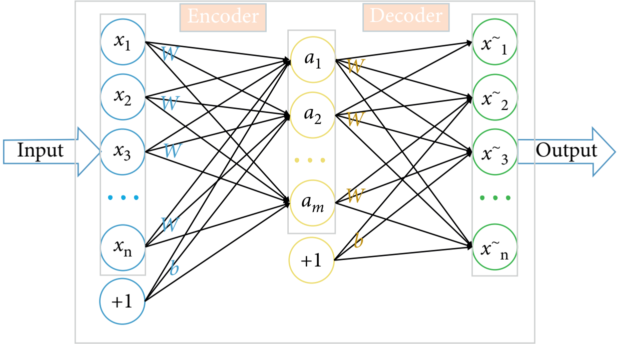


Figure 2: Training diagram of an SAE.

$$a_i^l = f\left(g_i^l\right) = f\left(\sum_{j=1}^{n} W_{ij}^{l-1} \cdot a_j^{l-1} + b_i^{l-1}\right), \tag{1}$$

where a_i^l denotes the i^{th} node of the hidden layer l, and the dimension of the hidden layer weight matrix denotes $W \in \mathbb{R}^{m\times n}$, with a bias vector $b \in \mathbb{R}^m$. We choose Clevert et al. [27] as the activation function (see equation (2)) where we α take 1.0.

$$f(z) = \begin{cases} \alpha\left(e^z - 1\right), & z < 0, \\ z, & z \geq 0. \end{cases} \tag{2}$$

Using the back propagation algorithm to find the best values of the weight matrix W and the bias vector b, the function to minimize the loss is represented as follows:

$$J_{\text{sparse}}(W, b, x, \hat{x}) = \frac{1}{2s}\sum_{i=1}^{s} \|\hat{x} - x\|^2 + \frac{\lambda}{2}\sum_{l=1}^{ls-1}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(W_{ij}^l\right)^2 + \beta\sum_{i=1}^{m} KL\left(\rho \parallel \hat{\rho}_i\right). \tag{3}$$

The first term represents the average sum-of-square errors for all s input data; the second term is the weight decay term, with the parameter λ to control the weight in a particular layer to avoid overfitting, while ls represents the total number of layers; the last term is the sparsity penalty term; for KL distance, which imposes a constraint in the hidden layer to maintain a low average activation value, the value of b controls the weight of the sparsity penalty term, and the KL distance represents here difference value between the two vectors; it can be seen from the constraint function expression that the larger the difference between the mean ρ and the desired sparsity mean $\hat{\rho}_i$, the larger the penalty, ensuring that this sparsity has the effect of making $\hat{\rho}_i$ close to ρ, as it ensures sparse activation of the training data against any given node in the hidden layer, thereby reducing the dependency between features. The output average of the hidden layer nodes is solved as follows:

$$\hat{\rho}_i = \frac{1}{s}\sum_{l=1}^{s}\left[a_i^l\left(x^{(l)}\right)\right]. \tag{4}$$

Besides, we noticed that removing redundant features cannot blindly reduce the dimensionality in [28]; the authors used AE to reduce the features to below 30; the accuracy of the model continued to decrease. Similarly in the literature [23], reducing the dimensionality to around 59, there were significant fluctuations in the performance of each classifier. In this paper, an SAE with two encoders and two decoders was used, with 79 nodes in the input layer, equal to the total number of features in the dataset. The first hidden layer of the SAE reduces the dimensionality to 68 features and reduces the features to 64 in the second hidden layer. After training the parameters, the final model can perform the classification task at the final stage. Algorithm 1 gives the pseudocode of the SAE training process.

3.3. Classification Algorithm. In the classification module, we used RF as the main algorithm of the classification module, and random forest is an ensemble method composed of decision trees. In the experimental stage, we also used decision tree (DT) for comparison experiments to verify the performance of RF.

Random forests have several advantages: as the number of trees in the forest increases, the variance of the model decreases while the bias remains constant; resistance to overfitting; the number of control and model parameters is small; and there is no need to select features because they can use a large number of potential attributes.

A DT is a tree structure (which can be binary or nonbinary tree). Each of its nonleaf nodes represents a test on a feature attribute, each branch represents the output of this feature attribute on certain value domain, and each leaf node stores a category. The process of making a decision using a DT involves starting from the root node, testing the corresponding feature attribute in the item to be classified, and selecting the output branch according to its value until it reaches the leaf node, where the category stored in the leaf node is used as the decision result. Classification and regression tree (CART) is chosen as the DT algorithm for this experiment (see [29] for details of CART).

