Trusted Encrypted Traffic Intrusion Detection Method Based on Federated Learning and Autoencoder

Wang Zixuan[1], Miao Cheng[1], Xu Yuhua[1], Li Zeyi[2], Sun Zhixin[1,*], Wang Pan[1]

[1] School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing 210003, China
[2] School of Computer Science, Nanjing University of Posts & Telecommunications, Nanjing 210003, China
[*] The corresponding author, email: sunzx@njupt.edu.cn

Cite as: Z. Wang, C. Miao, *et al*., "Trusted encrypted traffic intrusion detection method based on federated learning and autoencoder," *China Communications*, 2024, vol. 21, no. 8, pp. 211-235. **DOI**: 10.23919/JCC.ja.2022-0392

Abstract: With the rapid development of the Internet, network security and data privacy are increasingly valued. Although classical Network Intrusion Detection System (NIDS) based on Deep Learning (DL) models can provide good detection accuracy, but collecting samples for centralized training brings the huge risk of data privacy leakage. Furthermore, the training of supervised deep learning models requires a large number of labeled samples, which is usually cumbersome. The "black-box" problem also makes the DL models of NIDS untrustworthy. In this paper, we propose a trusted Federated Learning (FL) Traffic IDS method called FL-TIDS to address the above-mentioned problems. In FL-TIDS, we design an unsupervised intrusion detection model based on autoencoders that alleviates the reliance on marked samples. At the same time, we use FL for model training to protect data privacy. In addition, we design an improved SHAP interpretable method based on chi-square test to perform interpretable analysis of the trained model. We conducted several experiments to evaluate the proposed FL-TIDS. We first determine experimentally the structure and the number of neurons of the unsupervised AE model. Secondly, we evaluated the proposed method using the UNSW-NB15 and CICIDS2017 datasets. The experimental results show that the unsupervised AE model has better performance than the other 7 intrusion detection models in terms of precision, recall and f1-score. Then, federated learning is used to train the intrusion detection model. The experimental results indicate that the model is more accurate than the local learning model. Finally, we use an improved SHAP explainability method based on Chi-square test to analyze the explainability. The analysis results show that the identification characteristics of the model are consistent with the attack characteristics, and the model is reliable.

Keywords: autoencoder; federated learning; intrusion detection; model interpretation; unsupervised learning

Received: Jun. 12, 2022
Revised: Jan. 17, 2023
Editor: Liu Erwu

I. INTRODUCTION

With the rapid development of the Internet and information technology, the network has brought people a convenient life while exposing many security issues. With the frequency of attacks increasing and the scope of impact expanding, various cyber attacks occur. How to effectively prevent the damage caused by network security problems is becoming increasingly important. Intrusion Detection System (IDS) [1] is a proactive security protection technology that can effectively sense network attacks and provide security managers with response decisions through real-time network monitoring. By deploying intrusion detection models to edge devices, such as switches and routers,

computing resources can be effectively integrated to enhance identification efficiency, making applications such as dynamic access control, legitimate interception, and attack blocking possible. [2]

The evolution of traditional network intrusion detection methods has gone through three phases: port-based, net load feature-based, and flow statistics feature-based. However, due to the emergence of disguised port technologies, encryption techniques, and the high maintenance cost of feature design, these methods cannot adapt to the endless and dynamically changing network attack traffic. [3, 4]

In recent years, researchers have introduced Deep Learning (DL) into the study of intrusion detection. In deep learning, features are no longer artificial designed. The model has a strong generalization ability by training multiple nonlinear networks to extract complex structural features automatically. Therefore, deep learning is undoubtedly a very good choice, and many scholars have started to experiment with various different deep learning methods for detecting encrypted traffic, and have made some progress, especially in the detection accuracy and other indicators have improved significantly. [5–7]

However, traditional deep learning-based intrusion detection systems, which transmit data collected by edge devices to cloud servers for centralized training [8], suffer from the following problems. (1) Data privacy issues. With the development of big data and the enhancement of users' security and privacy awareness, there is a risk of privacy leakage in collecting and transferring data to cloud servers [9]. (2) Incomplete data distribution. The traffic collected by a single organization can hardly reflect the data distribution of the whole network. However, it is difficult to fully share data among enterprises [10] due to certain commercial factors [11], which cause serious data silos. These problems directly lead to the lack of sufficient training data for the model and affect the accuracy and reliability.

Google first proposed the theory of Federated Learning (FL) [12] in 2017. As a kind of distributed learning, federated learning is growing rapidly because of its unique training method: "data does not leave the device and model parameters are aggregated," which can effectively protect data privacy.

1.1 Motivation

As a bridge between LAN and WAN, network edge devices are well suited for traffic collection and deploying intrusion detection tasks using federation learning. Although federated learning is favored for its privacy-preserving advantages, the following challenges remain in practical applications due to the complex and frequently updated attack traffic and many edge devices with weak computing power:

- **Lack of a federation learning-based end-to-end intrusion detection framework:** Since work is deployed at the edge devices, a federated learning intrusion detection framework that incorporates traffic collection, data processing, model aggregation, and interpretable analysis is needed.
- **Data Dependency:** Since deep learning algorithms rely on large amounts of labeled data. The data labeling work requires expert experience and is time consuming and labor intensive. It is difficult for the edge nodes to perform the task of traffic labeling. Therefore, an unsupervised model is needed to achieve intrusion detection of traffic using unlabeled data.
- **Model Interpretation:** Deep learning models are nonlinear, with many parameters and high complexity, just like black boxes, so it is hard to interpret the model and verify the reliability of the model.

1.2 Key Contribution

In this paper, we propose a Federated Learning Encrypted Traffic Intrusion Detection System(FL-TIDS) by applying a combination of unsupervised DL and FL to edge devices. We conducted experiments on the CICIDS2017 and UNSW-NB15 datasets. The main contributions are as follows:

1. Combining the characteristics of Federated Learning and network topology, an end-to-end federation learning intrusion detection architecture based on edge devices is designed, including data processing, model aggregation, and interpretable analysis modules.
2. An auto-encoder-based intrusion detection model for network traffic is proposed. The model only needs unlabelled traffic data as training input, which is relatively easy to obtain. Greatly reduces

the workload of data labeling and dependency on labeled data. At the same time, the model has a simple structure and few parameters, which optimizes the recognition efficiency of edge terminals.

3. An improved SHAP interpretable method is proposed. The model is validated using the proposed method. The features learned by the model are visualized to verify the reliability of the model.
4. The newer CICIDS2017 and UNSW-NB15 datasets were selected to validate our model. On the selected dataset, Precision, Recall, F1-score, and AUC are significantly improved.

The rest of the paper is organized as follows: Chapter II presents the related work, and Chapter III describes the proposed methodology. Chapter IV presents the experimental setup. Chapter V presents the experimental results and analysis. Finally, a summary of the whole paper is presented.

II. RELATED WORKS

2.1 Types of IDS

An intrusion detection system (IDS) is a hardware or software system used to track a network for unauthorized behavior or policy violations. An IDS records detailed information related to an intrusion, issues warnings, and takes necessary mitigating or corrective actions when an intrusion is detected. IDS can be classified as misuse-based and anomaly-based by detection techniques. [13]

1) Misuse-Based Intrusion Detection System (MIDS) [14]: MIDS is also known as signature-based intrusion detection. MIDS matches incoming network traffic with existing signatures and sends an alert message to report abnormal behavior once it hits. MIDS designs one signature for each attack and has a high detection accuracy. There are three main types:

- **State Modeling [15]:** State modeling is the encoding of an attack into many different states in a finite automaton. Detecting attacks by looking at the traffic profile.
- **String Matching [16]:** String matching is a knowledge acquisition process that finds the attack signature by string pattern matching to determine whether there is an intrusion.
- **Expert System [17]:** Expert System is a system that classifies audit data according to a set of rules used to describe known attack scenarios. Different attributes and categories are first identified from the training data. Then, a set of classification rules or procedures are derived. Finally, the audit data is classified.

Most of the early IDSs used misuse-based techniques to detect intrusions. However, misuse-based intrusion detection systems are highly dependent on the existing signature knowledge, difficult to detect unknown attacks, and cannot adapt to new intelligent attack behaviors [18]. Therefore, anomaly-based intrusion detection is currently the focus of research and development by scholars.

2) Anomaly-Based Intrusion Detection System (AIDS): AIDS [19] is a system for the detection of anomalous behavior. These methods record and analyze the statistical characteristics of normal activity, capture network traffic and create records that represent its random behavior. When the characteristics of the behavior to be detected deviate significantly from the normal behavior, an alert message is issued. From the technical point of view, anomaly-based IDS includes machine learning-based, deep learning-based, and federated learning-based methods. The model proposed in this paper belongs to the anomaly-based IDS, and its related progress will be introduced in the next section.

2.2 Anomaly-Based Intrusion Detection Methods

1) Machine learning based intrusion detection methods: [20] The evolution of traditional network traffic intrusion detection methods has gone through three phases, namely port-based, net load feature-based, and flow statistics feature-based. However, the emergence of port masquerading, random ports, and tunneling techniques have rapidly rendered port-based intrusion detection methods ineffective [21]; net load feature-based, for example, deep packet inspection (DPI) techniques are unable to cope with the detection of encrypted traffic due to the need to match net load features [22]; flow statistical feature-based methods use machine learning algorithms to detect encrypted traffic, including Naive Bayes, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbor

(KNN), etc. are widely used due to their good generalization [23].

In the paper [24], the input features for network routing detection are obtained through PCA feature extraction and discretization. The processed dataset is then classified using Naive Bayes. The experimental results on the KDD CUP99 dataset show that the method can effectively detect Trojan horse attacks, fake message attacks, denial of service attacks, and unauthorized access attacks by remote users, with a detection rate (DR) of 87%˜97%. However, the evaluation metrics of this paper are too simple. Only the detection rate and false alarm are provided. The paper [25] designed an intrusion detection model based on decision trees and integrated three different classifiers. The highest accuracy verified on the CICIDS2017 dataset is 96.665%, and the lowest False Alarm Rate (FAR) is 1.145%. However, the network traffic is complex and constantly changing, and the intrusion detection model lacks an adaptive mechanism. The paper [26] performed feature processing by Z-score normalized and compressed sampling method then combined with SVM to process the data for classification. The experimental results on KDD CUP99 data show that it can effectively detect denial-of-service attacks, probe attacks. In the paper [27], an improved genetic algorithm optimized support vector machine approach for intrusion detection was proposed, and an adaptation function based on classification accuracy, false alarm rate, and dimensionality of data features was designed. Although these methods achieve good recognition results, they are too time-consuming to detect traffic in practice. The paper [28] uses PCA for feature extraction, combined with fuzzy techniques to obtain the degree to which the sample objects belong to each category. Then KNN is used to classify the attack categories. However, the accuracy of this algorithm gradually decreases as the number of data increases.

Although machine learning methods are fast in convergence and can detect anomalies despite incomplete data. However, these methods still have some limitations: (1) Machine learning methods have limited ability to express the nonlinear features of traffic and are prone to fall into local optimum. With the increase of data in the network and the increase of network bandwidth, the complexity of data and the diversity of features are also increasing, and it is difficult for machine learning methods to achieve the purpose of analysis and prediction. (2) Feature extraction and selection rely heavily on domain experts. Feature design, extraction, and selection are all manual, which is time-consuming and labor-intensive. The quality of features often directly affects the effectiveness of machine learning. (3) Traffic characteristics are easily outdated and require continuous tracking and frequent updates.

2) Deep learning based intrusion detection methods: [29] Unlike traditional machine learning, deep learning methods can learn intrinsic patterns and hierarchical representations of sample data. By building multiple nonlinear network structures, it can adapt to the requirements of higher dimensional learning and prediction. The model can be quickly established according to the identification target, and the detection efficiency is higher, which is very promising in solving the problem of intrusion detection [30].

Multi-layer Perceptron (MLP) is a simple deep learning model. In the paper [31, 32], an MLP is designed to automatically extract the nonlinear features of encrypted traffic and combine it with a softmax classifier to achieve the identification of encrypted traffic. The experimental results show a high recognition accuracy. However, the MLP, as a fully connected model, has many parameters, is complicated to train, and is prone to overfitting.

CNN is a feed-forward neural network that includes convolutional computation and has a deep structure, one of the most representative algorithms for deep learning. Paper [33, 34] uses 2D-CNN and 1D-CNN with both flows and sessions as model inputs to identity traffic. Experiments were conducted on the ISCX dataset, and the experimental results showed good Precision, Recall, and F1-score. The paper [35] designed a traffic classification algorithm based on the CNN model of LeNet-5, which uses the Min-Max Normalization (MMN) approach to pre-process traffic data. Validation results on the Morre public dataset show that the proposed MMN-CNN approach improves traffic classification accuracy and reduces the time used for classification.

However, although CNN has excellent feature learning ability, it cannot extract temporal features. Some scholars combined temporal models such as RNN/LSTM with CNN to extract spatial and temporal features of encrypted traffic to improve recognition performance [36, 37]. The experimental results show that

the classification performance using the combination of CNN+RNN outperforms that of CNN only.

