Improved Transformer-Based Privacy-Preserving Architecture for Intrusion Detection in Secure V2X Communications

Qifeng Lai, Chen Xiong, Jian Chen, *Graduate Student Member, IEEE*, Wei Wang, Junxin Chen, *Senior Member, IEEE*, Thippa Reddy Gadekallu, *Senior Member, IEEE*, Ming Cai, and Xiping Hu

***Abstract*—The Internet of Vehicles (IoVs) makes communications between numerous devices that use various protocols susceptible to hacker incursions and attacks, which can compromise privacy and seriously jeopardize driving safety. Many studies have been proposed to detect intrusions hitherto, but two major limitations remain. First, traditional Vehicles-to-Cloud (V2C) have difficulty in figuring out the decentralized distribution of data and computational power in IoVs. Second, the majority of studies suffer from unbalanced data in which the attacks only make up a small part and fail to detect low-probability attacks. To address these limitations, we design a Federated Learning-Edge Cloud (FL-EC) communication architecture for IoVs with a Feature Select Transformer (FSFormer) for effective intrusion detection: In FL-EC, mobile users collect and encrypt data before uploading it to edges for training, with edges and cloud functioning as clients and servers in FL, ensuring privacy and efficient data transmission. In FSFormer, we propose a Feature Attention mechanism to search and promote significant features. Furthermore, the Feed-Forward Network is replaced with a Routing module for a deeper but less-parameter network. Extensive experiments show that our model effectively boosts the detection rate of low-probability attacks and outperforms five baseline models in almost all scenarios.**

***Index Terms*—Intrusion detection, transformer, deep learning, data security, edge cloud.**

Manuscript received 12 June 2023; revised 21 August 2023; accepted 21 September 2023. Date of publication 12 October 2023; date of current version 26 April 2024. This work was supported in part by the Natural Science Foundation of China under Grant 52102400 and Grant 52222810, and in part by the Sustainable Development Science and Technology Project of Shenzhen Science and Technology Innovation Commission under Grant KCXFZ20201221173411032. *(Corresponding authors: Wei Wang; Xiping Hu.)*

Qifeng Lai and Jian Chen are with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518000, China, and also with the Guangdong–Hong Kong–Macao Joint Laboratory for Emotional Intelligence and Pervasive Computing, Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, Guangdong, China (e-mail: laiqf@mail2.sysu.edu.cn; chenj589@mail2.sysu.edu.cn).

Chen Xiong and Ming Cai are with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518000, China (e-mail: xiongch8@mail.sysu.edu.cn; caiming@mail.sysu.edu.cn).

Wei Wang and Xiping Hu are with the Guangdong–Hong Kong–Macao Joint Laboratory for Emotional Intelligence and Pervasive Computing, Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, Guangdong, China, and also with the School of Medical Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: ehomewang@ieee.org; huxp@smbu.edu.cn).

Junxin Chen is with the School of Software, Dalian University of Technology, Dalian 116621, China (e-mail: junxinchen@ieee.org).

Thippa Reddy Gadekallu is with the Research and Development, Zhongda Group, Jiaxing 314312, Zhejiang, China, also with the Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon, also with the School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India, also with the College of Information Science and Engineering, Jiaxing University, Jiaxing 314001, China, and also with the Division of Research and Development, Lovely Professional University, Phagwara 144001, India (e-mail: thippareddy@ieee.org).

Digital Object Identifier 10.1109/TCE.2023.3324081

I. Introduction

According to the International Organization of Motor Vehicle Manufacturers, there are currently more than 1.2 billion automobiles on the road, with that figure rising quickly in recent years [1]. Numerous automobiles on the road result in a considerable deal of accidents, traffic congestion, and other problems that cost lives and money. Researchers introduce the Internet of Vehicles (IoVs) from Intelligent Transport Systems to achieve traffic condition monitoring, traffic information interaction, and other benefits to help with the aforementioned issues [2].

Vehicle-to-everything (V2X) refers to the ability of various devices in IoVs to communicate with one another, including vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P), vehicle-to-infrastructure (V2I), and so on. These communications are being implemented via cellular V2X, millimeter wave, dedicated short-range communication, and other technologies. Despite the ease that multi-device and multi-way communication offers, it is highly vulnerable to hacker intrusions and attacks. Specifically, passive attacks like Probe can steal users' private information, while active attacks like Denial of Service (DoS) can disrupt the process of information interaction, drastically diminishing driving safety. As a result, it is vital to detect intrusions and attacks during IoV communications.

Intrusion detection systems (IDS) have a long history of being used to ensure the security of networks and all associated assets in cyberspace, alongside firewalls and antivirus. Among the various types of IDS, network-based IDS (NIDS) is now commonly used due to its great efficiency [3], and it would be used in our work as well. NIDS is typically placed on a network to offer security for all devices on that network by continuously monitoring network traffic for malicious and suspicious behavior. Combining this procedure with Anomaly-based IDS, a description of typical behavior is established, and behaviors that deviate from this definition are seen as anomaly behaviors [4], or intrusions.

Although many studies have been planned for NIDS, two major constraints remain. First, traditional Vehicles-to-Cloud (V2C) is not efficient or privacy-protecting. As the primary mode of communication in IoVs, mobile users, usually with low computational power, must upload their data directly to the cloud, which can compromise privacy and efficiency. Second, most network traffic in the real world is normal, with attacks accounting for a minor portion, as reflected in IDS datasets. Unfortunately, many studies are affected by this, making it difficult to detect low-probability attacks. As a result, for IoVs, we presented a Federated Learning and Edge Cloud communication architecture (**FL-EC**), as well as a Feature Select Transformer (**FSFormer**) model for robust intrusion detection. Our main contributions are summarized as follows: (i) We propose **FL-EC** communication architecture based on IoV data distribution, with communications devices including mobile users, edge devices, and cloud centers. We improve data transmission efficiency using edge computing and safeguard data privacy with Privacy Differential (DP) and FL. (ii) We choose Transformer, which can well extract features, as the base model. Then we enhance Transformer, named **FSFormer**, to adapt NIDS problem by observing its datasets and training processes. (iii) We conduct extensive experiments on two modern IDS datasets. The results indicate that the proposed model outperforms five baseline models in almost all settings and improves the detection rate of low-probability attacks.

The rest of the paper is structured as follows: Section II introduces existing studies of IDS according to their categories; Section III gives a description of the problem to be solved; Section IV illustrates the proposed FL-EC and FSFormer in detail; Section V contains extensive experiments; and Section VI summarises the paper and introduce future works.

II. Related Work

Existing NIDS methods can be split into four categories: statistic-based IDS, knowledge-based IDS, machine learning-based IDS, and deep learning-based IDS, which we will go over in order below. At the end of the section, we also introduce some intrusion detection frameworks for IoT/IoVs.

A. Statistic-Based IDS

Instead of network traffic, Statistic-based IDS focuses on statistical indicators such as median, mean, pattern, and standard deviation. Reference bib5 propose a Multivariate IDS that uses two or more measures to discover the relationship between variables. However, it has trouble estimating the distribution of high-dimensional data [6]. References [7] and [8] use time series models to discover anomalies by examining abrupt changes in time series data. These methods are straightforward and can meet the real-time requirement, but they demand a lot of statistical expertise and have low accuracy.