CART is a binary tree, which uses the binary segmentation method to cut the data into two parts to enter the left subtree and right subtree, respectively. And, each nonleaf node has two children, so CART has one more leaf nodes than nonleaves. In CART classification, the Gini index is used to select the best features for data partitioning, and Gini describes purity, the smaller the value, the higher the purity and the better the features. We select the attribute that minimizes the Gini index after division in the candidate set as the optimal subattribute; each iteration in CART reduces the Gini coefficient. For sample set D, the number is $|D|$. Suppose there are K classes and the k^{th} class is $|C_k|$, then the Gini index expression for sample set D is

$$\operatorname{Gini}(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2. \tag{5}$$

For sample D, the number is $|D|$, which divides D into $|D_1|, |D_2|, \ldots, |D_n|$ n parts according to certain value a of feature A; then, under the condition of feature A, the Gini index expression for sample set D is

$$\operatorname{Gini}(D, A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} \operatorname{Gini}(D_i). \tag{6}$$

Algorithm 2 gives the basic process for CART creation, which involves following variable selection criteria and splitting criteria to grow the tree until the stopping criteria is met.

Random forest works on the concept of fair elections because the result is based on the predictions of several independent DTs, and a group of DTs with the same prediction will outperform a single DT. For classification, the prediction is the majority vote, and for regression, the prediction is the mean. It uses a random subset of a given data set and a random subset of features to create an individual DT so that the individual DT is barely correlated with each other, and the lower correlations between trees become more independent.

For feature selection, the random forest technique assigns weights to different features based on the degree to which a particular feature reduces the accuracy of the individual DT classification. The fact that the dependent variable has a low weight is due to the fact that the random forest technique ensures that the correlations between individual DT are small. Algorithm 3 gives the normal tree structured classifier generation process.

4. Experimental Results

Classification performance is a core function of IDS; in this section, we focus on evaluating classification performance and the experimental validation of our proposed approach. We first described in detail the CICIDS2017 dataset used for the experiments and its corresponding preprocessing, gave the metrics and the experimental environment used for the evaluation, and compared and analyzed the experimental results. Finally, we performed a simple test of the restore function of the framework.

4.1. CICIDS2017 Dataset. Most current datasets are outdated and unreliable (some commonly used for intrusion detection evaluation are KDD'99 [30] and KDD-NSL [31]); some of which lack the diversity and volume of traffic to cover a wide range of known attacks, while others anonymize packet payload data and do not reflect current trends.

In order to best reflect the real traffic scenarios and updated attack methods in real networks, we chose the dataset CICIDS2017 [32], which contains the latest benign generic attack traffic representative of the real network environment. This dataset constructs the abstract behavior of 25 users based on HTTP, HTTPS, FTP, SSH, and e-mail protocols to simulate the real network environment accurately.

The CICIDS2017 dataset was collected based on real traces of normal and malicious activity in the network traffic. The total number of records in the dataset is 2,830,743. The normal

```
Input: training dataset
Output: trained SAE model
Initialization: $W, b, \lambda, \beta, \rho$
Step 1: perform forward propagation on all input samples
Step 2: calculate the output of each node a in the hidden layer (equation (1))
Step 3: calculate the output error of the cost function (equation (3))
Step 4: updating the weights and biases of each layer using the backpropagation algorithm to reduce errors
Step 5: repeat Step 2, 3, and 4 until the reconstruction error is minimum
End
```

Algorithm 1: The training procedures of SAE.

```
Input: training dataset $D$
Output: CART
$N$: threshold of the number of samples in node. $n$: number of samples in the node
$G$: Gini index threshold for $D$
Gini ($D$): Gini index of $D$
Based on $D$, starting from the root node, if $n < N$ or Gini ($D$) < $G$ or no more features, recursively perform the following operations on each node to construct a binary tree
For each feature $A$, for each of its possible values a, the split will be into $D1$ and $D2$ based on whether the test for $A = a$, and use equation (6) to calculate Gini ($D$, $A$)
Among all possible features $A$ and all its possible segmentation points $a$, the feature with the smallest Gini index and its corresponding segmentation point are selected as the optimal feature and the optimal segmentation point
Generate two subnodes from the current node and assign the training dataset to the two subnodes according to the features
Return CART
```

Algorithm 2: CART creation process.

```
Input: training dateset
Output: tree structured classifier
$S$: number of training samples
$M$: number of features. $m$: number of features input ($m << M$)
$N$: number of trees generated
If the tree to be generated is less than $N$,
  Step 1: from the $S$ training samples, take samples $S$ times in a way with a put-back sampling to form a training set
  Step 2: use unselected samples to make predictions and evaluate their errors
  Step 3: for each node, $m$ features are randomly selected
  Step 4: according to these $m$ features, calculate the best split method
  Step 5: grow to be largest extent possible without pruning
Return tree structured classifier
```

Algorithm 3: The tree classifier generation process.

activity traffic contains 2,273,097 records (about 80.3% of the data) and malicious behavior traffic records 557,646 records (about 19.7% of the data). The distribution of each tag set for the entire dataset is given in Figure 3. The CICIDS2017 dataset is a labeled dataset with 85 features that contain class labels (last column) corresponding to the traffic state. These features are listed in Table 1. The CICIDS2017 dataset is one of the unique datasets containing the latest attacks, and we used this dataset to test and evaluate the ideas presented.

4.2. Data Preprocessing. For the preprocessing of data, in addition to regular operations (such as handing values or nonnumber types, encoding labels, and so on), in the CICIDS2017 dataset, some features make the model adapt to a specific dataset after training, which affects the generalization ability of the model, so we removed the "Flow ID," "Source IP," "Source Port," "Destination IP," and "Timestamp" features among them. Therefore, the total number of features we use contains 79 features.