Deep learning methods can automatically extract higher-level, complex patterns and thus achieve good recognition performance. However, the training process of these deep learning models requires a large amount of labeled traffic samples. Whether collecting encrypted flow samples or labeling, it is a time - consuming work.

In addition, the generalization ability of models obtained by training based only on labeled data is often weak and difficult to adapt to changing encrypted applications. In contrast, unlabeled data are abundant and easy to obtain. Therefore, researchers actively explore unsupervised encrypted traffic-based intrusion detection methods and mechanisms. The basic idea is to select a suitable unsupervised model for automatic feature extraction on a large amount of "cheap" unlabeled encrypted traffic data, which are easy to obtain. When the model is trained, anomalous traffic can be identified by comparing reconstruction losses.

Generative Adversarial Network (GAN) is one of the most promising unsupervised methods in recent years [38]. The mutual game between the generative and discriminative models can generate high-quality samples to alleviate the dependence of the models on the samples. An efficient GAN-based model was proposed in the paper [39]. The loss function of this model has been specially designed. With its powerful generation capability, it can identify abnormal traffic by learning the distribution of normal traffic. The paper [40] proposed Cycle-GAN. The authors converted the ADFA-LD dataset into images and then used the generated data together with the original data to train the anomaly detection model. The experimental results show that the GAN can improve the accuracy of the anomaly detection model. Although GAN has powerful data generation capability, it is difficult to converge due to the problem of gradient disappearance, which is hard to deploy on edge devices.

As an unsupervised algorithm, AutoEncoder (AE) [41] is widely used in intrusion detection. The paper [42] proposes an autoencoder combining a stacked sparse denoising and softmax classifier, which can effectively detect denial of service attacks and probe attacks. Paper [43] proposes a deep autoencoder-based intrusion detection method with layer-by-layer greedy training to avoid overfitting. The evaluation results on KDD CUP99 show that binary classification and multi-classification have high accuracy. AE has a simple structure and can represent linear and nonlinear transformation, which is suitable for deployment on edge devices for intrusion detection.

However, both AE and other deep learning detection models have the following problems: (1) Most of the data used by deep learning models tend to come from a single institution. The amount of labeled data generated by a single institution is extremely limited. It cannot reflect the data distribution of network traffic comprehensively. At the same time, it is difficult to fully share data among enterprises due to certain commercial factors, and there are Isolated Data Islands. (2) With the development of big data, data privacy and security are attracting more and more public attention, and the mode of collecting data for centralized training may cause data privacy leakage.

2.3 Intrusion Detection Based on Federated Learning

Federated learning, as an effective solution to tackle the problem of Isolated Data Island, can compute multi-party models while ensuring the privacy and security of user data. [44] Therefore, combining federated learning and IDS helps to reduce the computational burden of the central processing server, protect data privacy, and expands the distribution of data to improve the accuracy of IDS. [45, 46]

The paper [47] proposes a GRU-based federated learning network intrusion detection method. This approach allows multiple ISPs or other organizations to perform joint deep learning training while retaining local data. The paper [48] proposes a joint learning-based intrusion detection approach, called MV-FLID, which is trained on multiple views of IoT network data in a decentralized format to detect, classify, and defend against attacks. The pape [49] proposes a Federated Learning-Based Attention Gated Recurrent Unit (FedAGRU). By avoiding uploading unimportant updates to the server, FedAGRU can significantly reduce the communication overhead while guaranteeing learning convergence. The paper [50] proposes FedACNN, an intelligent intrusion detection mechanism, which assists CNN in the intrusion detection task through a federation learning mechanism. To alleviate the communication latency limitation of fed-

erated learning, the authors innovatively integrate an attention mechanism. FedACNN can achieve the expected accuracy with a 50% reduction in communication time. The paper [51] builds a joint model using MLP and AE. The authors conducted comparative experiments using centralized, distributed, and federated learning architectures. The results show the superiority of the federated learning architecture, which guarantees higher data utilization. Existing research has combined federation learning and intrusion detection and made some progress.

However, these models are trained based on a supervised approach, and the papers do not consider how to label traffic at edge devices, making it difficult to apply in practice. At the same time, since the federated learning is trained at the edge device, the quality of the terminal data varies, and the lack of validation of the model reliability leads to unreliable accuracy of the model.

To address these issues, this paper combines autoencoder and federated learning to achieve intrusion detection of traffic while protecting data privacy. With an unsupervised approach, the reliance of deep learning on labeled data is reduced. The reliability of the model is verified using the improved SHAP interpretable method. By designing data collection, data processing, model aggregation, and interpretable analysis processes, an end-to-end Federated Learning Encrypted Traffic Intrusion Detection System (FL-TIDS) based on edge devices is developed.

III. PROPOSED METHOD

3.1 Overall Architecture

As shown in Figure 1, since the edge devices serve as a bridge connecting the LAN and WAN, by deploying intrusion detection-based models in the edge devices, model training can be achieved using the traffic in the LAN while performing intrusion detection on the traffic in the WAN. The edge devices of each company are used as the sub-nodes of federated learning. Since different companies generate different traffic application data, horizontal federation learning is composed to expand the breadth of traffic data and form a complete data distribution. Through the cloud management platform, the initial model is firstly distributed, then parameter aggregation and model interpretability are performed, and finally, the intrusion detection model is obtained. The whole FL-TIDS is divided into six steps:

Step 1: Design edge device-based network packet collection method for efficient packet collection.

Step 2: By using pre-processing data methods, bi-directional flow is extracted from the collected packets, and Flow Features are calculated to form a Flow Attributes Matrix (FAM) as model input.

Step 3: Design the initial model of unsupervised intrusion detection based on SAE in the cloud platform and design the model input, model structure, and loss function according to the detection task. The model is trained using unlabeled samples to achieve intrusion detection by anomaly discrimination.

Step 4: Distribute the designed initial model to each edge device. By embedding the code that implements step one step two into the edge devices, the data in the LAN is collected to form the data set for training. Federated training of the model is performed under the scheduling of the cloud server. When model training is performed on the edge devices, only the model weights and parameters are uploaded and aggregated, and the data remains in the LAN, ensuring data security and privacy.

Step 5: When federal learning is completed, the global model is subjected to interpretable analysis to assess the reliability of the model and form the final model.

Step 6: The cloud server distributes the final model to each edge device, and the edge devices collect the traffic accessing the LAN. The loss of access traffic is calculated using the intrusion detection model, and traffic detection is achieved by discriminating the loss. The specific implementation of each step will be elaborated on in later sections.

The internal traffic classification and model update process of the sub-node is shown in Figure 2. In the LAN, the sub-node receives traffic from different devices, copies the original data packets, and then preprocesses the data packets. After forming the flow feature, use the acquired initial model for local training. After training, the sub-node uploads weights and parameters to the cloud server. The cloud server aggregates the model weights of multiple sub-nodes, updates the parameters and gradients, and then returns the model to the sub-nodes. The edge device can use the latest model to identify the traffic and upload the

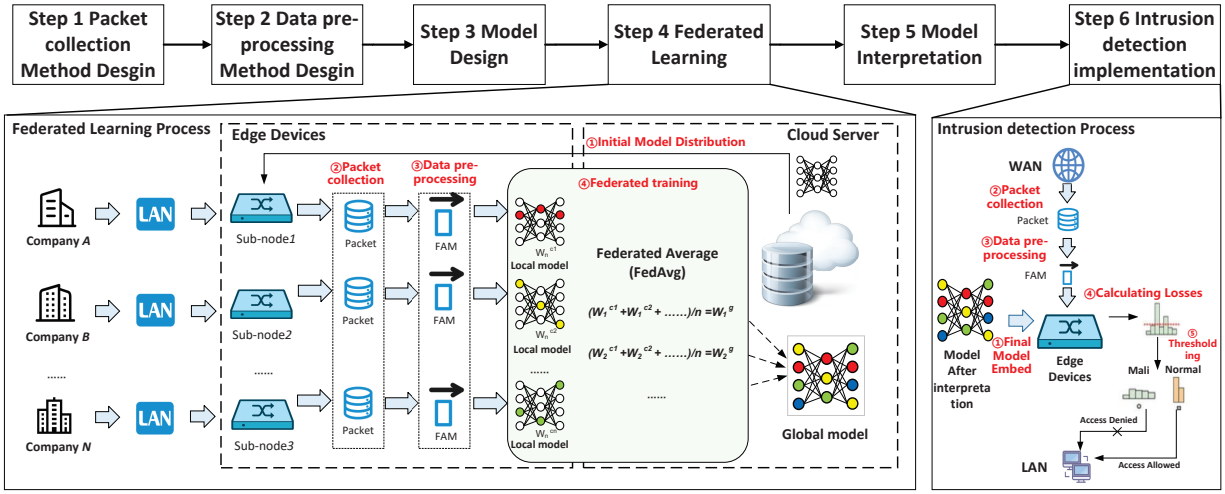


Figure 1. *Overall architecture.*

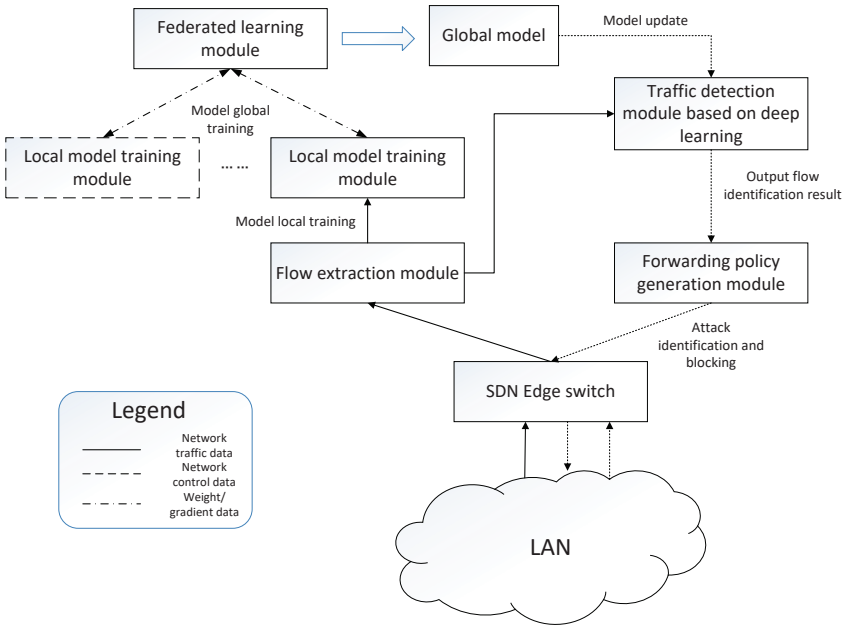


Figure 2. *Federated learning encrypted traffic identification architecture based on edge device (data flow diagram).*

identified threat ID as part of the custom flow information. Then the control platform generates a flow table according to the application protocol and sends it to the edge device. Finally, the edge device realizes the blocking of malicious traffic according to the flow table.

The internal processing sequence of the edge devices is shown in Figure 3. In the training phase, the edge device receives the traffic coming from the LAN. The original packets are first replicated, and then the replicated packets are pre-processed to form the flow features. Then train the acquired initial model by using the flow features locally. After the training, the weights and parameters are uploaded to the Federated Learning Platform. The federated learning platform updates the parameters after integrating the model weights from multiple edge devices, and distributes the final model to the edge devices. In the detection stage, by replicating the access traffic of WAN, after pre-processing and extracting flow characteristics, the flow characteristics are input into the detection model to calculate the loss, and the alarm of the attack is realized by loss discrimination.

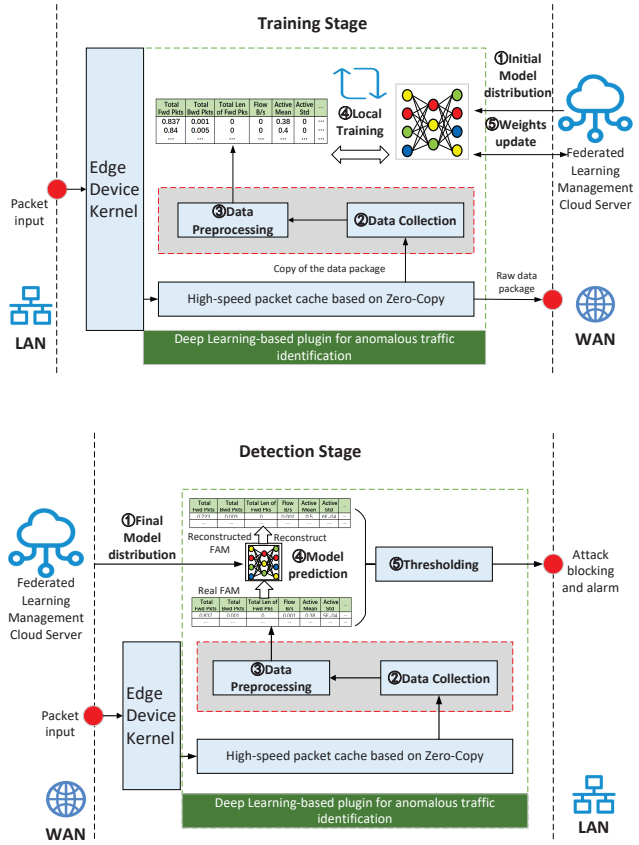


Figure 3. *Edge device internal processes.*

3.2 Packet Collection Method

The data acquisition module is based on the PF packet implementation. First, initialize a Ring Buffer, then add the hook function to the NIC driver. When a network packet passes through the NIC, use the hook function to copy the network packet to Ring Buffer. Finally, get Packets in User State by using address mapping. Using this method has the following advantages: 1) Data replication after the implementation of bypass processing does not affect the original data transmission. 2) The ring buffer will not block because the space is full. 3) If the data pre-processing module reaches a performance bottleneck and does not fetch the data in time. The lost packet can be viewed through the standard statistical interface. 4) The ring buffer mechanism is most efficient in data bypass processing. These advantages ensure efficient traffic collection without affecting the normal forwarding of packets.