B. Knowledge-Based IDS

Knowledge-based IDS, also known as the expert system technique, requires the creation of a knowledge base including aspects of all lawful traffic. Any operation that differs from the features is considered an incursion. Finite state machine [9], description language [10], expert system [11], and signature analysis [12] are all related methods. These strategies can lower the false positive rate, but they require constant knowledge upgrades, which is challenging and time-consuming.

C. Machine Learning-Based IDS

ML-based intrusion detection systems can capture characteristics and patterns in network traffic data and are often classified as supervised learning or unsupervised learning. Supervised learning predicts labels by learning patterns in traffic-label pairs, primarily using Naïve Bayes [13], Support Vector Machines [14], Hidden Markov Model [15], and K-Nearest Neighbors [16]. Reference [17] argue that tree-based models such as Decision Tree [18], Random Forest [19], and XGboost [20] better perform than other ML algorithms in intrusion detection. Unsupervised learning groups together network traffic with similar patterns, primarily using K-means [15] and Hierarchical Clustering [21]. Although these methods are efficient, they rely on manual feature selection and perform poorly on large datasets.

D. Deep Learning-Based IDS

DL-based IDS is more accurate and effective for having a deeper hierarchical structure than ML-based IDS [22]. Many deep learning algorithms are now used to identify intrusion: [23] propose a Gate Recurrent Unit (GRU) model as well as a classifier based on Multilayer Perceptron (MLP) and softmax. Deep AutoEncoder (AE) and a random forest classifier are combined in [24]. Reference [25] employ Deep Belief Network (DBN) to extract features, which are subsequently fed into an ensemble SVM to get vote results. Reference [26] initially reduces the dimension with Principle Component Analysis (PCA) or AE, then converts the data from 1D to 2D and feeds them into convolution networks. Although these algorithms attain great accuracy, they suffer from unbalanced data and have difficulties detecting low-probability attacks. Transformer [27] is now widely utilized in many industries and is regarded to be superior to typical deep learning methods in feature extraction from text series data [28]. And, more recently, some studies have attempted to apply Transformer to IDS: [29], [30], and [31] use Transformer, VisionTransformer [32] and Informer [33] for IDS, respectively. However, they just transplant Transformer into IDS, which may restrict its capacity.

E. Intrusion Detection Framework for IoT

Many studies also design privacy-preserving frameworks to assist IDS. Reference [34] develop a blockchain module to securely transmit the data between IoV-RSU-Cloud. Reference [35] propose a hierarchical blockchain-based FL framework to improve IoT security posture. Reference [31]

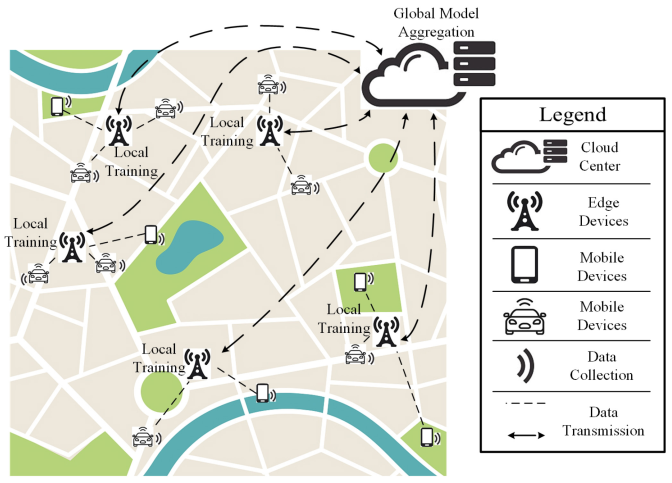


Fig. 1. The Overview of FL-EC Architecture. In FL-EC, mobile devices collect and encrypt data before uploading it to edge devices like RSUs for training. Edge devices are the clients of FL, uploading parameters after local training, while the cloud center is the server of FL, gathering parameters and training globally.

TABLE I
THE MAIN ABBREVIATIONS IN THIS PAPER

Abbreviations	**Descriptions**
IoVs	Internet of Vehicles
IoT	Internet of Things
V2X	Vehicle-to-everything
DoS	Denial of Service
IDS	Intrusion detection systems
FL	Federated Learning
EC	Edge Cloud
FSFormer	Feature Select Transformer
CVs	Connected Vehicles
GDP	Gaussian Differential Privacy
FA	Feature Attention
MHA	Multi-head Attention
FFN	Feed-Forward Network
X_{emb}	Feature Embedding
X_{rep}	Feature Representation
X_{ext}	Feature Extraction

design a framework that integrates edge computing and blockchain. Reference [36] propose a blockchain with a voting consensus method based on smart contracts to ensure secure communication among IoVs components.

III. PROBLEM DEFINITION

In IDS, the input is the features of network traffic packages, which are sequential text data such as IPs, ports, protocols, and so on, and the output is the categories of the input traffic, which include normal and all types of assaults such as DoS, Botnet, and so on. To summarize, IDS must determine the type of each network traffic package based on its features. Set the input to be $\mathbf{X} = (x_1, x_2, \dots, x_T) \in \mathbb{R}^T$ and the output to be y, then the problem can be formulated as:

$$y = f(\mathbf{X}), \tag{1}$$

where T is the total number of features and f is a trainable mapping that enables accurate classification of the network traffic.

IV. METHODOLOGY

As previously stated, we proposed **FL-EC-FSFormer** for privacy-preserving data transmission and effective intrusion detection. Figure 1 and Figure 2(a) illustrate overviews of the **FL-EC** architecture and **FSFormer**, respectively. The following specifically introduces our approach. The main abbreviations are shown in Table I.

A. FL-EC Architecture

According to V2X, connected vehicles (CVs) communicate with devices in IoVs, at which time they can collect network traffic data for IDS. However, CVs typically lack the computing capacity [37] required to enable effective IDS models. In conventional V2C, CVs send their data directly to the cloud for training, which takes time, has large connection costs, and puts a lot of strain on the cloud. Edge cloud, a technology that permits calculations to be performed at the network's edge [38], has recently been presented for this problem. We include it in IoVs so that CVs may transfer data to adjacent edge devices such as RSUs for basic training, considerably improving efficiency and practicability. Furthermore, CVs would apply Gaussian Differential Privacy noise to raw traffic data before sending it to edges to preserve their privacy:

$$M(D) = f(D) + Y \tag{2}$$

$$P[M(D) \in S] \le e^{\epsilon} P\big[M\big(D'\big) \in S\big] + \delta \tag{3}$$