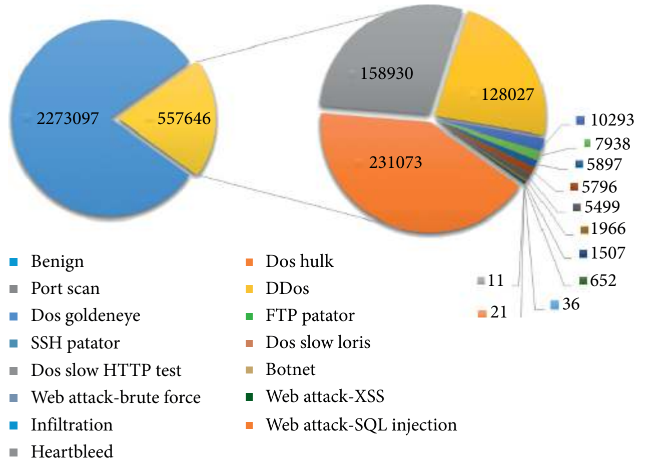


Figure 3: Distribution of labels in the CICIDS2017 dataset.

Table 1: Features of each network traffic in CICIDS2017.

No.	Feature
1	Flow ID
2	Source IP
3	Source port
4	Destination IP
5	Destination port
6	Protocol
7	Timestamp
8	Flow duration
9	Total fwd packets
10	Total backward packets
11	Total length of fwd packets
12	Total length of bwd packets
13	Fwd packet length max
14	Fwd packet length min
15	Fwd packet length mean
16	Fwd packet length std
17	Bwd packet length max
18	Bwd packet length min
19	Bwd packet length mean
20	Bwd packet length std
21	Flow bytes/s
22	Flow packets/s
23	Flow IAT mean
24	Flow IAT std
25	Flow IAT max
26	Flow IAT min
27	Fwd IAT total
28	Fwd IAT mean
29	Fwd IAT std
30	Fwd IAT max
31	Fwd IAT min
32	Bwd IAT total
33	Bwd IAT mean
34	Bwd IAT std
35	Bwd IAT max
36	Bwd IAT min
37	Fwd PSH flags
38	Bwd PSH flags
39	Fwd URG flags
40	Bwd URG flags

Table 1: Continued.

No.	Feature
41	Fwd header length
42	Bwd header length
43	Fwd packets/s
44	Bwd packets/s
45	Min packet length
46	Max packet length
47	Packet length mean
48	Packet length std
49	Packet length variance
50	FIN flag count
51	SYN flag count
52	RST flag count
53	PSH flag count
54	ACK flag count
55	URG flag count
56	CWE flag count
57	ECE flag count
58	Down/up ratio
59	Average packet size
60	Avg fwd segment size
61	Avg bwd segment size
62	Fwd header length
63	Fwd avg bytes/bulk
64	Fwd avg Packets/bulk
65	Fwd avg bulk rate
66	Bwd avg bytes/bulk
67	Bwd avg Packets/bulk
68	Bwd avg bulk rate
69	Subflow fwd packets
70	Subflow fwd bytes
71	Subflow bwd packets
72	Subflow bwd bytes
73	Init_Win_bytes_forward
74	Init_Win_bytes_backward
75	act_data_pkt_fwd
76	min_seg_size_forward
77	Active mean
78	Active std
79	Active max
80	Active min
81	Idle mean
82	Idle std
83	Idle max
84	Idle min
85	Label

Several features in the dataset have a wide range between the maximum value and minimum value, such as "Flow Duration," "Fwd IAT Total," "Idle Mean," and other features. These feature values are not suitable for processing. Therefore, we normalized these features by using equation (7) to map all feature values to the range [0, 1].

$$x_i = \frac{x_i - x_{\min}}{x_i - x_{\max}}, \tag{7}$$

where x_i denotes the value of a certain feature and $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values of this feature.

Table 2: Confusion martrix.

		Predicted class	
		Malicious	Benign
Actual class	Malicious	True positive (TP)	False negative (FN)
	Benign	False positive (FP)	True negative (TN)

In this study, the dataset was divided into training and testing sets in the ratio of 7 : 3.

4.3. Evaluation Metrics. The detection module as the main function of the framework needs to evaluate the performance of the framework accurately; we used four metrics to evaluate the performance of the classification module: accuracy, recall, preference, and F1-score, which are widely used to evaluate the performance of intrusion detection techniques. Their formulas are as follows.

Accuracy (Acc): it is defined as the ratio of correctly classified samples to the total number of samples, which represents the overall performance of the model.

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{8}$$

Recall (Re) or detection rate (DR): it is defined as the ratio of the number of samples correctly classified into a certain class to the actual number of samples of this class.

$$\mathrm{Re} = \mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{9}$$

Precision (Pr): it is defined as the ratio of the number of samples correctly identified as a category to the number of samples identified as such.