3.3 Data Pre-processing Method

We first obtain the Packet of the application interactions using the method described above. Then we extract the flows of the packets using 5-tuples. Finally, Packet-level features, Flow-level features, and statistical features are calculated.

3.3.1 Definition

Before introducing the data extraction method, we would like to describe the relevant mathematical definitions.

1) Packet: Packet is the smallest unit in the flow. The set P is the set of Packets, and M is the number of Packets in a flow.

$$P = \{p_1, p_2, p_3 \ldots\ldots p_{i-1}, p_i\}, 0 \leq \mathrm{i} \leq \mathrm{M}. \quad (1)$$

2) Session: The session is also called bi-directional flow, which refers to the up-flow and down-flow with the same source address, destination address, source port, destination port, and TCP/UDP protocol. The session is represented by the set S, and N is the number of all sessions that the application interacts with. Application traffic may contain multiple sessions, such as a game application that contains multiple sessions between the cell phone and the domain name server, the main game site, the advertising server, etc.

$$S = \{S_1, S_2, S_3 \ldots\ldots S_{i-1}, S_i\}, 0 \leq 1 \leq N. \quad (2)$$

3) Flow: The difference between Flow and Session is that flow is unidirectional. The set F represents flow, and K represents the number of flows in application interactions.

$$F = \{f_1, f_2, f_3 \ldots\ldots f_{i-1}, f_i\}, 0 \leq i \leq K. \quad (3)$$

4) Flow Features: According to the above definition, this paper computes Flow Features for Packet and Flow to form the feature set

$$FF = \{ff_1, ff_2, ff_3 \ldots ff_j\}. \quad (4)$$

3.3.2 Pre-Processing Method

First, we extract the TCP and UDP packets from the obtained application interaction traffic to form a

packet set $P = \{p_1, p_2, p_3 \ldots\ldots p_{i-1}, p_i\}, 0 \leq \mathrm{i} \leq \mathrm{M}$. Then the Packets in the Packet set are organized using the five-tuple to form the Flow set $F = \{f_1, f_2, f_3 \ldots\ldots f_{i-1}, f_i\}$. When extracting, we use a five-tuple as the flow's token (key), which is stored in the hash table, along with the timestamp of the first Packet and the timestamp of the last Packet of the flow. If the duration of a Flow exceeds 120 seconds, truncate the flow and clear the information in the hash table. The subsequent packet will be considered as the new flow to compute the FF; The FIN mark of TCP is considered to be the end mark of the flow The features of the flow are output and the information in the hash table is cleared as soon as the FIN mark appears.

To extract Session, TCP Flow is analyzed. The packet in the TCP flow that is consistent with the src IP and dst IP of the SYN packet will be recorded as the forward flow, and the opposite will be recorded as the backward flow; Those Flow without SYN packets, the src IP and dst IP of the first Packet is used as a basis for judgment. The set containing forward and backward flows is formed for feature extraction $S_j = \{f_1, f_1', f_2, f_2' \ldots\}$.

We write the feature extraction code based on the feature calculation method in CICFLOWMETER [52] for edge-based devices. Extract features for each session's bi-directional flow to form a flow feature vector $FF = \{ff_1, ff_2, ff_3 \cdots, ff_j\}$. By definition, there are three levels, including packet-level features, flow-level features, and statistical features. Each flow feature vector FF is stitched into a flow attributes matrix FAM according to Packet, as shown in Table 1.

Table 1. *Flow attribute matrix (FAM).*

Packet Level				
Total Fwd Pkts	**Total Bwd Pkts**	**Total Len of Fwd Pks**	**Total Len of Bwd Pkts**	...
32	16	6448	1152	...
32	16	6448	5056	...
545	0	0	0	...
...	...	...	...	...
Flow-level				
Flow Duration	**Flow B/s**	**Flow IAT Max**	**Flow IAT Min**	...
112740690	67.411331	16400000	3	...
112740560	102.03959	16400000	2	...
113757377	0	20800000	0	...
...	...	...	...	...
Statistics				
Mean (Pkt-len)	**Average (Pkt-size)**	**Active Mean**	**Active Std**	...
163.3265306	166.72917	359.4285714	11.99801571	...
243	248.0625	320.2857143	15.74499165	...
0	0	9361828.6	7324645.883	...
...	...	...	...	

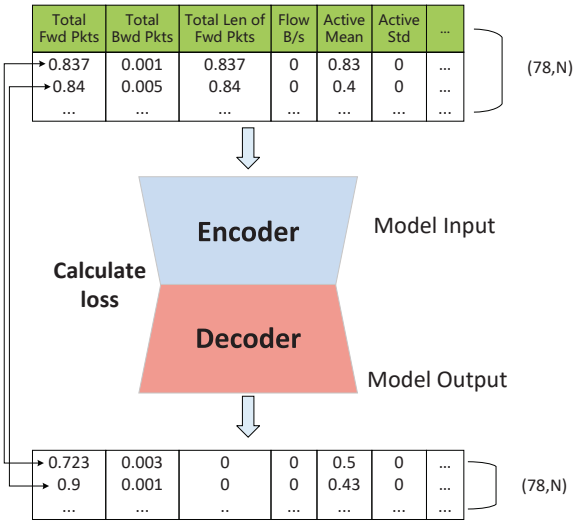


Figure 4. *Model input design.*

3.4 Model Design

3.4.1 Model Input Design

As shown in Figure 4, the FAM obtained in Section 3.3 is first subjected to data normalization and data normalization. Then the FAMs are cropped according to different lengths to form a set H containing multiple FAM subsets as the input to the model. $H = [FAM_1, FAM_2, FAM_3, \ldots]$, where the shape of each FAM subset is 78*N. During training, N is the batch size. During identification, N is set to 10000, which means when 10,000 flows are captured by the edge device, they are sent to the model for detection once. Intrusion detection is achieved by comparing the loss between each vector of each FAM subset.

3.4.2 Model Structure

In FL-TIDS, the sub-nodes use the same model structure and only change the weights of the neurons in the model by federated learning. Upon completion of federated learning, each sub-node can implement intrusion detection by using the latest global model. We analyzed the detection performance of the AE model in different structures and different neurons through experiments (EXP1). In order to keep the model lightweight, we use the model with the fewest number of neurons while ensuring accuracy. The final structure of the model is shown in the Figure 5, the model

consists of an encoder, a hidden layer, and a decoder, with a total of 8 layers of fully connected networks. The encoder consists of 3 fully connected layers with 256,128,64 neurons, and the decoder also consists of 3 fully connected layers with 64,128,256 neurons, respectively. The encoder and decoder are connected by a hidden layer containing 32 neurons, and finally, a fully connected layer with 78 neurons is used to reduce the vector dimension.

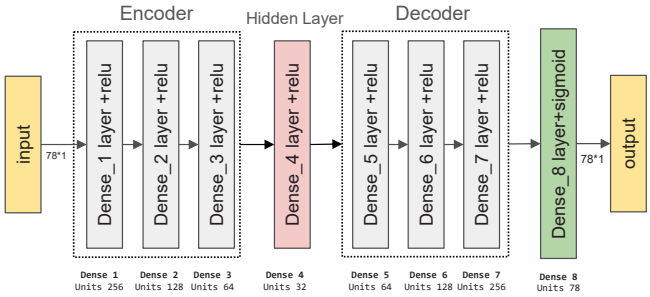


Figure 5. *Model structure.*

3.4.3 Loss Function

We use the mean squared error (MSE) to evaluate the difference between the predicted (reduced) and true values of the model, as shown in Equation (5):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{f}_i - f_i\right)^2, \tag{5}$$

$\hat{f}_i$ denotes the i-th position of each FF vector in each FAM submatrix, and f_i is its corresponding predicted value.

3.5 Federated Learning

As shown in Figure 6, the whole training process consists of two parts: global training and local training. In each training round, each client independently computes an update of the current model based on its local data and transmits this update to the central server, where the client updates are aggregated to compute a new global model.

Local training consists of four steps: 1) First, receive the initial model from the central server. 2) Then, multiple FAMs are collected in the trusted LAN to form a set $H = [FAM_1, FAM_2, FAM_3, \ldots]$, as the training dataset. 3) Take one FAM from the set H

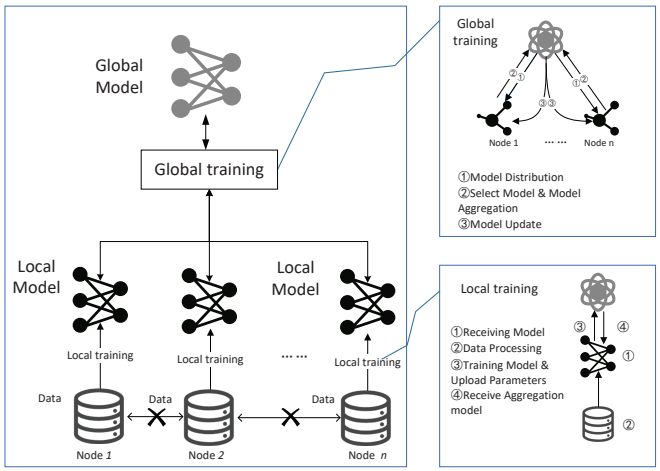


Figure 6. *Federated learning process.*

for each training, and train the model using the gradient descent algorithm until all the FAMs in the set H are trained 4) Upload the weights and biases of the trained model to the central server. Receive the updated weights bias returned by the central server. Repeat steps 3)-4) several times until the global node is judged to be globally converged, and then stop.

Global training consists of 3 steps: 1) Design intrusion detection model and distributes it to each node. 2) Collect model parameters, and losses for local training at each node. 3) The parameters and losses of each node are aggregated and resent to each node. Repeat steps 2)-3) several times until the global loss reaches the set convergence threshold. Considering that the data volumes of different sub-nodes are not the same when the model is globally aggregated, the ratio of the data volumes of each node and all nodes are used as the weight for aggregation.

The specific training process is shown in Algorithm 1. where k represents k nodes and ω_t represents the global model parameters at t iterations. ω_t^k represents the model parameters of the kth node at iteration t.

Algorithm 1. *The FL-TIDS method.*

Require: The set H containing multiple FAM subsets.
Input: H is the set of training FAMs collected by each node.
$\boldsymbol{H = [FAM_1, FAM_2, FAM_3, \ldots]}$
Output:global model $\boldsymbol{M_{t+1}}$.
Set hyper-parameters: Set the activation function. Global epoch $\boldsymbol{e_g}$.Local epoch $\boldsymbol{e_l}$. Number of nodes **k**. Learning rate $\boldsymbol{\eta}$, Batch size $\boldsymbol{\tau}$.

for each Global Round **t =1,2,3,...** to e_g **do**
  for each node **k = 1,2,3,...** **do**
    if global model exists **then**
      Load global model;
    else
      Get initial model from central server.
    end if
    while $\mathbf{FAM_i}$ in **H do**
      Feed $\mathbf{FAM_i}$ as input to the SAE model.
      Input: Feature vectors in FAM_i. $F = \{f_1, f_2, \cdots, f_n\}$.
      Output: $\widehat{F} = \left\{\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n\right\}$, Updated weights ω_t^k,b_t^k.
      for local epoch $\mathbf{e_l}$ **do**
        Using the formula $\boldsymbol{Z = \sum WF + b}$ to calculate the Forward Propagation, Z is the weight of the neuron of SAE model;
        Use the formula $\boldsymbol{ReLU(Z) = \max[0, Z]}$ to activate;
        Calculate and minimize losses: $\mathbf{min}\left(\frac{1}{n}\sum_{i=1}^{\tau}\left(\boldsymbol{F_i - \hat{F}_i}\right)^{\mathbf{2}}\right)$;
        Back propagation to update the weights: $\omega_t^k = \omega_t^k - \eta\frac{\partial}{\partial}\mathscr{L}$;$b_t^k = b_t^k - b\frac{\partial}{\partial}\mathscr{L}$
      end for
    end while
    Send local trained model to central server.
  end for(Finish Local Training)
  if update model parameters **then**
    Average the model parameters: $\omega_{t+1} = \frac{1}{k}\sum_{k=1}^{k}\frac{d_k}{d}\omega_t^k$; $b_{t+1} = \frac{1}{k}\sum_{k=1}^{k}\frac{d_k}{d}b_t^k$;
    Load the new model parameters ω_{t+1},b_{t+1} and get the new global model M_{t+1}.
  end if
  Send the new global model M_{t+1} to node.
end for(Finish Global Training)
return: M_{t+1};

3.6 An Improved SHAP Interpretable Method Based on Chi-Square Test

SHAP was proposed by Lundberg and Lee in 2017 [53] to explain various hard-to-understand black box models. The SHAP values are mainly used to quantify the contribution made by each feature to the model prediction. The basic design idea is to first calculate the marginal contribution of a feature when it is added to the model, then calculate the different marginal contributions of the feature in the sequence of all features, and finally calculate the mean value of all marginal contributions of the feature to obtain the SHAP value of the feature. The result of encrypted traffic identification can be seen as a result of the cooperation of the features with each other.