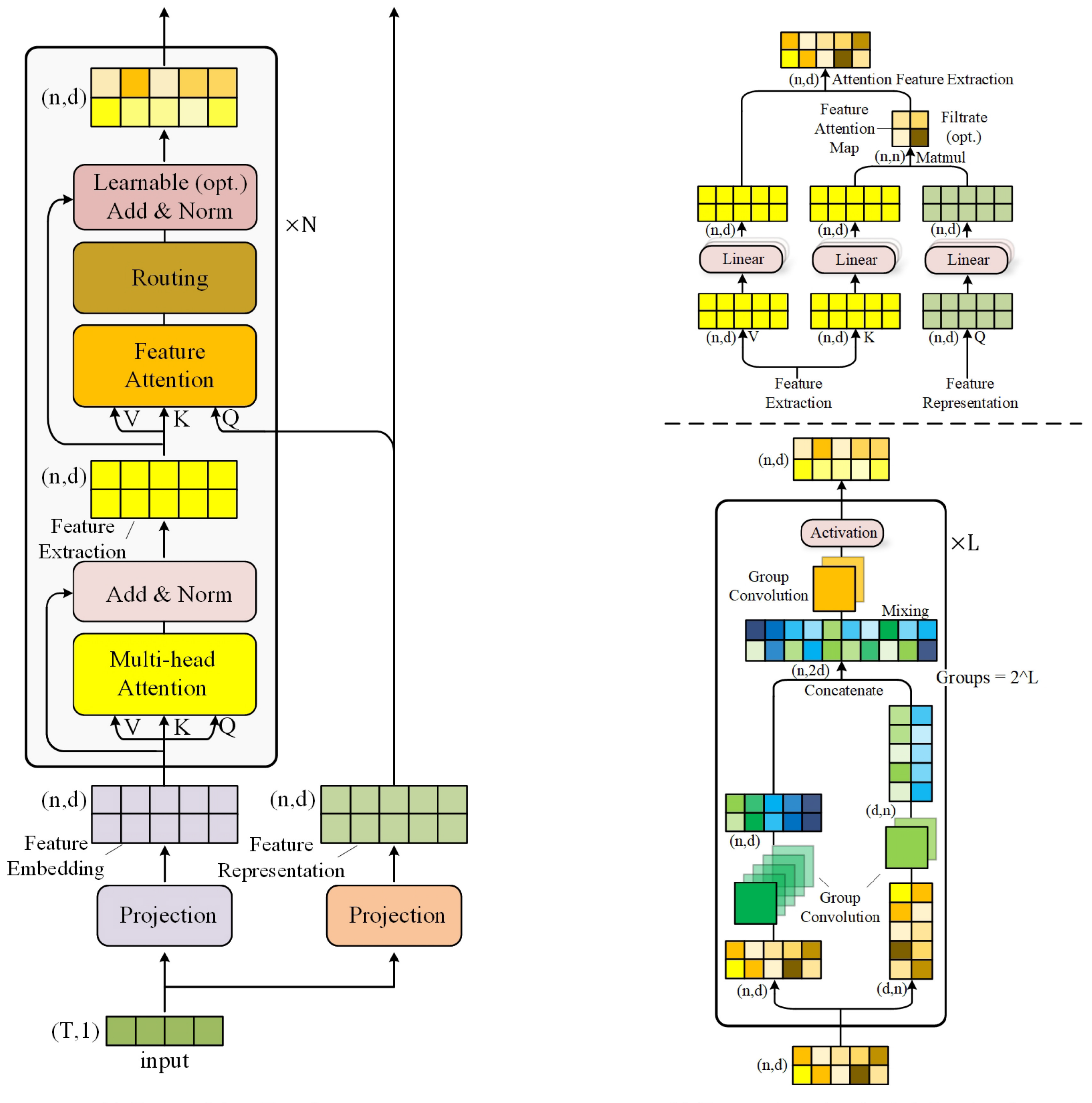
where $M(D)$ is the result after adding noise, $Y \sim N(0, \sigma^2)$ is the noise, and $\sigma > \sqrt{2ln(1.25/\delta)}\Delta f/\epsilon$, δ is a slack variable, and ϵ is privacy bucket that is negatively correlated with noise. Actually, ϵ is the bound of the max divergence of two distributions $M(D)$ and $f(D)$. Generally, smaller ϵ means better privacy as well as larger noise. The utility decreases and the processed data further deviates from the original data as privacy bucket ϵ decreases. Different values of ϵ are used in the following experiment to examine models' capability to extract features. Besides, slack variable δ is used to loosen the condition of Differential Privacy and improve its practicability. We call this (ϵ, δ)-GDP.

However, with limited communication ranges, edge devices might not gather enough information to provide precise training. Consequently, we present FL for safe data centralization and global training. In our FL framework, edge devices act as clients, uploading parameters through local training, whilst the cloud acts as a server, gathering parameters, eliminating vicious ones, and then training globally with FedAvg [39] algorithm. The above process is expressed as:

$$\omega_{t+1}^{k} \leftarrow \omega_{t}^{k} - \eta \nabla f\Big(\omega_{t}^{k}\Big), \tag{4}$$

$$\omega_{t+1} \leftarrow \sum_{k=1}^{K} \frac{K}{N} \omega_{t+1}^{k}, \tag{5}$$

where η is the learning rate of clients, N and K are the number of all clients and the chosen clients respectively, and t is the communication round. Later, the global models are



(a) Feature Select Transformer (b) Feature Attention (top) & Routing (bottom)

Fig. 2. **(a)** The Overview of FSFormer. We replace Embedding and Positional encoding with Projection, add Feature Attention to select key features, and replace the Position-wise Feed-Forward with Routing to realize two-dimensional transformations and a deeper but less-parameter structure. **(b)** The Process of Feature Attention and Routing. In Feature Attention, we compute the similarity scores by the scaled dot product between feature extraction and feature representation and drop some scores that are lower than the threshold. In Routing, we employ three group convolutions in each layer for two-dimensional transformations and different dependencies extraction.

distributed from cloud servers to edge devices, and then to CVs inside the ranges of the edge devices, allowing CVs to detect intrusion precisely throughout their communications. In a word, we design a data transmission and training process based on the decentralized data distribution in IoVs. Compared to traditional V2C, FL-EC architecture is more efficient and privacy-protecting. Details about the implementation process and communication costs are not provided since we pay more attention to the intrusion detection model.

B. FSFormer

FSFormer is designed to be deployed in the cloud server, however, directly transmitting data from data sources, i.e., CVs, is costly and unsafe as mentioned above. Therefore, we propose the FL-EC framework and train FSFormer on edge devices with data from near CVs. Then, the parameters of FSFormer are sent to the cloud server for aggregation and finetuning. Finally, the well-trained FSFormer would be sent to CVs to detect attacks and intrusions.

We now introduce our intrusion detection model FSFormer in detail. Yin et al. [40] hold the view that some features in raw network traffic data are duplicated, and they employ Random Forest to select the most important ones for training. We conduct a series of mini-tests for verification, and some results are presented in Table II, where "Ratio" is the ratio of reserved features to all features, and "Fuzzers" is one of the attacks with a low probability. In the table, the overall performance declines when the ratio of features decreases, which meets the expectation, however, the detection rate of "Fuzzers" rather increases. We deduce that some features being dropped when the ratio of the features decreases may impede the detection

TABLE II
SAME MODEL LEARNS WITH DIFFERENT FEATURE RATIO

Ratio	Accuracy	F1	Fuzzers Accuracy	Fuzzers F1
0.9	0.9873	0.9644	0.9907	0.1332
0.7	0.9826	0.9596	0.9910	0.2272

of "Fuzzers" but benefit "normal" or other attacks, and that explains the results mentioned above to some extent. Through further observations, we discover that, in summary, removing certain features reduces overall accuracy while increasing the detection rate of particular attacks, which means many attacks depend primarily on a few key features. Unfortunately, different attacks don't necessarily share the same key features for which directly removing certain features before training might reduce the detection precision, and it is impossible to manually reserve the key features for every attack. On this basis, we develop FSFormer, as shown in Figure 2(a), to automatically select key features for every traffic packet (note that the detection of an attack lies on the extraction of features in a traffic packet, so our model is designed to weight the features in every traffic packet and focus on the important ones to make a better prediction). In **FSFormer**, we discard Decoders because rebuilding the series is unnecessary. Meanwhile, we design Projection Module, Feature Attention Module, and Routing Module, which will be discussed in detail below.

