GENERAL INSTRUCTION

- **Authors: Please check and confirm whether the name of the corresponding author is correct as set.**
- **Authors: When you submit your corrections, please either annotate the IEEE Proof PDF or send a list of corrections. Do not send new source files as we do not reconvert them at this production stage.**
- **Authors: Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections. Your article has been peer reviewed, accepted as final, and sent in to IEEE. No text changes have been made to the main part of the article as dictated by the editorial level of service for your publication.**
- **Authors: Per IEEE policy, one complimentary proof will be sent to only the Corresponding Author. The Corresponding Author is responsible for uploading one set of corrected proofs to the IEEE Author Gateway.**

QUERIES

Q1. Please provide ORCID for the authors Gang Xiong, Xinjie Zhao, Shize Guo in the byline.
Q2. Please confirm or add details for any funding or financial support for the research of this article.
Q3. Please provide complete bibliographic details for Refs. [4], [5], and [16].
Q4. Please provide issue and month for Ref. [20].

I²RNN: An Incremental and Interpretable Recurrent Neural Network for Encrypted Traffic Classification

Zhuoxue Song, Ziming Zhao, Fan Zhang, *Member, IEEE*, Gang Xiong, Guang Cheng, *Member, IEEE*, Xinjie Zhao, and Shize Guo

***Abstract*—Traffic classification occupies a significant role in cybersecurity and network management. The widespread of encryption transmission protocols such as SSL/TLS has led to the dominance of deep learning based approaches. In cybersecurity, strong adversaries often complicate their strategies by constantly developing emerging attacks. Meanwhile, security practitioners desire to grasp the reasons for inference results. However, existing deep learning approaches lack efficient adaptation for incremental traffic types and often have less interpretability. In this paper, we propose I²RNN, an Incremental and Interpretable Recurrent Neural Network for encrypted traffic classification. The I²RNN proposes a novel propagation process to extract the sequence fingerprints from sessions with local robustness. Meanwhile, this proposal provides interpretability including time-series feature attribution and inter-class similarity portrait. Moreover, we design I²RNN in an incremental manner to adapt to emerging traffic types. The I²RNN only needs to train an additional set of parameters for the newly added traffic type rather than retraining the whole model with the entire dataset. Extensive experimental results show that our I²RNN can achieve remarkable performance in traffic classification, incremental learning, and model interpretability. Compared with other local interpretability methods, our I²RNN exhibits excellent stability, robustness, and effectiveness in the interpretation of network traffic data.**

Manuscript received 17 February 2022; revised 10 November 2022; accepted 11 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grants 62227805 and 62072398, in part by SUTD-ZJU IDEA for visiting professors under Grant SUTD-ZJUVP201901, in part by the National Key R&D Program of China under Grant 2020AAA0107700, in part by the Alibaba-Zhejiang University Joint Institute of Frontier Technologies, in part by Zhejiang Key R&D Plan under Grant 2021C01116, in part by the Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang under Grant 2018R01005, in part by the Research Institute of Cyberspace Governance in Zhejiang University, in part by the National Key Laboratory of Science and Technology on Information System Security under Grant 6142111210301, in part by the State Key Laboratory of Mathematical Engineering and Advanced Computing, and in part by the Key Laboratory of Cyberspace Situation Awareness of Henan Province under Grant HNTS2022001. *(Corresponding author: Fan Zhang.)*

Zhuoxue Song, Ziming Zhao, Xinjie Zhao, and Shize Guo are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: songzhuoxue@zju.edu.cn; zhaoziming@zju.edu.cn; zhaoxinjieem@163.com; nsfgsz@126.com).

Fan Zhang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China, and with the ZJU-Hangzhou Global Scientific and Technological Innovation Center, Hangzhou 311200, China, and with the Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province, Jiaxing Research Institute, Zhejiang University, Jiaxing 314000, China, and also with the Zhengzhou Xinda Institute of Advanced Technology, Zhengzhou 450001, China (e-mail: fanzhang@zju.edu.cn).

Gang Xiong is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xionggang@iie.ac.cn).

Guang Cheng is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: chengguang@seu.edu.cn).