Suppose the ith sample of sample set M is x_i, the jth feature of sample x_i is x_{ij} and the predicted value of the model for sample x_i is y_i:

where y_{base} is the mean of all sample evaluated values and $f\left(x_{ij}\right)$ is the shapely value of x_{ij}

$$f\left(x_{ij}\right) = \sum_{S\subseteq\{x_1,\cdots,x_p\}\setminus\{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(v_x\left(S\cup\{x_j\}\right) - v_x(S)\right), \tag{6}$$

where p is the number of features of the sample, $\{x_1, \cdots, x_p\} \setminus \{x_j\}$ is the set of all possible input features excluding $\{x_j\}$, and $v_x(S)$ is the prediction result of the feature subset S.

The contribution of feature j is obtained by summing the shapely mean of feature j for all samples with the following formula:

$$f\left(x_j\right) = \sum_{i=1}^{M} f\left(x_{ij}\right). \tag{7}$$

The traditional SHAP interpretation of model classification results does not consider the case where multiple features act together. In cases where multiple features jointly affect the classification result, the shapely values of each feature are averaged out by using SHAP. For example, in the process of encrypted traffic detection, when the flow rate and size of packets work together, their final SHAP values are very close, and the contribution of each feature is averaged out, which is not conducive to the mining of the main features. To solve the problem of averaging the contribution of features in the SHAP calculation, this paper uses the chi-square test to calculate the correlation between each feature variable and the results to enhance the weight of features with a larger contribution to sample classification in the SHAP method, increasing the variability between the SHAP values of each feature, which helps to distinguish the importance of

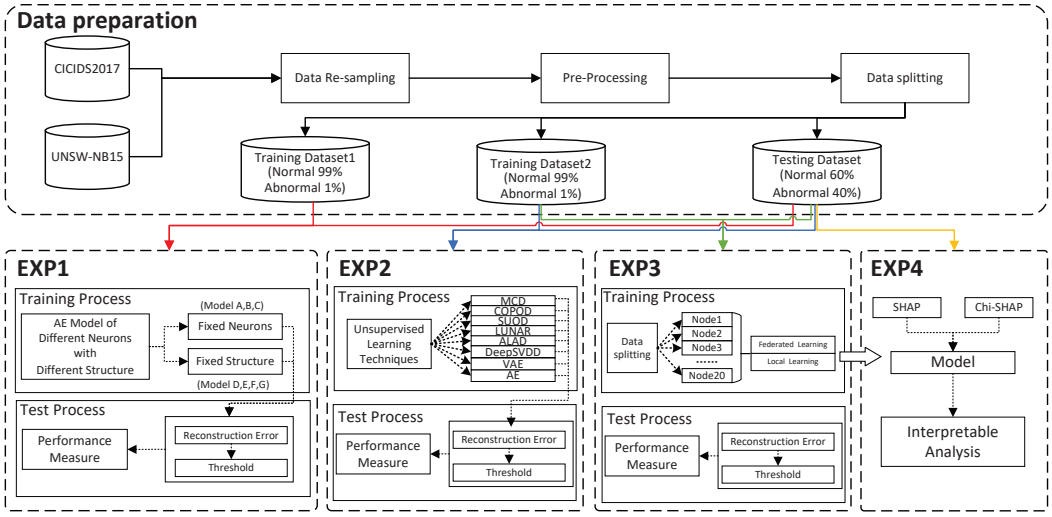


Figure 7. *Experimental design.*

features and unearth the features learned by the model.

The chi-square test is a hypothesis testing method that can be used for the analysis of the association between two variables. Its principle can be expressed as:

$$x^2 = \sum \frac{(A-T)^2}{T}, \tag{8}$$

where A is the actual frequency, T is the theoretical frequency, and x^2 is the chi-square test value. The chi-square test first assumes that the two variables are independent of each other, and under the condition that this assumption holds, the theoretical frequencies of each cell in the column table of variables are calculated, as well as the actual frequencies of these two variables; Compare the difference between the actual frequency and the theoretical frequency, the larger the difference, the greater the difference between the actual and the hypothesis, and the smaller the difference, the smaller the difference between the actual and the hypothesis. This difference is expressed as a chi-square value test, i.e., if the chi-square test result is insignificant, the original hypothesis is not valid, and the two variables are independent of each other; if the chi-square test result is significant, the original hypothesis is valid, and the two variables are correlated. If there are multiple influence characteristics in a sample, the chi-square test is used to compare which influence characteristics are more favorable for sample delineation. In this paper, the chi-square test is transformed to some extent to make it more convenient to calculate the weights as the SHAP method, and the improved SHAP formula for the chi-square test method is as follows:

$$\lambda = \frac{x^2 - x_{\min}^2}{x_{\max}^2 - x_{\min}^2}. \tag{9}$$

$$f(x_j) = \sum_{i=1}^{M} \lambda f(x_{ij}). \tag{10}$$

The SHAP method, improved by using the chi-square test method, can increase the weights of the features of the sample classification in the case of the joint action of multiple feature factors. It is able to fully exploit the classification features learned by the model and thus verify the reliability of the model.

$$y_i = y_{\text{base}} + f(x_{i1}) + f(x_{i2}) + \cdots + f(x_{ij}). \tag{11}$$

IV. EXPERIMENT SETUP

We conducted experiments on two datasets, UNSW-NB15 and CICIDS2017, to verify the performance of the proposed method. In particular, we have concerns regarding four aspects: (1) The structure and number of neurons in the SAE model; (2) The performance of the unsupervised SAE model; (3) The performance of federal learning; (4) The validity of the proposed model. Therefore, we conducted four sets of experi-

Table 2. *Data resampling.*

UNSW-NB15					
Original Data			After Resamping		
Class Name	Total records	Ratio	Class Name	Total records	Ratio
BENIGN	2218764	87.35%	BENIGN	2000000	97.56%
Generic	215481	8.48%	Generic	10000	0.49%
Exploits	44525	1.75%	Exploits	10000	0.49%
Fuzzers	24246	0.95%	Fuzzers	10000	0.49%
DoS	16353	0.64%	DoS	10000	0.49%
Reconn-aissance	13987	0.55%	Reconn-aissance	10000	0.49%
Analysis	2677	0.11%	/	/	/
Backdoor	1795	0.07%	/	/	/
Shellcode	1511	0.06%	/	/	/
Backdoors	534	0.02%	/	/	/
Worms	174	0.01%	/	/	/
CICIDS2017					
Original Data			After Resamping		
Class Name	Total records	Ratio	Class Name	Total records	Ratio
BENIGN	2273097	80.30%	BENIGN	2000000	98.52%
PortScan	158930	5.61%	PortScan	10000	0.49%
DDoS	128027	4.52%	DDoS	10000	0.49%
DoS Hulk	231073	8.16%	Dos	10000	0.49%
DoS GoldenEye	10293	0.36%			
DoS Slowloris	5796	0.20%			
DoS Slowhttptest	5499	0.19%			
FTP-Patator	7938	0.28%	/	/	/
SSH-Patator	5897	0.21%	/	/	/
Bot	1966	0.07%	/	/	/
Web Attack Brute Force	1507	0.05%	/	/	/
Web Attack XSS	652	0.02%	/	/	/
Infiltration	36	0.00%	/	/	/
Web Attack Sql Injection	21	0.00%	/	/	/
Heartbleed	11	0.00%	/	/	/

ments, and Figure 7 illustrates the overall experimental process.

4.1 Data Preparation

4.1.1 Data Resampling

We first resampled the original dataset in order to discard categories with insufficient data and ensure a balanced experimental sample. The data resampling results are shown in Table 2. As part of CICIDS2017, we extracted 2,000,000 records from the BENIGN sample and 10,000 records from the PortScan and DDOS samples. As a result, we merged the four DOS attacks and extracted 10,000 records to create the resampling dataset. For the UNSW-NB15 dataset, we extracted 2,000,000 records from the BENIGN sample and 10,000 records from Generic, Exploits, Fuzzers, DoS, and Reconnaissance sample to create the resampling dataset.

Table 3. *Dataset after splitting.*

UNSW-NB15			
	Class Name	Total records	Ratio
Training Dataset 1	BENIGN	990000	99%
	Generic	2000	1%
	Exploits	2000	
	Fuzzers	2000	
	DoS	2000	
	Reconnaissance	2000	
Training Dataset 2	BENIGN	1980000	99%
	Generic	5000	1%
	Exploits	5000	
	Fuzzers	5000	
	DoS	5000	
	Reconnaissance	5000	
Testing Dataset	BENIGN	1500	60%
	Generic	1000	40%
	Exploits	1000	40%
	Fuzzers	1000	40%
	DoS	1000	40%
	Reconnaissance	1000	40%
CICIDS2017			
	Class Name	Total records	Ratio
Training Dataset 1	BENIGN	990000	99%
	PortScan	3300	1%
	DDoS	3300	
	DoS	3300	
Training Dataset 2	BENIGN	1980000	99%
	PortScan	6700	1%
	DDoS	6700	
	DoS	6700	
Testing Dataset	BENIGN	1500	60%
	PortScan	1000	40%
	DDoS	1000	40%
	DoS	1000	40%

4.1.2 Data Pre-processing

After data resampling, we pre-processed the two resampling datasets, mainly including feature cropping, data digitization and data normalization.

We first remove the network identifiers from the two resampling datasets in order to ensure the model's generalization performance. We then performed data digitization operations on the two resampling datasets. On the UNSW-NB15 reresampling dataset, data containing string formats such as (proto_map, state_map, service_map) were digitally encoded. On the CICIDS2017 resampling dataset, data containing INF and NAN were discarded. Finally, we performed min-max normalization on the two resamping datasets.

Table 4. *Exp1: Model structure.*

	InPut layer	Neurons of Encoder			Hidden Representation	Neurons of Decoder			OutPut layer
Model A	Shape of FAM	64			32	64			Shape of FAM
Model B		64	64	64	32	64	64	64	
Model C		64 64	64	64 64	32	64 64	64	64 64	
Model D		128	64	32	16	32	64	128	
Model E		256	128	64	32	64	128	256	
Model F		512	256	128	64	128	256	512	
ModelG		1024	512	256	128	256	512	1024	

4.1.3 Data Spliting

As shown in Table 3, We divided the pre-processed resampling dataset into two training sets and a test set for the subsequent experiments. Neither the training nor the test sets were repeated in order to ensure the reliability of the experimental results. During training, we mixed normal samples with malicious samples, removed labels, and trained the model in an unsupervised manner. Considering that this is an unsupervised model, it cannot directly generate classification labels. Therefore, during testing, each kind of malicious data is combined with normal data to determine whether it is an attack to evaluate the performance of the model.

4.2 Contrast Experiment Setup

In order to verify our four concerns, we set up four groups of experiments.

Table 5. *Experimental hardware and software environment.*

	Category	Parameters
Hardware	**GPU**	NVIDIA RTX3070 8G
	CPU	AMD Ryzen7 5800H
	Memory	16GB
Software	**Tensorflow-gou**	2.6.0
	Keras	2.6.0
	CUDA Version	11.3.1
	CuDNN Version	8.1
	Operating System	Win 11

4.2.1 Experiment 1

We designed seven intrusion detection models based on AE, named Model A˜G their structure and number of neurons are shown in Table 4. In Model A˜Model C, only the model structure is changed in order to analyze the effect of model structure on model performance. Model D˜Model G fixes the model structure and changes only the number of neurons, aiming to analyze how the number of neurons affects the model performance. These seven models were used to conduct experiments on two training datasets1. The Adam optimizer was applied to train each model for 100 epochs. The batch size is set to 256. Table 5 provides information regarding the software and hardware environment used in the experiment.