1) Projection: In this module, we first discard the Positional Encoding Module since the traffic features in the input are independent of the position. Next, for better representation, we project both two dimensions of input. In particular, if the input is $\mathbf{X} \in \mathbb{R}^{T\times 1}$, use a linear layer to transfer the feature length T to n and a 1d-convolutional layer to expand the feature depth from 1 to d:

$$Projection(\mathbf{X}) = Conv1d(\mathbf{WX} + \mathbf{b})_2, \tag{6}$$

where $Projection(\mathbf{X}) \in \mathbb{R}^{n\times d}$ and $Conv1d(x)_z$ means convolution on z-th dimension of x.

We employ distinct Projection layers to obtain two input projections, namely feature embedding and feature representation. The former is the input of the first self-attention module for feature extraction, while the latter is the query of the Feature Attention Module in each Encoder layer as the baseline of feature selection.

2) Feature Attention: Features are weighed in this module, and all features are reserved to guarantee the entirety accuracy, while those with high weights play a major role in the dependence extraction and representation. Before Feature Attention (FA), feature embedding $\mathbf{X}_{emb}$ is fed into the self-attention module:

$$\begin{aligned}\mathbf{X}_{ext} &= \mathrm{MHA}(Q, K, V)\\ &= Concat(head_1, \ldots, head_h)W^O, \end{aligned}\tag{7}$$

$$head_i = \Phi\left(\frac{Q_i K_i^T}{\sqrt{d_{ki}}}\right)V_i, \tag{8}$$

where $\mathbf{X}_{ext}$ is feature extraction that contains key information of input, MHA is multi-head attention, Φ is Softmax function, $Q_i, K_i, V_i = \mathbf{X}_{emb}W_i^Q, \mathbf{X}_{emb}W_i^K, \mathbf{X}_{emb}W_i^V$ and $W_i^Q \in \mathbb{R}^{d\times d_k}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_k}$ and $W^O \in \mathbb{R}^{hd_k\times d}$ are projection matrices. Then, $\mathbf{X}_{ext}$ is fed into FA together with feature representation $\mathbf{X}_{rep}$ for weighting $\mathbf{X}_{ext}$, which is depicted in the top figure of Figure 2(b). FA, similar to MHA, can be defined as:

$$\mathrm{FA}\left(\hat{Q}, \hat{K}, \hat{V}\right) = Concat(head_1, \ldots, head_h)W^{\hat{O}}, \tag{9}$$

$$head_i = \Gamma\left(\Phi\left(\frac{\hat{Q}_i \hat{K}_i^T}{\sqrt{d_{ki}}}\right)\right)\hat{V}_i, \tag{10}$$

where Γ is the filtrate function, and $\hat{Q}_i, \hat{K}_i, \hat{V}_i = \mathbf{X}_{rep}W_i^{\hat{Q}}, \mathbf{X}_{ext}W_i^{\hat{K}}, \mathbf{X}_{ext}W_i^{\hat{V}}$. We obtain the similarity scores by computing the scaled dot product of queries, i.e., $\mathbf{X}_{rep}W^{\hat{Q}}$, and keys, i.e., $\mathbf{X}_{ext}W^{\hat{K}}$. The scores express the interdependence of feature extraction and feature representation, and features with high scores are regarded as significant. We further enhance key features' influence by dropping scores whose ratios to the average are lower than threshold ν and padding them with 0. At last, we multiply values, i.e., $\mathbf{X}_{ext}W^{\hat{V}}$, by the similarity scores to make feature extraction focus on significant features.

In summary, multi-head attention is first used to learn the correlation and dependence between features and output the extraction of features that contain the captured information, namely feature extraction $\mathbf{X}_{ext}$. However, $\mathbf{X}_{ext}$ represents the comparative weights between features instead of global weights for predictions. Thus, feature representation $\mathbf{X}_{rep}$, a learnable mapping of the input that describes input features, is introduced, and a dot product between $\mathbf{X}_{rep}$ and $\mathbf{X}_{ext}$ in feature attention is calculated for the similarity scores, which represent the degree of relation between the input features and extracted information that helps with prediction. As a result, the elements in $\mathbf{X}_{rep}$ with higher similarity scores are thought to be the features that are more important for predictions. After filtrating the scores lower than the threshold ν, the key features are reserved and weight the extracted information, i.e., feature extraction $\mathbf{X}_{ext}$. It is worth noting that $\mathbf{X}_{rep}$ is reused in each Encoder layer to weight feature extraction in different levels.

3) Routing: The Routing Module replaces the Position-wise Feed-Forward Network (FFN) in Transformer. FFN is thought to be able to fuse the position-wise information (i.e., the dimension of d_{model}) for each word vector and further extract their features. In our work, however, we not only need to fuse the position-wise information but also need to further capture the dependencies between features. Inspired by [41], we employ several two-dimensional 1d group convolution layers, which are deeper, have fewer parameters, and can learn more complex representations than FFN, to meet our requirements. In a Routing layer (see the bottom of Figure 2(b)), the input is imported into two group convolutions: one routes the length, while the other routes the depth. The two output matrices' dimensions remain unchanged. Then, on the second dimension, or depth, two matrices are concatenated into one, and features of this new matrix are also shuffled in this dimension. Following that, the matrix is transferred from $2d$ to d using the third group convolution and activated using the

TABLE III
THE DETAILS OF TWO CHOSEN DATASETS

Datasets	UNSW-NB15		CSE-CIC-IDS2018	
No.Attacks	9		7	
No.Features	49		83	
Probabilities of Types	Normal	87.35%	Normal	79.80%
	DoS	0.64%	Bot	2.34%
	Reconnaissance	0.55%	Infiltration	1.33%
	Shellcode	0.059%	HeartBleed	0.00082%
	Exploits	1.75%	Web	0.0073%
	Fuzzers	0.95%	BruteForce	3.15%
	Generic	8.48%	DoS	5.36%
	Backdoor	0.092%	DDoS	8.00%
	Analysis	0.11%	-	-
	Worms	0.0069%	-	-

GELU. This process is expressed below:

$$Routing\left(\mathbf{X}_{fa}\right) = GELU\left(Conv1d\left(\mathbf{X}, groups = 2^l\right)_2\right), \quad (11)$$

$$\mathbf{X} = Shuffle(Concat(\mathbf{X}_n, \mathbf{X}_d)), \quad (12)$$

$$\mathbf{X}_n = Conv1d\left(\mathbf{X}_{fa}, groups = 2^l\right)_1, \quad (13)$$

$$\mathbf{X}_d = Conv1d\left(\mathbf{X}_{fa}, groups = 2^l\right)_2, \quad (14)$$

where $0 < l \leq L$ is the current number of layers while the L is the total number of layers. We use the groups that grow exponentially as the layers become higher to reduce the number of parameters, extract different dependencies in each layer, and reduce overfitting [42].