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{10}$$

F1-score (F1) or F-measure (FM): it is defined as the harmonic mean of Precision and Recall.

$$F1 = \mathrm{FM} = \frac{2 \cdot \mathrm{Pr} \cdot \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}}. \tag{11}$$

All these evaluation metrics are derived from the four values found in the confusion matrix (Table 2), and for each type of sample, TP is the number of samples correctly classified as that type, TN is the number of samples correctly classified as not that type, FP is the number of samples misclassified as that type, and FN is the number of samples misclassified as not that type.

4.4. Experimental Environment. The details of all experimental implementation configurations are shown in Table 3.

4.5. Results and Discussion. Our experiment was conducted to investigate the performance of our proposed framework in both binary and multiclass classification and to verify the effectiveness of the low-dimensional features extracted by our method using the CICIDS2017 dataset. Moreover, we calculated training and testing times to evaluate the validity of our model. Figures 4 and 5 show the performance of the algorithm using different parameters (here, we consider DT as a tree RF model) on the values of the four evaluation metrics in binary and multiclass classification. Tables 4 and 5 give the training and testing times of the algorithms with different parameters in binary and multiclass classification. From these figures and tables, we can see that different parameter algorithms (except DT) have subtle differences in performance against different attacks; however, there are big differences in the training and testing time of RF, especially for RF-50 trees and RF-100 trees, where the training and testing time is almost doubled. A better algorithm can detect more attacks in shorter time and more adapt to the modern Internet; we choose the appropriate algorithm based on the evaluation metrics and time; RF-10 trees are undoubtedly the most suitable classifier.

Tables 6 and 7 provide a detailed summary of the proposed framework in terms of Accuracy, Precision, Recall, and F1-score, respectively. Table 6 describes the detailed results for binary classifications (Bengin and Abnormal), and Figure 6 is the corresponding confusion matrix, where 681,564 samples were correctly identified as benign traffic and 167,012 samples were correctly identified as abnormal traffic. Table 7 gives the results of the evaluation of various types of traffic, showing that our approach can identify 15 different types of attacks; Figure 7 illustrates the corresponding confusion matrix, and checking the confusion matrix shows the difference. Although the wrong category is detected, the traffic that is still classified as an attack is greater than the traffic that is classified as benign. Attack samples like Infitration, Heardtbleed, and Sql Injection have too few records than other attack samples, but all of them were identified.

We also validated the superiority of our model by comparing its performance with the classification approaches used in related research. Tables 8 and 9 present a comparison of the proposed framework with related work and some traditional ML methods mentioned in their paper. Table 8 focuses on the comparison between the two classification results of the related work, while Table 9 is a comparison between the related work using multiclass classification results. Our proposed framework outperforms previous studies in terms of Accuracy, Precision, Recall, and F1-score.

4.6. Additional Test for Restore Function. We used the compressed traffic in the database and restored it to the original traffic using SAE. Compared with the classification results of the original traffic, we obtained 91.83% accuracy in binary classification and 91.65% accuracy in the multiclass classification.

Table 3: Experimental environment.

Software/hardware	Details
OS	Windows 10
CPU	Intel (R) Core (TM) i5-4200H CPU @ 2.80 Hz
RAM	8 GB
Anaconda	4.8.3
Python	3.7.3
Keras	2.2.4

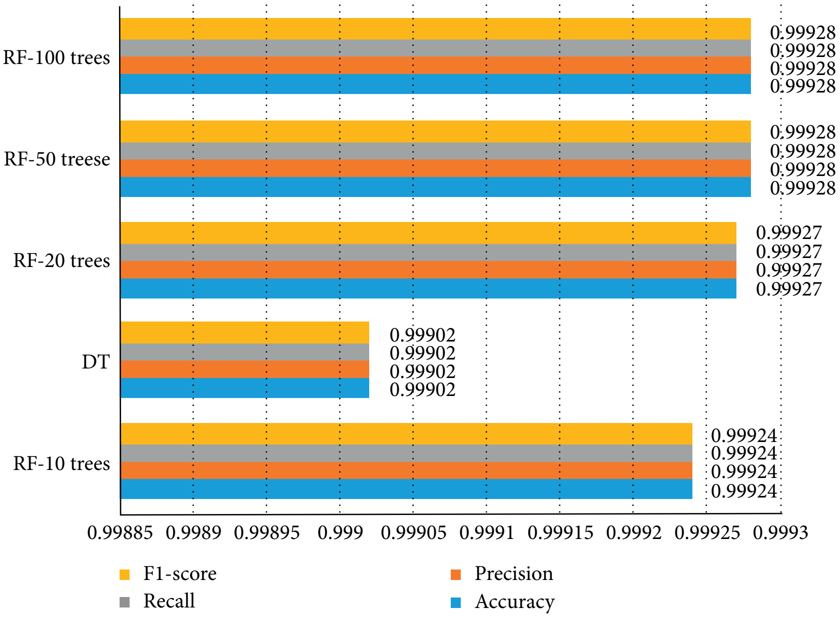


Figure 4: Performance of the algorithm for the values of the four metrics using different parameters in binary classification.