4.2.2 Experiment 2

In order to verify the performance of AE, we implement seven unsupervised intrusion detection algorithms (MCD [54], COPOD [55], SUOD [56], LUNAR [57], VAE [58], DeepSVDD [59], ALAD [60]), including linear model, probabilistic model, outlier ensembles model, graph-based models and neural networks under the Pyod [61] framework. Table 5 shows the experimental environment. The parameters of the five models use the default values in the Pyod framework. The AE hyper-parameters are the same as those used in Experiment 1.

4.2.3 Experiment 3

Using virtualization techniques, we constructed 20 sub-nodes and one master node in order to verify the performance of federated learning. The resource situation of each node is shown in Table 5. Using the training dataset2, we divided it into 20 pieces to simulate the data collected on each sub-node as shown in Table 6. A comparison is made between the performance of the AE models trained locally and federally on each sub node separately. The AE models follow the same structure as MODEL E. Table 7 lists the hyper-parameters for the AE models.

4.2.4 Experiment 4

We perform interpretability analyses using the original SHAP and the proposed Chi-square SHAP on a trained model from Experiment 3. Model-interpretable results

Table 6. *Data volume of each sub-node.*

UNSW-NB15			
Node number	**Records**	**Node number**	**Records**
0	11871	10	9687
1	10743	11	8361
2	9926	12	2515
3	9575	13	2045
4	3224	14	2682
5	10548	15	8162
6	9671	16	7305
7	9755	17	4604
8	10968	18	7248
9	9237	19	6314
CICIDS2017			
Node number	**Records**	**Node number**	**Records**
0	36741	10	144962
1	55684	11	33195
2	118518	12	30504
3	147268	13	78260
4	115788	14	94426
5	74801	15	103380
6	50624	16	119949
7	69683	17	123227
8	67344	18	141198
9	147333	19	69260

Table 7. *Experimental hyper-parameters.*

	Federated Training	**Local Training**
optimizer	adam	adam
batch_size	256	256
local epoch	50	100
total epoch	50	/

are used to combine features and retrain the model. Evaluate the interpretable method's performance by analyzing the accuracy of the retrained model.

4.3 Evaluation Metrics

The performance evaluation indicators in this article include ROC and AUC, which are derived from four parameter indicators TP, TN, FP, and FN. The following is a detailed description of the performance evaluation indicators of this article.

1. ROC: Mainly used for two types of performance evaluation indicators. This experiment uses ROC as the most important indicator to evaluate the performance of the model. Each point of the ROC curve reflects the sensitivity to the same signal stimulus. The horizontal axis and vertical axis of the curve are respectively composed of false positive rate (FPR) and true positive rate (TPR). False positive rate: the specific ratio, that is, the proportion of actual negative cases predicted to be positive to all negative cases. The formula is:

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

TPR: represents the number of samples with predicted anomalies and actual anomalies, as a percentage of the actual total number of anomalies:

$$FPR = \frac{TP}{FN + TP}. \tag{13}$$

2. AUC: is the area under the ROC curve, which can intuitively evaluate the classifier's quality. The larger the AUC value, the better the effect of the model classifier. M is the number of positive samples, and N is the number of negative samples. Rank is the rank of the probability score from large to small. The formula is:

$$AUC = \frac{\sum_{\text{positive}} \text{rank}_i - \frac{M(1+M)}{2}}{M * N}. \tag{14}$$

3. Precision: It is mainly used to measure the accuracy of the model in predicting a category. The formula is:

$$Precision = \frac{TP}{TP + FN}. \tag{15}$$

4. Recall: describes how many positive cases in the sample were correctly predicted. The formula is:

$$Recall = \frac{TP}{TP + FN}. \tag{16}$$

V. EXPERIMENTAL RESULTS

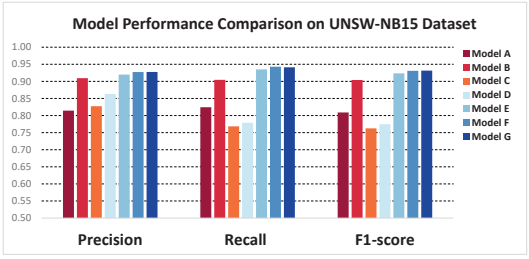
5.1 Experiment 1

Based on the two datasets, we compared the evaluation metrics and AUC of the seven models, and the results are shown in Figure 8, Table 8 and Table 9 show the performance of the model.

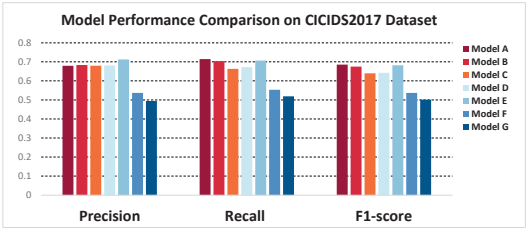
According to Table 8, on the UNSW-NB15 dataset, the seven unsupervised AE models achieve higher Precision, Recall, and F1-score, and their average values are basically higher than 0.8. While on CICIDS2017,

Table 8. *Exp1: evaluation metrics.*

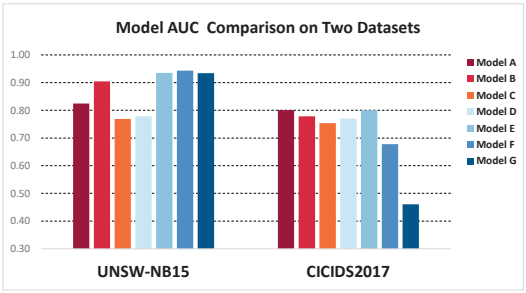
		UNSW-NB15			CICIDS2017		
		P	R	F	P	R	F
Model A	**NRM**	0.917	0.765	0.832	0.768	0.905	0.824
	ABN	0.711	0.883	0.786	0.591	0.524	0.547
	Avg	0.814	0.824	0.809	0.679	**0.715**	**0.686**
Model B	**NRM**	0.926	0.931	0.927	0.741	0.953	0.829
	ABN	0.893	0.878	0.881	0.625	0.455	0.521
	Avg	**0.909**	**0.904**	**0.904**	**0.683**	0.704	0.675
Model C	**NRM**	0.794	0.952	0.861	0.710	0.930	0.802
	ABM	0.861	0.585	0.665	0.649	0.396	0.477
	Avg	0.827	0.768	0.763	0.679	0.663	0.639
Model D	**NRM**	0.795	0.979	0.872	0.710	0.971	0.817
	ABM	0.931	0.579	0.677	0.652	0.374	0.467
	Avg	0.863	0.779	0.775	0.681	0.672	0.642
Model E	**NRM**	0.990	0.884	0.934	0.744	0.945	0.828
	ABM	0.850	0.986	0.913	0.681	0.468	0.535
	Avg	0.920	0.935	**0.923**	**0.713**	**0.707**	**0.682**
Model F	**NRM**	0.997	0.889	0.940	0.673	0.609	0.633
	ABM	0.857	0.996	0.922	0.401	0.497	0.440
	Avg	0.927	**0.943**	0.931	0.537	0.553	0.537
Model G	**NRM**	0.987	0.901	0.942	0.639	0.609	0.618
	ABM	0.868	0.982	0.921	0.349	0.428	0.384
	Avg	**0.928**	0.941	0.931	0.494	0.519	0.501



(a) *Model performance comparison on UNSW-NB15.*



(b) *Model performance comparison on CICIDS2017.*



(c) *Model AUC comparison.*

Figure 8. *Exp1: Model performance comparison.*

the lower recognition accuracy of PortScan by the seven models leads to lower average values of their metrics, but they also exceed 0.7, demonstrating the feasibility of using AE models for intrusion detection.

By comparing the evaluation metrics of models A,B,C, we can determine how model structure influences recognition accuracy. As can be seen in Figure 8, Precision, Recall, and F1-score do not increase linearly with increasing layers. According to UNSW-NB15, Model B had the highest average values for Precision, Recall, and F1-score. Model B has a higher average Precision on the CICIDS2017 dataset, while Model A has a better average Recall and F1-score metric. According to Figure 8, Model B has a higher average AUC on the UNSW-NB15 dataset than Model A, and Model A has a higher average AUC on the CICIDS2017 dataset. Compared to Model B, Model C has 1.5 times more neurons and takes 1.2 times longer to train. As a result, we use the structure of Model B as the model structure based on the evaluation metrics and the resource consumption of the model.

By comparing the evaluation metrics of models D, E, F, G, the influence of the number of neurons in each layer of the model on recognition accuracy can be determined. In Figure 8, it can be seen that the average values of Recall, F1-score, and AUC of models E, F, and G are better than those of model B on the UNSW-NB15 dataset. On the CICIDS2017 dataset, model E has higher Precision, Recall, F1-score, and AUC values than model B. This indicates that the average values of Precision, Recall, and F1-score improve as the number of neurons in each layer increases. However, this improvement did not increase with the number of neurons. On the UNSW-NB15 dataset, model G has lower Recall, F1-score and AUC values than model F. On the CICIDS2017 dataset, models F, G have lower Precision, Recall, F1-score and AUC values than model E. Despite performing slightly better than model E on the UNSW-NB15 dataset, model F has three times more neurons and takes a longer time to train. In the proposed method, we use model E to detect intrusions.

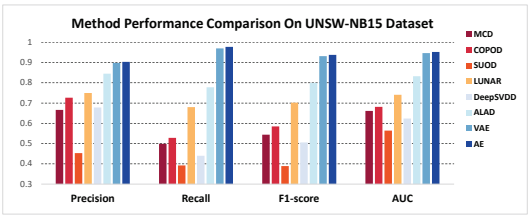
5.2 Experiment 2

Table 11 and Table 12 provide a detailed list of the evaluation metrics for the eight unsupervised intrusion detection models. Similarly to Experiment 1,

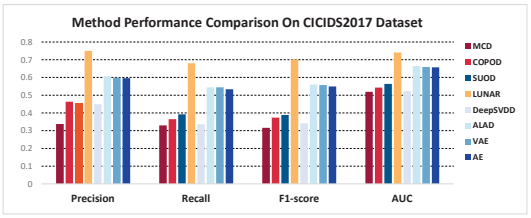
Table 9. *Exp1: Computing resource consumption comparison.*

Model Name		Model A	Model B	Model C	Model D	Model E	Model F	Model G
Parameter quantity		14,318	30,958	47,598	42,014	126,958	425,870	1,539,790
Model Analysis Report(FLOPs)	Mul	14.08k	30.46k	46.85k	41.47k	125.95k	423.94k	1.54m
	Sub	4	8	12	8	8	8	8
	Total	14084	30472	46860	41480	125960	423944	1536008
UNSW-NB15	used_time(s)	450.60	828.27	951.02	651.57	779.41	878.81	978.11
CICIDS2017	used_time(s)	803.30	1156.77	1445.18	1164.08	1236.24	1271.13	1310.30

all eight methods performed better on the UNSW-NB15 dataset than on the CICIDS2017 dataset. On the UNSW-NB15 dataset, Precision, Recall, and F1-score are at least 0.2 higher than on the CICIDS2017 dataset. Across all 8 classes of methods, PortScan is poorly recognized on the CICIDS2017 dataset. Precision, Recall, and F1-score are close to zero for all 8 methods.



(a) *Model performance comparison on UNSW-NB15.*



(b) *Model performance comparison on CICIDS2017.*

Figure 9. *Exp2: Model performance comparison.*

According to Figure 9, the mean values of the evaluation metrics of the Neural Networks methods are significantly better than those of Linear, Outlier Ensembles and Probabilistic Models, except for DeepSVDD. The Precision, Recall, and F1-score of the three methods are higher by at least 0.1 on the UNSW-NB15 dataset, and they are higher by nearly 0.2 on the CICIDS2017 dataset. Although LUNAR method is better on CICIDS2017 dataset, the accuracy of neural network is more balanced according to the two datasets. Thus, neural networks are more effective at detecting malicious traffic intrusions than other methods.

Figure 10 illustrates the confusion matrices of the four Neural Networks on the two datasets. It can be seen that VAE and SAE have excellent recognition performance on the UNSW-NB15 dataset. They can detect basically all malicious samples correctly, while the false recognition rate of normal samples is also low. The recognition performance of ALAD, VAE, and SAE is similar on the CICIDS2017 dataset, and all can detect DOS and DDOS with a low number of false positives.

As shown in Table 10, Linear Models and Probabilistic Models have significantly better training times than Neural Networks except for ALAD. The performance of DeepSVDD is not satisfactory. Although LUNAR method has good accuracy in CICIDS2017 data set, its training time is too long, 100 times that of AE method. Despite the similarity between VAE and SAE in terms of recognition performance and time consumption, the VAE model has a more complex structure. Overall, SAE appears to be a more appropriate model for FL-TIDS.