V. EXPERIMENTS

A. Experimental Settings

1) Datasets: We choose two intrusion datasets that can better reflect network traffic and modern low-footprint attacks than classical datasets such as KDDCUP'99 [43] and NSLKDD [44] in the current network threat environment.

a) UNSW-NB15: This dataset is created by the Australian Center for Cyber Security [45]. Its raw network packets are generated by the IXIA PerfectStorm tool, which combines genuine modern normal activities with synthetic modern attack characteristics. It was released in 2015 and has about 130k data volume.

b) CSE-CIC-IDS2018: This dataset is jointly created by the Communications Security Establishment and Canadian Institute of Cyber Security in 2018 [46]. It employs the notion of profiles to build datasets that contain precise descriptions of intrusions as well as abstract distribution models.It was released in 2018 and has a size of 1200k data. Details of these two datasets are displayed in Table III. It can be discovered that the datasets are severely unbalanced, with some attacks having extremely low appearance probabilities, making them difficult to detect.

2) Baselines:

- *Decision Tree* is a commonly used supervised machine learning method, including several generation algorithms such as ID3, C4.5, and C5.0. Song and Ying [18], Stein et al. [47], and Rai et al. [48] have employed Decision Tree for effective intrusion detection.
- *Random Forest* integrates many decision trees using Bagging in ensemble learning, improving the performance on high dimensional data. It is now widely used for accurate network intrusion detection [19], [49], [50].
- *LightGBM* (Light Gradient Boosting Machine) is a framework to realize Gradient Boosting Decision Tree algorithm and overcomes some defects of XGBoost. Jin et al. [51] and Liu et al. [52] utilize LightGBM to achieve fast and precise intrusion detection.
- *Transformer* [27] is a classical model designed for Machine Translation and is gradually applied to IDS recently [29], [30].
- *Informer* [33] is an enhanced version of Transformer for long sequence time-series forecasting problems and employed by Abdel-Basset et al. [31] for IDS.
- *Routing Transformer* [53] learns dynamic sparse attention patterns that avoid allocating computation and memory to attend to content unrelated to the query of interest.
- *BiDLSTM* [54] is one of the state-of-the-art models in IDS, successfully reducing the false alarm rate of attacks.
- *CNNAE-IDS* [26] is also a state-of-the-art model in IDS. It utilizes Auto Encoder for data reduction and Lenet-5 for anomaly detection.

3) Evaluation Metrics: Four metrics are used to evaluate the performance, and they are all defined based on the following attributes: *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)*, and *False Negative (FN)*. Metrics include:

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

$$\text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \text{Detection Rate} = \frac{TP}{TP + FN},$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

4) Implementation Details: For data preprocessing, we first drop timestamps, IPs, and ports. Then, we swap out those features that have several categories with their one-hot encoding. Finally, we normalize the raw data with Z-Score Normalization. We input all elements in raw data except timestamps, IPs, and ports into the models for training.

As we mentioned above, we don't work on the details of FL-EC framework, thus the experiments are not conducted on real edge-cloud devices. To simulate the FL-EC architecture, we set the number of clients and vehicles to 10 and 50, and we randomly allocate the same volume of data to clients and train them separately, aggregating model parameters at a GPU in the next. And we evenly distribute 60% and 20% of data into vehicles as the training set and testing set, respectively. We add Gaussian Differential Privacy noise in each vehicle's training set for privacy protection. The rest data is allocated to the server for validation. Moreover, we conduct centralized training using around 5% of the entire dataset to examine model performance in the absence of data.

TABLE IV
Overall Performance of Centralized Learning Under (9, 10^{-5})-GDP Noise

Dastaset	**CIC-CSE-IDS2018**				**UNSW-NB15**			
Model	**Training Set**		**Testing Set**		**Tranining Set**		**Testing Set**	
	Acurracy	F1	Acurracy	F1	Acurracy	F1	Acurracy	F1
FSFormer (ours)	**0.9710**	**0.9582**	**0.9657**	**0.9461**	**0.9979**	**0.9927**	**0.9883**	**0.9687**
Informer	0.9419	0.9125	0.9640	0.9416	0.9780	0.9491	0.9790	0.9509
Transformer	0.9399	0.9104	0.9610	0.9375	0.9846	0.9598	0.9853	0.9605
R_Transformer	0.9464	0.9193	0.9645	0.9419	0.9859	0.9648	0.9862	0.9652
CNNAE-IDS	0.9302	0.8977	0.9623	0.9396	0.9754	0.9503	0.9823	0.9577
BiDLSTM	0.9572	0.9359	0.9654	0.9444	0.9887	0.9680	0.9879	0.9672
Decision Tree	-		0.9090	0.8926	-		0.9658	0.9608
Random Forest	-		0.9492	0.9404	-		0.9716	0.9660
LightGBM	-		0.9481	0.9398	-		0.9700	0.9654

TABLE V
Performance of Detecting Attacks With Probabilities Close to or Less Than 1% Under (9, 10^{-5})-GDP Noise in CIC-CSE-IDS2018 Dataset

Dastaset	**CIC-CSE-IDS2018**							
Model	**Training Set**						**Testing Set**	
	Infiltration		Injection		Web		Infiltration	
	Recall	F1	Recall	F1	Recall	F1	Recall	F1
FSFormer	**0.2939**	**0.4275**	**0.6667**	**0.6667**	**0.5000**	**0.6445**	**0.0711**	**0.1052**
Informer	0.0004	0.0009	0		0		0	0
Transformer	0.0054	0.0107					0.0010	0.0020
R_Transformer	0.0224	0.0430					0.0030	0.0059
CNNAE-IDS	0	0					0	0
BiDLSTM	0.0994	0.1677					0.0400	0.0575
Decision Tree	-						0	0
Random Forest	-						0	0
LightGBM	-						0.0112	0.0112

For training setups, we use Adam optimizer with $\beta = (0.9, 0.98)$. We only utilize the learning rate warm-up, which depends on the size of the model, in the first epoch and gradually decrease the learning rate in the rest epochs. We set the batch size to 32 and the max epochs of centralized learning to 12. As for the parameters of the proposed model, we set the number of Encoders, i.e., N, as 6. In an Encoder, we set $n = 320$, $d = 128$, heads of self-attention $h_1 = 4$, heads of feature-attention $h_2 = 4$, the filtrate threshold $\nu = 0.05$, and the layers of Routing $L = 6$. In the meantime, we carefully tune the parameters of baseline models to get their best performance. Moreover, we fix the seed of data shuffling for every model to hold a fair comparison. All models are trained or tested on the GeForce RTX 4090 GPU with a memory of 24GB. The results of deep learning models are the average of three runs.