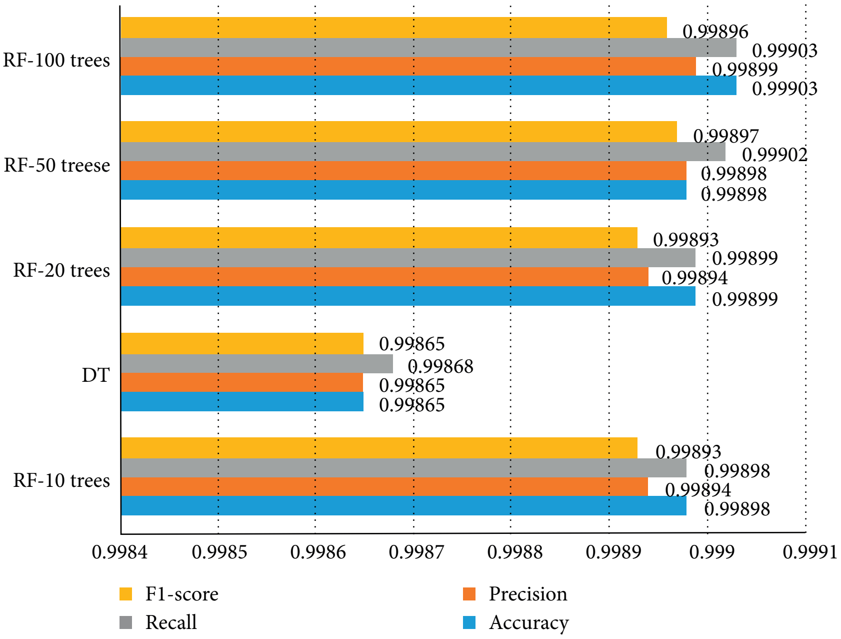


Figure 5: Performance of the algorithm for the values of the four metrics using different parameters in multiclass classification.

Table 4: Training and testing time for different parameters of the algorithm in binary classification.

Method	Training time (sec)	Testing time (sec)
RF-10 trees	246.689	1.696
DT	420.419	0.364
RF-20 trees	502.758	2.734
RF-50 trees	1266.263	6.465
RF-100 trees	2546.473	13.109

Table 5: Training and testing time for different parameters of the algorithm in multiclass classification.

Method	Training time (sec)	Testing time (sec)
RF-10 trees	291.551	3.138
DT	386.454	0.387
RF-20 trees	502.952	4.434
RF-50 trees	1209.425	10.573
RF-100 trees	2425.976	20.507

Table 6: Evaluation results for various types of traffic under binary classification.

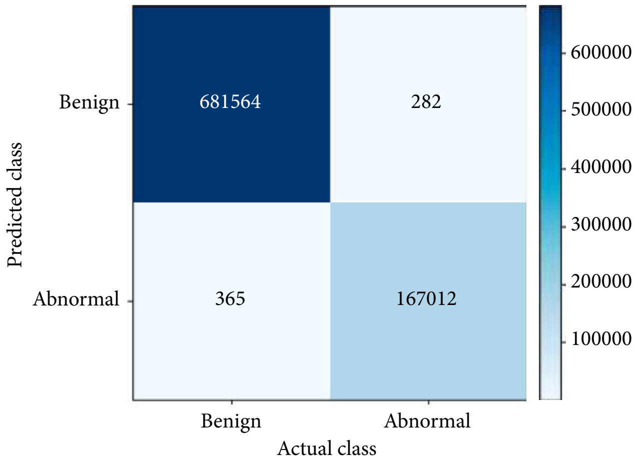
		Accuracy: 0.9992	
	Precision	Recall	F1-score
Bengin	0.9996	0.9994	0.9995
Abnormal	0.9978	0.9983	0.9981
Weighted avg	0.9992	0.9992	0.9992

Table 7: Evaluation results for various types of traffic under multiclass classification.

		Accuracy: 0.9990	
	Precision	Recall	F1-score
Bengin	0.9996	0.9995	0.9995
Bot	0.9142	0.8492	0.8805
DDos	0.9999	0.9997	0.9998
Dos goldeneye	0.9984	0.9793	0.9887
Dos hulk	0.9963	0.9993	0.9978
Dos slow http test	0.9963	0.9891	0.9927
Dos slow loris	0.9960	0.9908	0.9934
FTP-Patator	1.0000	0.9983	0.9992
Heartbleed	1.0000	0.3333	0.5000
Infiltration	1.0000	0.2727	0.4286
PortScan	0.9997	0.9999	0.9998
SSH-patator	0.9943	0.9853	0.9898
Web attack-brute force	0.6808	0.8872	0.7704
Web attack-sql injection	1.0000	0.1667	0.2857
Web attack-XSS	0.4655	0.1378	0.2126
Weighted avg	0.9989	0.9990	0.9989