Table 10. *Exp2: Computing resource consumption comparison.*

Method	UNSW-NB15		CICIDS2017	
	Training(s)	Testing(s)	Training(s)	Testing(s)
MCD	482.84	0.01	1369.16	0.01
COPOD	53.16	54.03	75.64	77.01
SUOD	22814.53	10.15	38066.95	16.96
LUNAR	54389.88	22.71	55050.09	23.18
ALAD	220.09	0.02	243.38	0.02
DeepSVDD	7579.20	0.24	8432.09	0.16
VAE	5854.54	0.23	7513.51	0.18
AE	5325.34	0.24	7354.83	0.19

5.3 Experiment 3

We used model E in Experiment 1 as the intrusion detection model and conducted experiments on two datasets with federal and local training respectively. The evaluation metrics of the experiments are shown in Table 13. From Table 13, it can be seen that on the UNSW-NB15 dataset, the average values of Precision, Recall, F1-score of FL-TIDS are higher than

Table 11. *Exp2: Evaluation metrics (UNSW-NB15).*

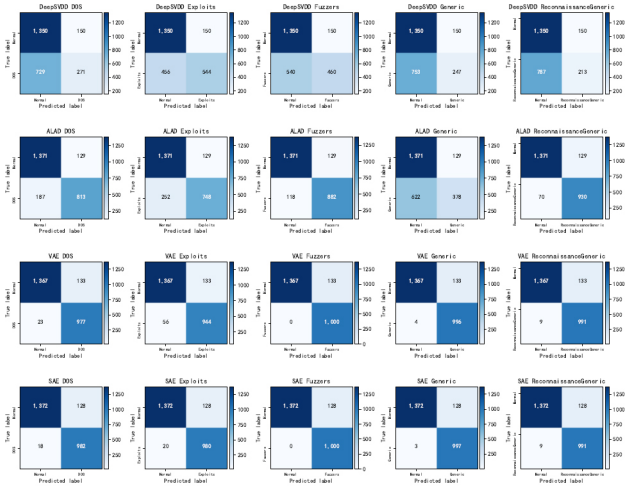
Method	Detection	P	R	F	AUC	Method	Detection	P	R	F	AUC
MCD	Normal	0.71	0.90	0.79	/	**ALAD**	Normal	0.86	0.91	0.88	/
	DOS	0.59	0.21	0.31	0.56		DOS	0.86	0.81	0.84	0.86
	Exploits	0.82	0.68	0.74	0.79		Exploits	0.85	0.75	0.80	0.83
	Fuzzers	0.83	0.69	0.75	0.80		Fuzzers	0.87	0.88	0.88	0.90
	Generic	0.28	0.06	0.09	0.48		Generic	0.75	0.38	0.50	0.65
	Reconnaissance	0.76	0.46	0.57	0.68		Reconnaissance	0.88	0.93	0.90	0.92
	avg	0.67	0.50	0.54	0.66		**avg**	0.84	0.78	0.80	0.83
COPOD	Normal	0.72	0.91	0.80	/	**DeepSVDD**	Normal	0.68	0.90	0.77	/
	DOS	0.67	0.27	0.39	0.59		DOS	0.64	0.27	0.38	0.59
	Exploits	0.83	0.66	0.74	0.79		Exploits	0.78	0.54	0.64	0.72
	Fuzzers	0.84	0.70	0.77	0.81		Fuzzers	0.75	0.46	0.57	0.68
	Generic	0.50	0.14	0.21	0.52		Generic	0.62	0.25	0.35	0.57
	Reconnaissance	0.78	0.48	0.60	0.70		Reconnaissance	0.59	0.21	0.31	0.56
	avg	0.73	0.53	0.58	0.68		**avg**	0.68	0.44	0.51	0.62
SUOD	Normal	0.71	0.91	0.80	/	**VAE**	Normal	0.99	0.91	0.95	/
	DOS	0.70	0.33	0.45	0.62		DOS	0.88	0.98	0.93	0.94
	Exploits	0.80	0.57	0.67	0.74		Exploits	0.88	0.94	0.91	0.93
	Fuzzers	0.82	0.65	0.72	0.78		Fuzzers	0.88	1.00	0.94	0.96
	Generic	0.68	0.30	0.42	0.61		Generic	0.88	1.00	0.94	0.95
	Reconnaissance	0.72	0.36	0.48	0.63		Reconnaissance	0.88	0.99	0.93	0.95
	avg	0.74	0.52	0.59	0.67		**avg**	**0.90**	0.97	0.93	**0.95**
LUNAR	Normal	0.69	0.73	0.70	/	**AE**	Normal	0.99	0.91	0.95	/
	DOS	0.51	0.42	0.46	0.57		DOS	0.88	0.98	0.93	0.95
	Exploits	0.65	0.77	0.71	0.75		Exploits	0.88	0.98	0.93	0.95
	Fuzzers	0.63	0.69	0.65	0.71		Fuzzers	0.89	1.00	0.94	0.96
	Generic	0.27	0.15	0.20	0.44		Generic	0.89	1.00	0.94	0.96
	Reconnaissance	0.47	0.37	0.41	0.55		Reconnaissance	0.89	0.99	0.94	0.95
	avg	0.54	0.52	0.52	0.60		**avg**	**0.90**	**0.98**	**0.94**	**0.95**

Table 12. *Exp2: Evaluation metrics (CICIDS2017).*

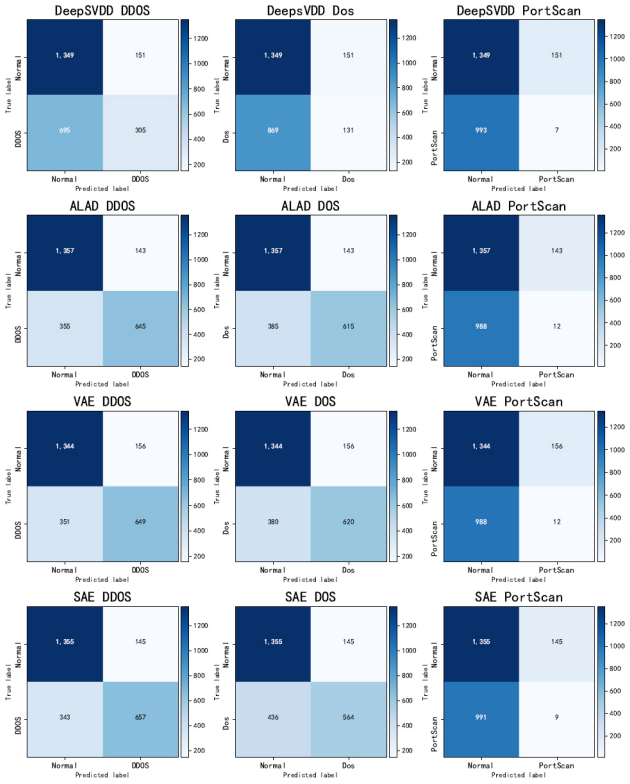
Method	Detection	P	R	F	AUC	Method	Detection	P	R	F	AUC
MCD	Normal	0.62	0.90	0.73	/	**ALAD**	Normal	0.72	0.90	0.80	/
	DDOS	0.00	0.00	0.00	0.45		DDOS	0.82	0.65	0.72	0.77
	DOS	0.73	0.42	0.53	0.66		DOS	0.81	0.62	0.70	0.76
	PortScan	0.00	0.00	0.00	0.45		PortScan	0.08	0.01	0.02	0.46
	avg	0.34	0.33	0.32	0.52		**avg**	0.61	0.54	0.56	0.66
COPOD	Normal	0.63	0.90	0.74	/	**DeepSVDD**	Normal	0.61	0.90	0.73	/
	DDOS	0.49	0.14	0.22	0.52		DDOS	0.67	0.31	0.42	0.60
	DOS	0.73	0.41	0.53	0.66		DOS	0.46	0.13	0.20	0.51
	PortScan	0.01	0.00	0.00	0.45		PortScan	0.04	0.01	0.01	0.45
	avg	0.46	0.36	0.37	0.54		**avg**	0.45	0.34	0.34	0.52
SUOD	Normal	0.65	0.91	0.75	/	**VAE**	Normal	0.72	0.90	0.79	/
	DDOS	0.36	0.08	0.13	0.49		DDOS	0.81	0.65	0.72	0.77
	DOS	0.81	0.58	0.68	0.74		DOS	0.80	0.62	0.70	0.75
	PortScan	0.00	0.00	0.00	0.45		PortScan	0.07	0.01	0.02	0.45
	avg	0.45	0.39	0.39	0.56		**avg**	0.60	0.54	0.56	0.66
LUNAR	Normal	0.78	0.87	0.82	/	**AE**	Normal	0.71	0.90	0.79	/
	DDOS	0.78	0.70	0.74	0.78		DDOS	0.82	0.66	0.73	0.78
	DOS	0.81	0.82	0.81	0.84		DOS	0.80	0.56	0.66	0.73
	PortScan	0.63	0.33	0.43	0.60		PortScan	0.06	0.01	0.02	0.46
	avg	**0.75**	**0.68**	**0.70**	**0.74**		**avg**	0.60	0.53	0.55	0.66

Local-TIDS except Generic. On CICIDS2017, the average values of Precision, Recall, F1-score of FL-TIDS are also higher than those of Local-TIDS. It can be seen that the accuracy of the intrusion detection model is significantly improved after the addition of federal learning.

Figure 11 and Figure 12 illustrate a comparison of the AUC of FL-TIDS and Local-TIDS on 20 nodes. Clearly, the AUC of Local-TIDS changes more dramatically and is greatly influenced by the amount of node data. Nevertheless, the AUC of FL-TIDS is less affected by the amount of node data and changes



(a) *Confusion matrix comparison on UNSW-NB15 dataset.*



(b) *Confusion matrix comparison on CICIDS2017 dataset.*

Figure 10. *Exp2: Method performance confusion matrix comparison.*

more gently. FL-TIDS achieved an average AUC of 0.92 on UNSW-NB15, which was 0.06 higher than that of Local-TIDS. According to CICIDS2017, FL-TIDS achieved an average AUC of 0.91, 0.12 higher than Local-TIDS. Each node uses a small amount of UNSW-NB15 data, and only about 7% of training

Table 13. *Exp3: Evaluation metrics.*

UNSW-NB15						
	Local-TIDS			FL-TIDS		
	P	R	F	P	R	F
Normal	0.74	0.88	0.78	**0.83**	**0.91**	**0.87**
DOS	0.84	0.56	0.63	**0.86**	**0.81**	**0.83**
Exploits	0.84	0.57	0.64	**0.87**	**0.90**	**0.88**
Fuzzers	0.78	0.37	0.45	**0.85**	**0.75**	**0.80**
Generic	**0.84**	**0.70**	**0.72**	0.81	0.58	0.68
Reconna-issance	0.74	0.30	0.38	**0.79**	**0.49**	**0.61**
ALL	0.80	0.56	0.60	**0.83**	**0.74**	**0.78**
CICIDS2017						
	Local-TIDS			FL-TIDS		
	P	R	F	P	R	F
Normal	0.69	0.84	0.75	**0.87**	**0.86**	**0.86**
DDOS	0.67	0.45	0.50	**0.80**	**0.84**	**0.82**
DOS	0.75	0.61	0.65	**0.77**	**0.71**	**0.74**
PortScan	0.77	0.71	0.74	**0.80**	**0.83**	**0.81**
ALL	0.72	0.65	0.66	**0.81**	**0.81**	**0.81**

dataset2 is used. About 91% of the training dataset2 is used by each node using CICIDS2017. The difference in AUC on the two datasets indicates that FL-TIDS improves more in AUC with more data.

The differences between the two methods cannot be determined based solely on AUC. For the analysis, we selected the confusion matrix of nodes 4 and 11. According to Figure 13 and Figure 14, both Local-TIDS and FL-TIDS have high recognition rates for normal traffic on the UNSW-NB15 dataset.

Local-TIDS, however, incorrectly identifies 299, 604, and 752 flow records for Exploits, Fuzzers, and Reconnaissance. In contrast, FL-TIDS had only 97, 255, and 522 flow records that were misidentified, significantly improving recognition accuracy. On the CICIDS2017 dataset, Local-TIDS was only able to identify a small number of flow records for PortSCAN, whereas FL-TIDS accurately identified 752 records. Furthermore, DDOS and DOS were also identified more accurately. The introduction of federated learning has greatly improved the recognition accuracy of the model.