B. Comparison Experiments

Eight baseline models are compared with the proposed model in centralized training while the five deep learning methods are further compared in Federated learning scenes, and the results of centralized learning are shown in Table IV, Table V, Table VI, and Table VII, while the results of FL are shown in Table VIII, Table IX, and Table X. Some attacks that meet the condition are **not shown in** Table VI and Table VIII because all the models fail to get an F1-Score that is greater than 0. The results in the testing set are worse than the training set, that's the reason why **some attacks appear only in the training set**. The best results are boldfaced.

1) Centralized Learning: As mentioned above, it is possible that the edge devices can not gather enough data for the mobility of vehicles in a real-world scenario, we thus employ only 5% data volume of the entire dataset to examine the performance of models in the absence of data. The results of the training set are presented to show the models' fitting ability in the extremely unbalanced dataset as well as generalization ability by comparing testing and training sets.

According to the results of Table IV, the proposed model performs best under all the metrics, especially in the training set. Besides, our model has an average advantage of 1.5% and 2.5% over other models on "accuracy" and "F1" respectively. It's worth noting that the metric "accuracy" has a bias in measuring the performance in these two datasets because true negative samples of low-probability attacks, which are easy to be categorized by models, account for the majority of all samples, resulting in a high "accuracy" for each low-probability attack. "F1", in the contrast, can better depict the ability to detect the attacks with low probabilities. Therefore, higher "F1" scores demonstrate that the proposed model is robust in the severely unbalanced dataset. From Table V and Table VI, it is obvious that our model can best detect nearly all the attacks with low probabilities. In the training set, its detection rates, i.e., "recall", of attacks with probabilities less than 0.1%, including Injection, HeartBleed, Web, Backdoor, and Shellcode, are greater than 50% (the best one even reaches 67.21%). And other models' detection rates for these five attacks are all 0. To some extent, this alleviates the challenge of detecting low-probability attacks in IDS. Then, we raise the

TABLE VI
PERFORMANCE OF DETECTING ATTACKS WITH PROBABILITIES CLOSE TO OR LESS THAN 1% UNDER (9, 10^{-5})-GDP NOISE IN UNSW-NB15 DATASET

Dastaset	**UNSW-NB15**													
Model	Analysis		Backdoor		DoS		Fuzzers		Reconnaissance		Shellcode		Worms	
	Recall	F1	Recall	F1	Recall	F1	Recall	F1	Recall	F1	Recall	F1	Recall	F1
	Training Set													
FSFormer	**0.7068**	**0.7787**	**0.6721**	**0.7476**	**0.8314**	**0.8280**	**0.8408**	**0.8404**	**0.8281**	**0.8341**	**0.5446**	**0.6420**	**0.1250**	**0.1879**
Informer	0	0	0	0	0.0152	0.0263	0.0015	0.0030	0	0	0			
Transformer	0	0	0	0	0.0904	0.1216	0.1175	0.1760	0.0895	0.1282				
R_Transformer	0.0025	0.0049	0	0	0.2178	0.2661	0.2419	0.2988	0.1378	0.1977				
CNNAE-IDS	0	0	0	0	0.0222	0.0363	0.1178	0.1822	0.0005	0.0010				
BiDLSTM	0	0	0	0	0.0861	0.1447	0.2023	0.2865	0.3283	0.3619				
	Testing Set													
FSFormer	0.0711	0.0763	**0.0056**	**0.0087**	**0.2136**	**0.2095**	**0.2623**	**0.2982**	**0.5269**	0.4331	0			
Informer	0	0	0		0.0052	0.0099	0.0014	0.0027	0	0				
Transformer	0	0			0	0	0	0	0.0191	0.0326				
R_Transformer	0.0164	0.0303			0.0671	0.0964	0.1482	0.2081	0.2712	0.3139				
CNNAE-IDS	0	0			0	0	0.0690	0.1178	0	0				
BiDLSTM	0	0			0.0124	0.0240	0.0847	0.1393	0.4540	0.4203				
Decision Tree	0	0			0.0326	0.0593	0.0995	0.1451	0.0260	0.0446				
Random Forest	0	0			0	0	0.1149	0.1875	0.4141	**0.4356**				
LightGBM	**0.1429**	**0.1391**			0.0023	0.0046	0.2586	0.2953	0.1094	0.1732				

TABLE VII
OVERALL PERFORMANCE OF CENTRALIZED LEARNING UNDER (3, 10^{-5})-GDP NOISE

Dastaset	**CIC-CSE-IDS2018**				**UNSW-NB15**			
Model	**Training Set**		**Testing Set**		**Tranining Set**		**Testing Set**	
	Acurracy	F1	Acurracy	F1	Acurracy	F1	Acurracy	F1
FSFormer (ours)	**0.8906**	**0.8436**	**0.8988**	**0.8471**	**0.9907**	**0.9842**	**0.9860**	**0.9630**
Informer	0.8507	0.7739	0.8667	0.8045	0.8985	0.8522	0.9105	0.8648
Transformer	0.8438	0.7563	0.8493	0.7551	0.9567	0.9317	0.9761	0.9499
R_Transformer	0.8550	0.7813	0.8654	0.7952	0.9503	0.9568	0.9836	0.9586
CNNAE-IDS	0.8372	0.7341	0.8809	0.8129	0.9020	0.8658	0.9375	0.9071
BiDLSTM	0.8748	0.8173	0.8790	0.8287	0.9751	0.9532	0.9743	0.9504
Decision Tree	-		0.8412	0.8040	-		0.9459	0.9473
Random Forest	-		0.8245	0.7614	-		0.9534	0.9379
LightGBM	-		0.8549	0.8320	-		0.9663	0.9596

noise level to see how it affects our model, and the results are displayed in Table VII. We can see that our model excels in all circumstances and widens the gap with other models when compared to low noise level outcomes (the average overshoots on "accuracy" and "F1" are 4.6% and 6.7%, while these numbers in the low noise level are 1.5% and 2.5%), indicating that our model has superior anti-noise performance.

2) Federated Learning: The proposed model remains the best model in all settings for overall performance in FL, which can be found in Table IX and Table X. When compared to centralized learning under the same noise, our model further exceeds the second-best models. Meanwhile, when the number of candidates changes, our model is more stable than other models, allowing it to better adapt to the edge cloud of IoVs, where connections between mobile users and edge devices are unfixed and regularly changed. In most circumstances, our model works well in detecting low-probability attacks, as illustrated in Table VIII. As expected, decentralized training significantly lowers the detection rates of most models. However, the detection rates of DoS, Fuzzers, and Reconnaissance in our model increase. We believe this is because our model can fully utilize the additional data (The data volume in FL exceeds the small sample centralized learning performed above).