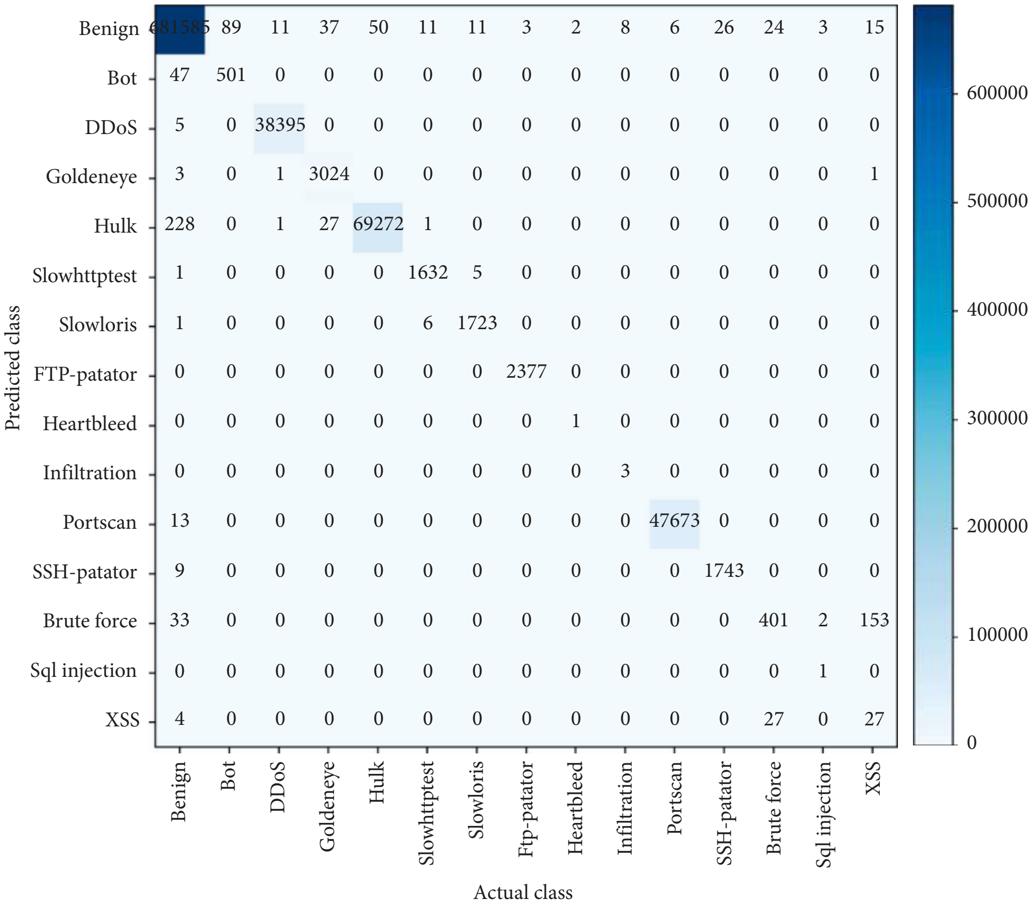
5. Discuss the Application of IDS to Edge/ Fog Networks

The development of technology and realistic demands has extended the cloud computing paradigm to the edge of the network, thus realizing various new applications and services, while facing security and privacy issues. Although our research has successfully demonstrated the effectiveness of the framework and the technology has led to better results in several evaluation metrics of IDS as well as in classification speed, IDS also needs to adapt to the new network architecture [33]. IDS is rapidly developing towards edge/fog networks, and the most likely application of this generic intrusion detection framework we have developed to edge networks is to split some of these modules, removing the database module and the feedback module and embedding the rest in edge devices. Using only binary classification, from our experimental results, where the trained model in our experimental environment was able to process 849,223 traffic within 25 seconds (time for feature extraction and classification), the main drawback compared with the use of feature extraction methods in the literature [34] is that dimensionality reduction is not considered. By deploying the complete framework in the cloud, the restore function serves to some extent to check for duplicate data coming from multiple devices. The cloud completes the retraining



	Benign	Abnormal
Benign	681564	282
Abnormal	365	167012

Figure 6: Confusion matrix of binary classification.



	Benign	Bot	DDoS	Goldeneye	Hulk	Slowhttptest	Slowloris	Ftp-patator	Heartbleed	Infiltration	Portscan	SSH-patator	Brute force	Sql injection	XSS
Benign	681585	89	11	37	50	11	11	3	2	8	6	26	24	3	15
Bot	47	501	0	0	0	0	0	0	0	0	0	0	0	0	0
DDoS	5	0	38395	0	0	0	0	0	0	0	0	0	0	0	0
Goldeneye	3	0	1	3024	0	0	0	0	0	0	0	0	0	0	1
Hulk	228	0	1	27	69272	1	0	0	0	0	0	0	0	0	0
Slowhttptest	1	0	0	0	0	1632	5	0	0	0	0	0	0	0	0
Slowloris	1	0	0	0	0	6	1723	0	0	0	0	0	0	0	0
FTP-patator	0	0	0	0	0	0	0	2377	0	0	0	0	0	0	0
Heartbleed	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Infiltration	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
Portscan	13	0	0	0	0	0	0	0	0	0	47673	0	0	0	0
SSH-patator	9	0	0	0	0	0	0	0	0	0	0	1743	0	0	0
Brute force	33	0	0	0	0	0	0	0	0	0	0	0	401	2	153
Sql injection	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
XSS	4	0	0	0	0	0	0	0	0	0	0	0	27	0	27