Table 14 illustrates the resource consumption of FL-TIDS and Local-TIDS. The avg_used_time(s) in 'localtrain' refers to the time consumed to train 50 epochs per node, while the avg_used_time(s) in 'globaltrain' refers to the time consumed to complete training 50 epochs for 20 nodes. In terms of GPU, memory, and CPU usage, FL-TIDS and Local-TIDS have little difference. The main difference is in training time, FL-TIDS is half that of Local-TIDS since each node

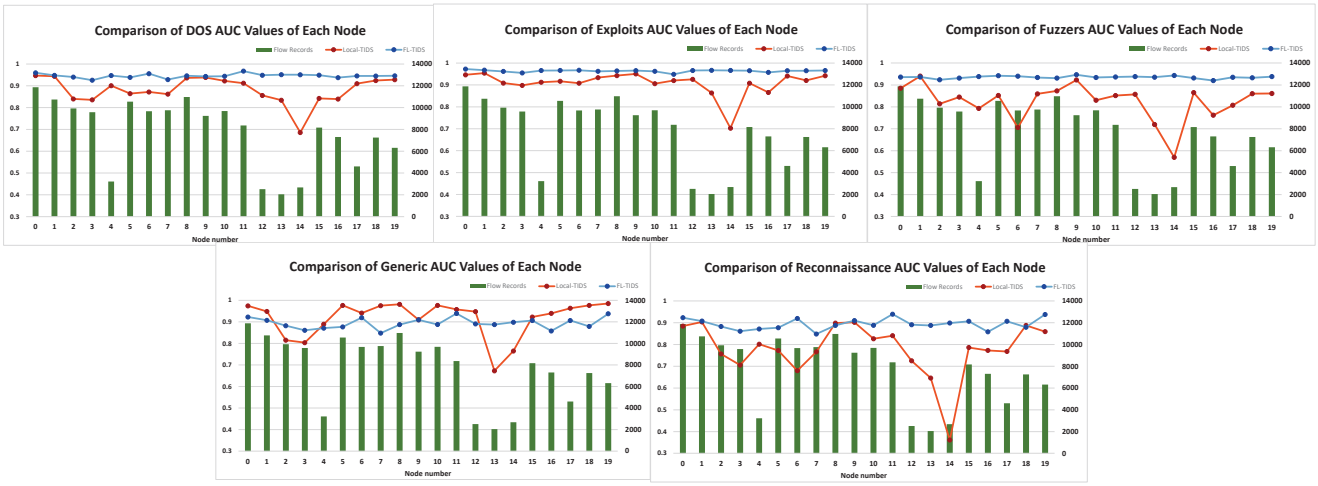


Figure 11. *Exp3: Model performance comparison on UNSW-NB15.*

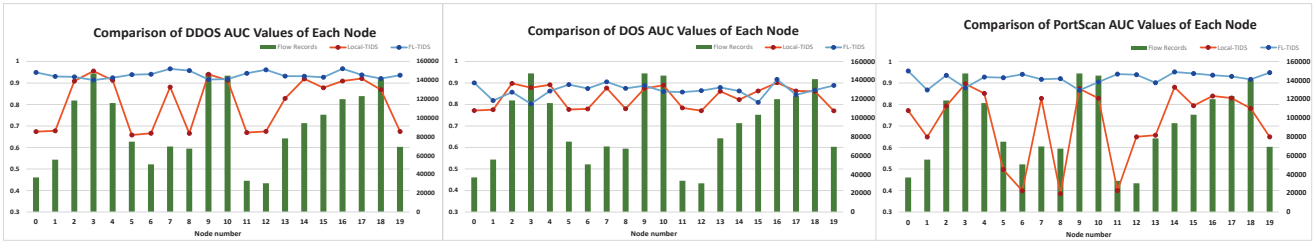


Figure 12. *Exp3: Model performance comparison on CICIDS2017.*

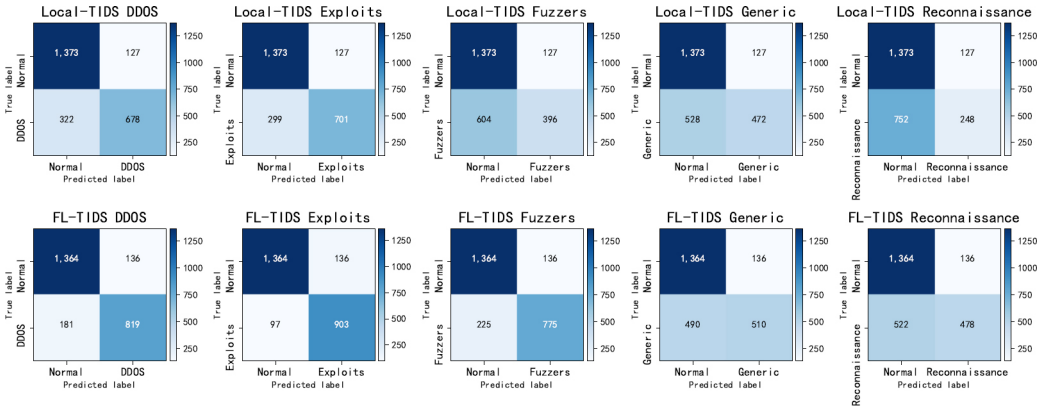


Figure 13. *Exp3: Method performance confusion matrix comparison on UNSW-NB15.*

only trains 50 epochs. FL-TIDS, however, requires 50 epochs for global training, which results in an 18-fold time consumption over Local-TIDS. It is important to note, that since FL-TIDS is currently trained in sequence rather than asynchronously, there is still room for improvement. When the model is used for prediction, FL-TIDS and Local-TIDS consume similar amounts of time due to the similar structure of the models. It is estimated that 11,000 flow records can be processed in one second, which meets the actual requirements for deployment.

5.4 Experiment 4

Using the intrusion detection model of FL-TIDS of node 7 in Experiment 3 as a benchmark, we conducted interpretable analysis experiments based on SHAP and the proposed Chi-square SHAP method. Based on the CICIDS2017 dataset, we conducted in-

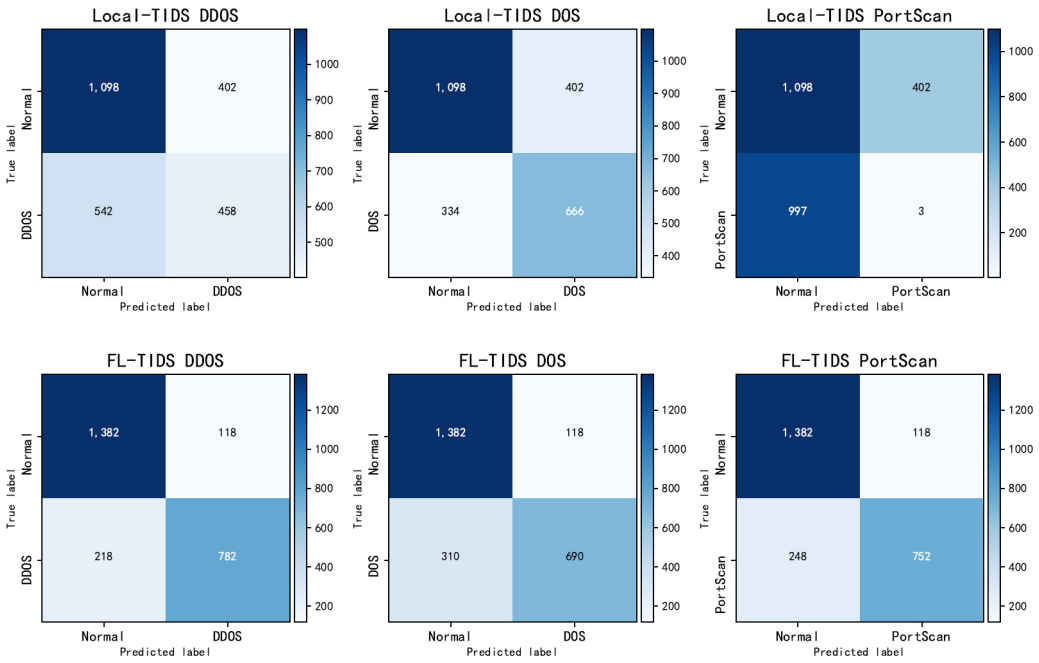


Figure 14. *Exp3: Method performance confusion matrix comparison on CICIDS2017.*

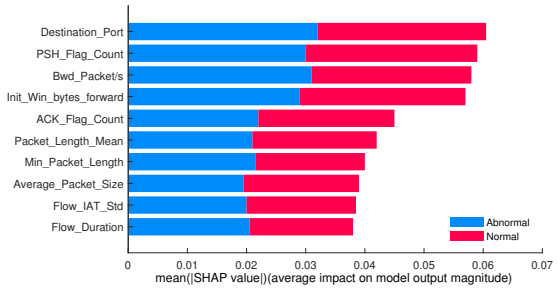
Table 14. *Exp3: Computing resource consumption comparison.*

	UNSW-NB15				
	Local-TIDS		FL-TIDS		
	Local Training	Testing	Local Training	Global Training	Testing
GPU MEM(%)	85.21	85.55	83.58	/	87.78
GPU Load(%)	36.81	19.89	34.4	/	17.04
MEM(%)	11.2	10.51	12.18	2.87	10.91
CPU Load(%)	136.19	105.65	130.52	136.78	102.73
Time(s)	29	0.22	10.63	222.78	0.29
Total Time(s)	580.97	/	11139.23		/
	CICIDS2017				
	Local-TIDS		FL-TIDS		
	Local Training	Testing	Local Training	Global Training	Testing
GPU MEM(%)	87.96	87.46	86.22	/	86.79
GPU Load(%)	36.55	20.34	36.2	/	18.26
MEM(%)	13.81	10.45	14.18	4.37	10.46
CPU Load(%)	134.25	100.35	140.24	138.92	105.59
Time(s)	116.17	0.24	40.57	837.33	0.22
Total Time(s)	2303.18	/	41866.9		/

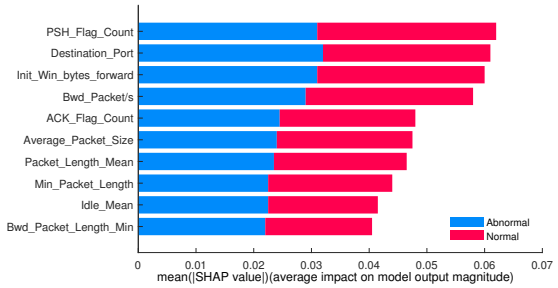
terpretable analysis for PortScan in an attempt to solve the problem of low recognition rates by the AE model. The results of the interpretable analysis are shown in Figure 15. In both interpretable methods, the top ten features analyzed are: "Init_Win_Bytes_Forward", "Bwd_Packet/s" and "PSH_Flag_Count".

In accordance with the suggestions provided by the Canadian Institute for Network Security [52], the analysis results obtained by both methods are highly reliable. However, there are some differences between the two methods. According to the SHAP method, "Flow_IAT_Std" and "Flow_Duration" are key features, but these features are not included in the Chi-square SHAP method, instead they are replaced by "Idle_Mean" and "Bwd_Packet_Length_Min".

To compare the reliability of the two interpretable methods more intuitively, we filtered the top 10 SHAP-based and Chi-square SHAP based features from the CICIDS2017 dataset in the training dataset2 and retrained the model locally. Two retrained models have the same structure as model E. Both models are trained with the same hyper-parameters. The Adam optimizer was applied to train each model for 100 epochs. The batch size is set to 256. The experimental results are shown in Table 15. It shows that Precision, Recall, and F1-score have been greatly improved after interpretable analysis and retraining. The results



(a) *PortScan key features from SHAP.*



(b) *PortScan key features from Chi-square SHAP.*

Figure 15. *Exp4: Model interpretable analysis.*

indicate that the features derived from the two interpretable methods have a significant impact on the accuracy of the model. Among them, the Precision, Recall, F1-score, and AUC of the model retrained based on the Chi-square SHAP analysis results are higher than those of the model retrained based on the SHAP analysis results.

Table 15. *Exp4: Evaluation metrics.*

	SHAP Based Re-trained Model				SHAP Based Re-trained Model			
	P	**R**	**F**	**AUC**	**P**	**R**	**F**	**AUC**
NBM	0.93	0.82	0.87	/	1	0.86	0.92	/
ABM	0.77	0.91	0.84	0.94	0.83	1	0.9	0.99
AVG	0.85	0.87	0.86		**0.91**	**0.93**	**0.91**	

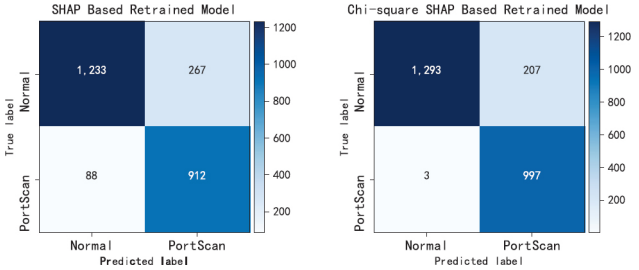


Figure 16. *Exp4: Method performance confusion matrix comparison.*

The confusion matrix for both models is shown in Figure 16, where it can be seen that the number of flow records correctly identified by the model retrained based on the Chi-square SHAP analysis results is higher for both normal and malicious traffic. Results indicate that the features derived from the interpretable Chi-square SHAP analysis have a greater influence on model accuracy. The time consumption of the two methods is shown in Figure 17. As a result of the addition of chi-square calculations, Chi-square SHAP takes about 10% longer than SHAP. Nevertheless, given that the proposed method improves model accuracy, we believe the additional time consumption is justified.