C. Ablation Studies

We conduct ablation studies to explain the effect of each module in FSFormer, and try to figure out the ways to improve the model in future work. We take out a module and reserve

TABLE VIII
PERFORMANCE OF DETECTING ATTACKS WITH PROBABILITIES CLOSE TO OR LESS THAN 1% IN FL UNDER (9, 10^{-5})-GDP NOISE

Dataset	Candidates	Model	FSFormer (ours)		Informer		Transformer		Routing Transformer		CNNAE-IDS		BiDLSTM	
		Metric	Recall	F1	Recall	F1	Recall	F1	Recall	F1	Recall	F1	Recall	F1
CSE-CIC-IDS2018	6	Infiltration	**0.0227**	**0.0375**	0.0145	0.0224	0.0127	0.0212	0	0	0.0152	0.0252	0.0187	0.0202
	3	Infiltration	0.0160	0.0259	0.0061	0.0108	0.0127	0.0207	0	0	**0.0261**	**0.0417**	0.0007	0.0012
UNSW-NB15	6	Analysis	0.0584	**0.0847**	0	0	0	0	**0.0639**	0.0420	0	0	0	0
		DoS	**0.5750**	**0.4035**	0.0003	0.0005	0.0256	0.0420	0.0073	0.0132	0.1112	0.0808	0	0
		Fuzzers	**0.5012**	**0.4034**	0.1898	0.0242	0.0331	0.0461	0.1821	0.0929	0.1287	0.1661	0.0555	0.0425
		Reconnaissance	**0.4667**	**0.4231**	0	0	0	0	0.0006	0.0012	0.0207	0.0370	0.2071	0.0605
		Worms	0	0	0	0	**0.3722**	**0.0013**	0	0	0	0	0	0
	3	Analysis	**0.0330**	**0.0559**	0	0	0	0	0.0216	0.0363	0.0071	0.0118	0	0
		DoS	**0.4568**	**0.2784**	0.0001	0.0003	0.2814	0.1419	0.0548	0.0797	0.0961	0.1063	0	0
		Fuzzers	**0.4959**	**0.4051**	0.0326	0.0424	0.1561	0.1270	0.0787	0.1297	0.2376	0.2409	0.0268	0.0409
		Reconnaissance	**0.4402**	**0.4072**	0	0	0.0950	0.0890	0.0800	0.0731	0.0569	0.0863	0.0746	0.0981
		Shellcode	0	0	0	0	0	0	**0.1978**	**0.0049**	0	0	0	0

TABLE IX
OVERALL PERFORMANCE OF FL (CANDIDATES=6)
UNDER (9, 10^{-5})-GDP NOISE

Dastaset	**CIC-CSE-IDS2018**				**UNSW-NB15**			
Model	**Testing Set**				**Testing Set**			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
FSFormer (ours)	**0.9649**	**0.9410**	**0.9484**	**0.9410**	**0.9880**	**0.9734**	**0.9684**	**0.9701**
Informer	0.9363	0.9110	0.9148	0.9093	0.9369	0.8908	0.8955	0.8887
Transformer	0.9392	0.9143	0.9204	0.9146	0.9430	0.9116	0.8866	0.8928
R_Transformer	0.9036	0.8540	0.8789	0.8449	0.9607	0.9181	0.9419	0.9248
CNNAE-IDS	0.9487	0.9302	0.9291	0.9247	0.9816	0.9528	0.9643	0.9566
BiDLSTM	0.9537	0.9146	0.9327	0.9213	0.9839	0.9576	0.9635	0.9583

TABLE X
OVERALL PERFORMANCE OF FL (CANDIDATES=3)
UNDER (9, 10^{-5})-GDP NOISE

Dastaset	**CIC-CSE-IDS2018**				**UNSW-NB15**			
Model	**Testing Set**				**Testing Set**			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
FSFormer (ours)	**0.9642**	**0.9377**	**0.9479**	**0.9394**	**0.9883**	**0.9726**	**0.9687**	**0.9692**
Informer	0.9578	0.9296	0.9396	0.9316	0.9253	0.9084	0.8479	0.8667
Transformer	0.9603	0.9329	0.9439	0.9355	0.8150	0.8583	0.7764	0.7697
R_Transformer	0.8669	0.8058	0.8386	0.7867	0.9838	0.9608	0.9598	0.9553
CNNAE-IDS	0.9481	0.9309	0.9285	0.9250	0.9846	0.9602	0.9657	0.9609
BiDLSTM	0.9609	0.9324	0.9442	0.9355	0.9775	0.9468	0.9645	0.9539

TABLE XI
OVERALL PERFORMANCE IN ABLATION STUDIES
UNDER (9, 10^{-5})-GDP NOISE

Dastaset	**CIC-CSE-IDS2018**							
Model	**Training Set**				**Testing Set**			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
FSFormer	**0.9710**	**0.9591**	**0.9611**	**0.9582**	**0.9657**	**0.9453**	**0.9522**	**0.9461**
-w/o MHA	0.9303	0.8987	0.9055	0.9001	0.9313	0.9325	0.9035	0.9132
-w/o Projection	0.9659	0.9518	0.9545	0.9500	0.9597	0.9366	0.9442	0.9384
-w/o FA	0.9509	0.9253	0.9317	0.9235	0.9626	0.9368	0.9496	0.9413
-w/o Routing	0.9641	0.9475	0.9507	0.9453	0.9647	0.9428	0.9501	0.9440
Transformer	0.9399	0.9144	0.9198	0.9104	0.9610	0.9353	0.9460	0.9375

TABLE XII
PERFORMANCE OF DETECTING ATTACKS WITH PROBABILITIES CLOSE TO OR LESS THAN 1% IN ABLATION STUDIES UNDER (9, 10^{-5})-GDP NOISE

Dastaset	**CIC-CSE-IDS2018**							
Model	**Training Set**						**Testing Set**	
	Infiltration		**Injection**		**Web**		**Infiltration**	
	Recall	F1	Recall	F1	Recall	F1	Recall	F1
FSFormer	**0.2939**	**0.4275**	**0.6667**	**0.6667**	**0.5000**	**0.6445**	**0.0711**	**0.1052**
-w/o MHA	0.0296	0.0446	0	0	0	0	0.0190	0.0267
-w/o Projection	0.1743	0.2848	0.3333	0.2222	0.1111	0.1905	0.0240	0.0370
-w/o FA	0.0161	0.0313	0	0	0.1111	0.1667	0	0
-w/o Routing	0.1250	0.2127	0.3333	0.3333	0.1111	0.1905	0.0531	0.0821
Transformer	0.0054	0.0107	0	0	0	0	0.0010	0.0020

the other in FSFormer each time: 1) without MHA: MHA is removed and feature embedding is input into FA; 2) without Projection: Project module for feature representation is removed and feature extraction is fed into FA; 3) without FA: feature representation is directly added to feature extraction; 4) without Routing: Routing module is replaced with FFN. Results of ablation studies are presented in Table XI and Table XII, where the best are in boldface. Some attacks with probabilities close to or less than 1% are **not shown in** Table XII also because all the models fail to get an F1-Score that is greater than 0.

From Table XI, it can be found that the overall performance is worst without MHA, indicating that the MHA plays an important role in the extraction of features' information and the running of our model strongly depends on the information extraction. Then, FA is the module with the second-largest influence on overall performance. As we stated above, FA helps the model to focus on the key features of each packet, without which the performance can be disturbed by the redundant features. When FSFormer runs without Projection for feature representation, the performance in the testing set has an evident decline, which is close to Transformer. The replacement of FFN, i.e., Routing, seems not to provide a significant improvement. We design Routing for a sparse structure that can be deeper but has fewer parameters, and it does reduce the number of parameters but fails to improve the performance, which will be modified in the future.