Figure 7: Confusion matrix of multiclass classification.

with data from its own database and data transmitted from the Edge IDS and updates the parameters downwards. Edge IDS is independent of each other; by continuously learning new parameters from Cloud IDS, Edge IDS can update and upgrade their detection rules as the network security status changes, avoiding conflicts between Edge IDSs. Compared with [35], we use 14 different attack samples to validate our method; in terms of algorithm, there are some uncertainties in the gene immune detection algorithm; we use the more mature RF algorithm for classification; the RF algorithm and its improved algorithms are widely used in intrusion detection and showed good performance [36]. However, it still has certain limitations and challenges [37], such as multi-source data, false alarm response, and so on. Indeed, edge computing brings benefits and also poses challenges for the design of new IDS.

Table 8: Comparison of the proposed framework in the binary classification with related work using CICIDS2017 dataset evaluation and some of the traditional ML approaches mentioned in their papers.

	Reference + method	Accuracy	Precision	Recall	F1-score
[8]	MDAE + LSTM	0.9990	0.9990	0.9980	0.9990
	Naive Bayes	0.3130	0.3000	0.9790	0.4590
	SVM	0.7990	0.9920	0.3280	0.4930
	DNN	0.9310	0.8270	0.9740	0.9840
	MDAE	0.9140	0.8980	0.8020	0.8400
	LSTM	0.9980	0.9990	0.9970	0.9980
[24]	DNN + without IPs			0.9677	
[18]	Feature selection + ID3	0.9515	0.9500	0.9500	0.9500
[23]	PCA + RF	0.9960		0.9880	0.9970
	AE + RF	0.9950		0.9850	0.9960
	Our approach	**0.9992**	**0.9992**	**0.9992**	**0.9992**

Note. The best values are in bold, and missing value means not provided in reference.

Table 9: Comparison of the proposed framework in the multiclass classification with related work using CICIDS2017 dataset evaluation and some of the traditional ML approaches mentioned in their papers.

	Reference + method	Accuracy	Precision	Recall	F1-score	Remark
[8]	MDAE + LSTM	0.9860	0.9860	0.9960	0.9860	8 classification
	Naive Bayes	0.2500	0.7670	0.2500	0.1880	
	SVM	0.7990	0.7570	0.7990	0.7230	
	DNN	0.9480	0.9650	0.9480	0.9530	
	MDAE	0.9040	0.9920	0.9000	0.9110	
	LSTM	0.9700	0.9680	0.9860	0.9730	
[17]	DT + rule-based	0.9967		0.9448		15 classification
	RF	0.9559		0.9305		
	REP tree	0.9340		0.9164		
	Multilayer Perceptron	0.8524		0.7783		
	Naive Bayes	0.7453		0.8251		
	Jrip	0.9447		0.9340		
	J48	0.9348		0.9199		
[13]	DBN-SVM		0.9774	0.9768	0.9768	Use only Tuesday's dataset with 5 classification
[9]	CNN + LSTM	0.9867		0.9721	0.9332	7 classification
	CNN	0.9844		0.9646	0.9311	
	LSTM	0.9683		0.9421	0.9097	
[23]	PCA + RF	0.9880	0.9890	0.9880	0.9880	15 classification
	AE + RF	0.9950			0.9950	
	Our approach	**0.9990**	**0.9990**	**0.9989**	**0.9989**	15 classification

Note. The best values are in bold, and missing value means not provided in reference.

6. Conclusion and Future Work

A novel framework of intrusion detection was proposed and discussed, which is comprised of five modules: preprocessing module, autoencoder module, database module, classification module, and feedback module. The preprocessed data were compressed by the SAE model of the AE module to obtain a lower-dimensional reconstruction feature, and the classification result was obtained through the classification module. The compressed features of each traffic were stored in the database of the database module; this library can provide retraining and testing for the classification module and can also restore these features to the original traffic for postevent analysis and forensics. The performance of framework proposed was evaluated with the CICIDS2017 dataset to simulate the real traffic of the network. The proposed framework, shown by experiment result, provided higher accuracy than other similar work and traditional ML methods, especially in the case of multiclass classifications. In the upcoming time, effort will be made to improve the two functions, the restore function and retraining function, turning retraining function into adaptive update so as to reduce manual intervention, whereas getting traceable effect in restore function. Furthermore, this research will serve as the basis for further research and investigation to enable the development of effective IDSs that can be used in a complex network (e.g., edge networks) environment.

Data Availability

The CICIDS2017 dataset used to support the findings of this study can be downloaded from https://www.unb.ca/cic/datasets/ids-2017.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Guizhou Province Project of Innovation Talents Teams of Electrostatic and Electromagnetic Protection (no. [2017]5653) and Academician Liu Shanghe Fund of Electrostatic Protection Research (no. BOIMTLSHJD20161004).

References

[1] A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2017.

[2] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[3] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.