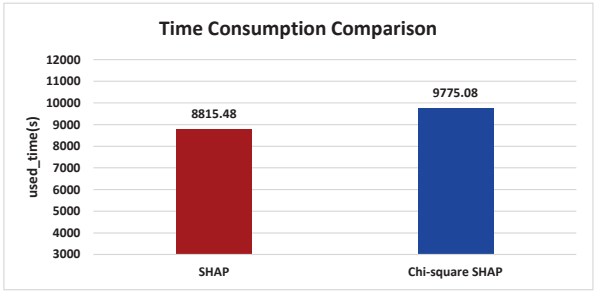


Figure 17. *Exp4: Time consumption comparison.*

VI. CONCLUSION

In order to solve the problem that it is difficult to obtain labelled data and difficult to collect network traffic involving privacy, we innovatively combine federal learning and intrusion detection and propose the FL-TIDS architecture. The proposed FL-TIDS architecture integrates traffic collection, data processing, model aggregation and interpretable analysis functions and utilizes an unsupervised intrusion detection method based on anomaly discrimination, using only normal traffic for model training. With effective protection of data privacy, the workload of data annotation is reduced, while the recognition performance of each node is improved.

In this paper, we first determine experimentally the structure and the number of neurons of the unsupervised AE model. Secondly, we evaluated the proposed method using the UNSW-NB15 and CICIDS20117 datasets. The experimental results show that the unsupervised AE model has higher accuracy than the other intrusion detection models. Then, federated learning

is used to train the intrusion detection model. The experimental results indicate that the model is more accurate than the local learning model. As a final step, we conducted an interpretability analysis utilizing the improved SHAP interpretability method based on the chi-square test. The analysis results show that the features identified by the model are consistent with those of the attack, and the proposed model is credible and reliable.

This paper is an attempt to combine intrusion detection with federated learning. The experimental results show that there are still many shortcomings. For example, there is still room for improvement in the recognition accuracy of malicious traffic in both global and local models. Moreover, the global training time is too long and much longer than the local training, which is not conducive to the application. In the future, we will focus on optimizing the model size and convergence speed and strive to train a better-performing intrusion detection model in less time and with fewer resources to enhance the practical application value.

VII. ACKNOWLEDGMENT

The paper is supported by National Natural Science Fundation of China under Grant 61972208; National Natural Science Fundation (General Program) of China under Grant 61972211; National Key Research and Development Project of China under Grant 2020YFB1804700; Future Network Innovation Research and Application Projects under Grant No.2021FNA02006 and 2021 Jiangsu Postgraduate Research Innovation Plan under Grant No.KYCX210794.

References

[1] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.

[2] S. Agrawal, S. Sarkar, *et al.*, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *arXiv preprint arXiv:2106.09527*, 2021.

[3] W. Zhou, S. Lei, *et al.*, "Lda-id: An lda-based framework for real-time network intrusion detection," *China Communications*, 2023, vol. 20, no. 12, pp. 166–181.

[4] R. Arthi, S. Krishnaveni, *et al.*, "An intelligent sdn-iot enabled intrusion detection system for healthcare systems using a hybrid deep learning and machine learning approach," *China Communications*, 2024.

[5] Y. Zong, Y. Luo, *et al.*, "Online intrusion detection mechanism based on model migration in intelligent pumped storage power stations," *China Communications*, 2023, vol. 20, no. 4, pp. 368–381.

[6] J. Wu, H. Zheng, *et al.*, "Unknown application layer protocol recognition method based on deep clustering," *China Communications*, 2024.

[7] Z. Wu, Y. Li, *et al.*, "A hybrid intrusion detection method based on convolutional neural network and adaboost," *China Communications*, 2024.

[8] T. Li, A. K. Sahu, *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020, vol. 37, no. 3, pp. 50–60.

[9] R. Lohiya and A. Thakkar, "Application domains, evaluation data sets, and research challenges of iot: A systematic review," *IEEE Internet of Things Journal*, 2020, vol. 8, no. 11, pp. 8774–8798.

[10] E. Hesamifard, H. Takabi, *et al.*, "Privacy-preserving machine learning as a service." *Proc. Priv. Enhancing Technol.*, 2018, vol. 2018, no. 3, pp. 123–142.

[11] Q. Yang, Y. Liu, *et al.*, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019, vol. 10, no. 2, pp. 1–19.

[12] B. McMahan, E. Moore, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[13] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, 2020, vol. 167, pp. 636–645.

[14] R. Agarwal and M. Joshi, "A new framework for learning classifier models in data mining," *ReportNo. RC-21719*, 2000.

[15] P. A. Porras and R. A. Kemmerer, "Penetration state transition analysis: A rule-based intrusion detection approach," in *Proceedings Eighth Annual Computer Security Application Conference*. IEEE Computer Society, 1992, pp. 220–221.

[16] T. Sheu, N. Huang, *et al.*, "Nis04-6: A time-and memory-efficient string matching algorithm for intrusion detection systems," in *IEEE Globecom 2006*, 2006, pp. 1–5.

[17] Z. Pan, H. Lian, *et al.*, "An integrated model of intrusion detection based on neural network and expert system," in *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, 2005.

[18] R. Bronte, H. Shahriar, *et al.*, "A signature-based intrusion detection system for web applications based on genetic algorithm," in *Proceedings of the 9th International Conference on Security of Information and Networks*, 2016, pp. 32–39.

[19] E. Nikolova and V. Jecheva, "Some similarity coefficients and application of data mining techniques to the anomaly-based ids," *Telecommunication Systems*, 2012, vol. 50, no. 2, pp. 127–135.

[20] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of ids for iot: Recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, 2021, vol. 28, no. 4, pp.

3211–3243.
[21] A. Dainotti, A. Pescape, *et al.*, "Issues and future directions in traffic classification," *IEEE Network*, 2012, vol. 26, no. 1, pp. 35–40.
[22] G. Sun, Y. Xue, *et al.*, "An novel hybrid method for effectively classifying encrypted traffic," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010, pp. 1–5.
[23] A. Thakkar and R. Lohiya, "Analyzing fusion of regularization techniques in the deep learning-based intrusion detection system," *International Journal of Intelligent Systems*, 2021, vol. 36, no. 12, pp. 7340–7388.
[24] Y. Nuo, "A novel selection method of network intrusion optimal route detection based on naive bayesian," *International Journal of Applied Decision Sciences*, 2018, vol. 11, no. 1, pp. 1–17.
[25] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "An anomaly-based intrusion detection system for the smart grid based on cart decision tree," in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–5.
[26] F. Jiang, C. Wang, *et al.*, "Relative decision entropy based decision tree algorithm and its application in intrusion detection," *Computer Science*, 2012, vol. 39, no. 4, pp. 223–226.
[27] L. Teng, S. Teng, *et al.*, "A collaborative and adaptive intrusion detection based on svms and decision trees," in *2014 IEEE International Conference on Data Mining Workshop*, 2014, pp. 898–905.
[28] T. Hurley, J. E. Perdomo, *et al.*, "Hmm-based intrusion detection system for software defined networking," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 617–621.
[29] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, 2021, pp. 1–111.
[30] R. Lohiya and A. Thakkar, "Intrusion detection using deep neural network with antirectifier layer," in *Applied Soft Computing and Communication Networks*. Springer, 2021, pp. 89–105.
[31] P. Wang, F. Ye, *et al.*, "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway," *IEEE Access*, 2018, vol. 6, pp. 55 380–55 391.
[32] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015, vol. 24, no. 11, pp. 1–10.
[33] W. Wang, M. Zhu, *et al.*, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*. IEEE, 2017, pp. 712–717.
[34] W. Wang, M. Zhu, *et al.*, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 43–48.
[35] H. Zhou, Y. Wang, *et al.*, "A method of improved cnn traffic classification," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, 2017, pp. 177–181.
[36] W. Wang, Y. Sheng, *et al.*, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, 2017, vol. 6, pp. 1792–1806.
[37] J. Kim, S. Bu, *et al.*, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, 2018, vol. 460, pp. 83–102.
[38] Z. Li, P. Wang, *et al.*, "Flowgananomaly: Flow-based anomaly network intrusion detection with adversarial learning," 2022, p. 1. https://cje.ejournal.org.cn/en/article/doi/10.23919/cje.2022.00.173.
[39] H. Chen and L. Jiang, "Efficient gan-based method for cyber-intrusion detection," *arXiv preprint arXiv:1904.02426*, 2019.
[40] S. Lin, Y. Shi, *et al.*, "Character-level intrusion detection based on convolutional neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.
[41] M. Qi, M. Liu, *et al.*, "Research on pca-based svm network intrusion detection," *Information Network Security*, 2015, vol. 15, no. 2, pp. 15–18.
[42] A. Krizhevsky, I. Sutskever, *et al.*, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012, vol. 25.
[43] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, pp. 178–183.
[44] H. Yang, Z. Zhao, *et al.*, "Enabling intelligence at network edge: An overview of federated learning," *ZTE Communications*, 2020, vol. 18, no. 2, p. 2.
[45] W. Lim, N. Luong, *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020, vol. 22, no. 3, pp. 2031–2063.
[46] B. Tang, C. Zhang, *et al.*, "Neursafe-fl: A reliable, efficient, easy-to-use federated learning framework," *ZTE Communications*, 2022, vol. 20, no. 3, pp. 43–53.
[47] Z. Tang, H. Hu, *et al.*, "A federated learning method for network intrusion detection," *Concurrency and Computation: Practice and Experience*, 2022, p. e6812.
[48] D. C. Attota, V. Mothukuri, *et al.*, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, 2021, vol. 9, pp. 117 734–117 745.
[49] Z. Chen, N. Lyu, *et al.*, "Intrusion detection for wireless edge networks based on federated learning," *IEEE Access*, 2020, vol. 8, pp. 217 463–217 472.
[50] D. Man, F. Zeng, *et al.*, "Intelligent intrusion detection based on federated learning for edge-assisted internet of things," *Security and Communication Networks*, 2021, vol. 2021.
[51] V. Rey, P. Sánchez, *et al.*, "Federated learning for malware detection in iot devices," *Computer Networks*, 2022, p. 108693.
[52] I. Sharafaldin, A. H. Lashkari, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, 2018, vol. 1, pp. 108–116.
[53] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, 2017, vol. 30.
[54] M. Hubert, M. Debruyne, *et al.*, "Minimum covariance de-

terminant and extensions," *Wiley Interdisciplinary Reviews: Computational Statistics*, 2018, vol. 10, no. 3, p. e1421.
[55] Z. Li, Y. Zhao, *et al.*, "Copod: Copula-based outlier detection," in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 1118–1123.
[56] Y. Zhao, X. Hu, *et al.*, "Suod: Accelerating large-scale unsupervised heterogeneous outlier detection," *Proceedings of Machine Learning and Systems*, 2021, vol. 3, pp. 463–478.
[57] A. Goodge, B. Hooi, *et al.*, "Lunar: Unifying local outlier detection methods via graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, no. 6, pp. 6737–6745.
[58] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[59] L. Ruff, R. Vandermeulen, *et al.*, "Deep one-class classification," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4393–4402.
[60] Z. Houssam, R. Manon, *et al.*, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 727–736.
[61] Y. Zhao, Z. Nasrullah, *et al.*, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, 2019, vol. 20, no. 96, pp. 1–7. http://jmlr.org/papers/v20/19-011.html.

Biographies

Wang Zixuan was born in Nanjing, Jiangsu, China, in 1994. He obtained a master's degree in logistics engineering at Nanjing University of Posts and Telecommunications in 2020. He is currently pursuing a Ph.D degree at Nanjing University of Posts and Telecommunications. His research interests include encrypted traffic dentification and data balancing.

Miao Cheng was born in Taixing, Jiangsu, China, in 1998. He is currently pursuing a Master degree at Nanjing University of Posts and Telecommunications. His research interests include encrypted traffic identification and model interpretability.

Xu Yuhua is a Ph.D candidate in Nanjing University of Posts and Telecommunications. She received her B.Sc. and M.S. from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2011 and 2016. Her research area mainly includes network security, machine learning and cryptology.

Li Zeyi is currently pursuing the Ph.D degree in Cyberspace Security at Nanjing University of Posts and Telecommunications. He was born in Soochow, Jiangsu, China, in 1997. He received his bachelor's degree in mathematics in 2019 and received M.S. degree in computer science in 2022. His research interests include network security, communication network security, anomaly detection and analysis, deep packet inspection, and graph neural networks.

Sun Zhixin is the dean of School of Modern Posts & Institute of Modern Posts, Nanjing University of Posts and Telecommunications. He received his Ph.D degree in Nanjing University of Aeronautics and Astronautics, China in 1998 and worked as a post doctor in Seoul National University, South Korea between 2001 and 2002. He has published more than 50 literatures on journals worldwide. His research area includes information security, computer networks, computer science, etc.

Wang Pan (M'18) received the B.S. degree from the Department of Communication Engineering, Nanjing University of Posts & Telecommunications, Nanjing, China, in 2001, and the Ph.D degree in Electrical & Computer Engineering from Nanjing University of Posts&Telecommunications, Nanjing, China, in 2013. He is currently an Associate Professor in the School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing, China. His research interests include cyber security and communication network security, network measurements, Quality of Service, Deep Packet Inspection, SDN, big data analytics and applications. From 2017 to 2018, he was a visiting scholar of University of Dayton (UD) in the Department of Electrical and Computer Engineering.