As for the detection of low-probability attacks (see Table XII), FA is credited to the good performance. In Table XI, our model without FA, for example, performs much better than our model without MHA both in the training and testing set. However, these two models both work badly in detecting low-probability attacks, and the former is even worse in attack "Infiltration". Similar cases also happen to FSFormer without FA and Projection. In extremely unbalanced datasets, models tend to categorize samples into the types that appear most often to boost the overall accuracy (if a type accounts for 90%, categorizing all the samples into this type can obtain an accuracy of 90%). By selecting key features for each type, FA enhances the robustness of the model thus reducing the impact of unbalanced datasets to some extent.

VI. Conclusion

As the technology of IoV is constantly developing, the safety of communication in IoV becomes much more important. There is a large number of communication processes in IoV, which are easily exposed to the intrusion and attacks of hackers, threatening driving privacy and safety. Consequently, we propose **FL-EC** communication architecture for privacy-preserving data transmission in IoVs and **FSFormer** for effective intrusion detection. The experimental results show that the proposed model performs best in both centralized learning and FL-EC architecture. Furthermore, our model improves the detection rate of low-probability attacks, alleviating one of the most difficult problems of IDS.

However, there are also some challenges that need to be overcome. Though our model improves detection rates of low-probabilities attacks and has the best overall performance, it suffers from high computational costs. Meanwhile, we can't explain straightforwardly how our model works, which may cause trouble for improvement. As a result, we will try to visualize the training process for a better explanation, then modify the structure of the model to reduce training time and further enhance the performance.

References

[1] Oica. "World vehicles-in-use." Accessed: Dec. 2020. [Online]. Available: https://www.oica.net/category/vehicles-in-use/

[2] S. Sakshi, "Vehicular ad-hoc network: An overview," in *Proc. Int. Conf. Comput. Commun. Intell. Syst. (ICCCIS)*, 2019, pp. 131–134.

[3] P. Kabiri and A. A. Ghorbani, "Research on intrusion detection and response: A survey," *Int. J. Netw. Secur.*, vol. 1, no. 2, pp. 84–102, 2005.

[4] W. Ma, "Analysis of anomaly detection method for Internet of Things based on deep learning," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 12, 2020, Art. no. e3893.

[5] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 810–820, Jul. 2002.

[6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[7] J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, and M. Tarvainen, "Processing intrusion detection alert aggregates with time series modeling," *Inf. Fusion*, vol. 10, no. 4, pp. 312–324, 2009.

[8] Q. Wu and Z. Shao, "Network anomaly detection using time series analysis," in *Proc. Joint Int. Conf. Autonom. Auton. Syst. Int. Conf. Netw. Services (ICAS-ISNS)*, 2005, pp. 42.

[9] N. Walkinshaw, R. Taylor, and J. Derrick, "Inferring extended finite state machine models from software executions," *Empiric. Softw. Eng.*, vol. 21, no. 3, pp. 811–853, 2016.

[10] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, "A language-based intrusion detection approach for automotive embedded networks," *Int. J. Embedded Syst.*, vol. 10, no. 1, pp. 1–11, 2018.

[11] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Exp. Syst. Appl.*, vol. 41, no. 4, pp. 1690–1700, 2014.

[12] P. S. Kenkre, A. Pai, and L. Colaco, "Real time intrusion detection and prevention system," in *Proc. 3rd Int. Conf. Front. Intell. Comput. Theory Appl. (FICTA)*, 2015, pp. 405–411.

[13] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naïve bayes multiclass classifier," *Exp. Syst. Appl.*, vol. 39, no. 18, pp. 13492–13500, 2012.

[14] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Exp. Syst. Appl.*, vol. 39, no. 1, pp. 424–430, 2012.

[15] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden Markov models for malware classification," *J. Comput. Virol. Hack. Techn.*, vol. 11, no. 2, pp. 59–73, 2015.

[16] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl. Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.

[17] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in Internet of Vehicles," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[18] Y.-Y. Song and L. Ying, "Decision tree methods: Applications for classification and prediction," *Shanghai Arch. Psychiat.*, vol. 27, no. 2, pp. 130–135, 2015.

[19] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, Aug. 2016.

[20] S. S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using XGBoost," *Information*, vol. 9, no. 7, p. 149, 2018.

[21] C. Shen, C. Liu, H. Tan, Z. Wang, D. Xu, and X. Su, "Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 26–31, Dec. 2018.

[22] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja, *Network intrusion Detection Using Deep Learning: A Feature Learning Approach.* Singapore: Springer, 2018.

[23] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[24] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[25] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018.

[26] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.

[27] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[28] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Oct. 2022.

[29] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.

[30] Y.-G. Yang, H.-M. Fu, S. Gao, Y.-H. Zhou, and W.-M. Shi, "Intrusion detection: A model based on the improved vision transformer," *Trans. Emerg. Telecommun. Technol.*, vol. 33, Sep. 2022, Art. no. e4522.

[31] M. Abdel-Basset, N. Moustafa, H. Hawash, I. Razzak, K. M. Sallam, and O. M. Elkomy, "Federated intrusion detection in blockchain-based smart transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2523–2537, Mar. 2022.

[32] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[33] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 11106–11115.

[34] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, and N. Kumar, "P2SF-IoV: A privacy-preservation-based secured framework for Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 22571–22582, Nov. 2022.

[35] M. Sarhan, W. W. Lo, S. Layeghy, and M. Portmann, "HBFL: A hierarchical blockchain-based federated learning framework for a collaborative IoT intrusion detection," 2022, *arXiv:2204.04254*.

[36] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "BDTwin: An integrated framework for enhancing security and privacy in cybertwin-driven automotive industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17110–17119, Sep. 2022.

[37] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, Jun. 2018, Art. no. 7476201.

[38] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[40] Y. Yin et al., "IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 Dataset," 2022, *arXiv:2203.16365*.

[41] S. Mehta, M. Ghazvininejad, S. Iyer, L. Zettlemoyer, and H. Hajishirzi, "DeLighT: Deep and light-weight transformer," 2020, *arXiv:2008.00623*.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[43] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The UCI KDD archive of large data sets for data mining research and experimentation," *ACM SIGKDD Explor. Newslett.*, vol. 2, no. 2, pp. 81–85, 2000.

[44] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.

[45] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[46] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.

[47] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," in *Proc. 43rd Annu. Southeast Region. Conf.-Vol. 2*, 2005, pp. 136–141.

[48] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *Int. J. Adv. Netw. Appl.*, vol. 7, no. 4, pp. 2828–2834, 2016.

[49] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[50] M. Hasan, A. Mehedi, M. Nasser, B. Pal, and S. Ahmad, "Support vector machine and random forest modeling for intrusion detection system (IDS)," *J. Intell. Learn. Syst. Appl.*, vol. 6, no. 1, pp. 45–52, 2014.

[51] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao, "SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism," *Comput. Security*, vol. 97, Oct. 2020, Art. no. 101984.

[52] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM," *Comput. Security*, vol. 106, Jul. 2021, Art. no. 102289.

[53] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," in *Proc. Trans. Assoc. Comput. Linguist.*, vol. 9, 2021, pp. 53–68.

[54] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," *Exp. Syst. Appl.*, vol. 185, Dec. 2021, Art. no. 115524.