[4] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, "A comparison of intrusion detection by K-means and fuzzy C-means clustering algorithm over the NSL-KDD dataset," in *Proceedings of the 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, Chennai, India, 2017.

[5] I. M. Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ann classifier," *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.

[6] Y. Chuan-Long, Z. Yue-Fei, F. Jin-Long et al., "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[7] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, Article ID 112963, 2019.

[8] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren, "A novel multimodal-sequential approach based on multi-view features for network intrusion detection," *IEEE Access*, vol. 7, pp. 183207–183221, 2019.

[9] P. Sun, P. Liu, Q. Li et al., "DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, Article ID 8890306, 11 pages, 2020.

[10] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753–762, 2013.

[11] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: adaptively detecting anomalies over unlabeled audit data streams in computer networks," *Knowledge-Based Systems*, vol. 70, pp. 103–117, 2014.

[12] H. Yao, Q. Wang, L. Wang, P. Zhang, M. Li, and Y. Liu, "An intrusion detection framework based on hybrid multi-level data mining," *International Journal of Parallel Programming*, vol. 47, no. 4, pp. 740–758, 2019.

[13] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah, and T. Huang, "A real-time and ubiquitous network attack detection based on deep belief network and support vector machine," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 790–799, 2020.

[14] E. Min, J. Long, Q. Liu et al., "Su-IDS: a semi-supervised and unsupervised framework for network intrusion detection," in *Proceedings of the International Conference on Cloud Computing and Security*, pp. 322–334, Springer, Cham, Switzerland, 2018.

[15] M. R. Karbir, R. Onik, and T. Samad, "A network intrusion detection framework based on bayesian network using wrapper approach," *International Journal of Computer Applications*, vol. 166, no. 4, pp. 975–8887, 2017.

[16] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Computers & Security*, vol. 86, pp. 53–62, 2019.

[17] A. Ahmim, L. Maglaras, M. A. Ferrag et al., "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 228–223, IEEE, Santorini, Greece, 2019.

[18] V. Kumar, V. Choudhary, V. Sahrawat et al., "Detecting intrusions and attacks in the network traffic using anomaly based techniques," in *Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 554–560, IEEE, Coimbatore, India, 2020.

[19] A. S. Qureshi, A. Khan, N. Shamim, and M. H. Durad, "Intrusion detection using deep sparse auto-encoder and self-taught learning," *Neural Computing and Applications*, vol. 32, no. 8, pp. 3135–3147, 2020.

[20] S. Nathan, T. N. Ngoc, V. D. Phai et al., "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[21] A. Y. Javaid, Q. Niyaz, W. Sun et al., "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th Eai International Conference on Bio-inspired Information & Communications Technologies, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*, pp. 21–26, Nairobi, Kenya, 2016.

[22] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[23] R. Abdulhammed, H. Musafer, A. Alessa et al., "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, 2019.

[24] G. C. Fernández and S. Xu, "A case study on using deep learning for network intrusion detection," in *Proceedings of the MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pp. 1–6, IEEE, Norfolk, VA, USA, 2019.

[25] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," *The Journal of Supercomputing*, vol. 76, no. 9, pp. 6690–6719, 2020.

[26] P. Madani and N. Vlajic, "Robustness of deep autoencoder in intrusion detection under adversarial contamination," in *Proceedings of the 5th Annual Symposium and Bootcamp*, pp. 1–8, ACM, New York, NY, USA, 2018.

[27] E. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units(elus)," 2016, https://arxiv.org/abs/1511.07289.

[28] S. Ustebay, Z. Turgut, and M. A. Aydin, "Cyber attack detection by using neural network approaches: shallow neural network, deep neural network and autoencoder," in *Proceedings of the 2019 International Conference on Computer Networks*, pp. 144–155, Spinger, Cham, Switzerland, 2019.

[29] L. Breiman, J. Friedman, J. Charles et al., *Classification and Regression Trees*, Chapman and Hall/CRC, London, UK, 1984.

[30] KDDCup1999: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[31] M. Tavallaee, E. Bagheri, W. Lu et al., "A detailed Analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, Ottawa, Canada, 2009.

[32] I. Sharafaldin and A. Ali, "Ghorbani toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, Funchal, Portugal, 2018.

[33] M. Patel, B. Naughton, C. Chan et al., "Mobile-edge computing introductory technical white paper: mobile-edge computing(MEC) industry initiative," 2014.

[34] M. Eskandari, Z. H. Janjua, M. Vecchio et al., "Passban IDS: an intelligent anomaly-based intrusion detection system for IoT Edge devices," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020.

[35] Y. Zhang, W. Jia, and K. Wang, "An edge IDS based on biological immune principles for dynamic threat detection," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8811035, 15 pages, 2020.

[36] P. A. A. Resende and A. C. Drummond, "A survey of random forest based Methods for intrusion detection systems," *ACM Computing Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[37] S. Raponi, M. Caprolu, and R. Di Pietro, "Intrusion detection at the network edge: solutions, limitations, and future directions," in *Proceedings of the International Conference on Edge Computing*, Springer, Rome, Italy, 2019.