Digital Object Identifier 10.1109/TDSC.2023.3245411

***Index Terms*—Encrypted traffic classification, incremental learning, interpretability, recurrent neural network.**

I. Introduction

TRAFFIC classification is important to the entire network for many different purposes, such as network management, Quality of Service (QoS) guarantees, and cybersecurity [1], [2], [3]. Over the last decades, the volume of traffic starts to be encrypted by application-layer encryption transmission protocols, such as Secure Socket Layer/Transport Layer Security (SSL/TLS) [4], [5]. Such encryption technology protects the privacy of Internet users, yet it provides attackers chances to evade firewall detection and circumvent surveillance systems. For example, an attacker may exploit encryption technology to invade and attack the system anonymously. That is to say, encryption technology brings new challenges to traffic identification. Therefore, the classification of encrypted traffic has attracted great attention in both academia and industry [6].

Previous traffic classification methods can be roughly divided into four main categories: port-based [7], payload-based [8], [9], machine-learning-based (ML-based) [10], [11], and deep-learning-based (DL-based) [12], [13]. The wide adoption of traffic encryption techniques, such as the SSL/TLS protocols, causes traditional port-based and payload-based methods to nearly fail. The payload values in packets can be considered as totally *randomized* after cryptographic encryption [14], which are extremely difficult (nearly impossible to some extent) for those port-based and payload-based methods to handle. Therefore, many researchers turn to ML-based and DL-based methods, which have become the mainstream methods for encrypted traffic classification nowadays.

The workflow of traffic classification with machine learning mainly contains two phases, namely feature engineering and model training [15]. The former is to design and select statistical features from traffic flows, such as the average packet length, the average interval of packet arrival time, the maximum TCP window size, etc. The latter is to feed the features into a specific classification model, e.g., SVM [16]. Both phases will directly affect the eventual performance and effectiveness of the classification. Meanwhile, the ML-based method is especially dependent on the so-called feature selection process which requires the sophisticated experience of those experts in the area. Therefore, many *end-to-end* or nearly *end-to-end* methods based on deep learning were proposed as in demand [17], [18]. These

methods directly use the raw traffic as input and fully automate the feature extraction process.

Nonetheless, there are still some problems with existing state-of-the-art (SOTA) DL-based methods.

(i) *They are not suitable in incremental learning scenarios:* Incremental learning refers to sequentially learning from data for new categories, which are available over time, without accessing past data while preserving the learned knowledge for old categories [19]. A fatal limitation of current DL-based traffic classification methods lies in assuming the training data for all categories are always available, making them unsuitable in some real-world situations, where new traffic types are always emerging. In order to append a new capability of identifying an additional traffic type, a common way in the previous works is to retrain the whole model with the entire dataset, which will consume a lot of computing time and memory resources. (ii) *They lack interpretability in modeling traffic behavior:* An interpretable model allows us to understand how it comes to specific conclusions [20]. These DL-based traffic classification methods are designed as end-to-end, which naturally lack interpretability. The calculation process is put into a black-box setting, whose details are hard to infer. Specifically, it is much more difficult to grasp the internal cause of inference results.

In this paper, we propose I^{2}RNN, an **I**ncremental and **I**nterpretable **R**ecurrent **N**eural **N**etwork model for encrypted traffic classification. The I^{2}RNN leverages fingerprint learning from the raw session sequences rather than manually designed features. A list of fingerprint modules is trained in fingerprint learning. Each fingerprint module is a long short-term memory unit with an encoding layer. The fingerprint modules characterize the patterns of sequential features and thus learn the fingerprints of different traffic types. Particularly, (i) I^{2}RNN maintains a set of specific yet *independent* parameters for each traffic type. Therefore, it only needs to train an additional set of parameters for the newly added traffic type. (ii) I^{2}RNN is inherently interpretable with respect to time-series feature attribution and inter-class similarity portrait. It determines the traffic type by comparing the output losses of different fingerprint modules. Therefore, by decomposing the process of losses comparison, we can provide explanations for the model classification results.

Our main contributions can be briefly summarized as below:

- We propose I^{2}RNN, an incremental and interpretable recurrent neural network model for encrypted traffic classification. It learns fingerprints from the raw session sequences and holds local robustness.
- I^{2}RNN can be updated in an incremental manner. It only needs to train an additional set of parameters for the newly added traffic type.
- I^{2}RNN is an interpretable model. It can rank features for each traffic type, identify important features for classification, and depict the inter-class distance between traffic types in specific dimensions.
- I^{2}RNN achieves excellent results on the real-world network traffic dataset and outperforms several state-of-the-art methods.

The rest of this paper is organized as follows: Sections II and III list some related works and preliminaries, respectively. The detailed design of I^{2}RNN is introduced in Section IV. Section V presents model interpretability. In Section VI, we exhibit validation results. After a brief discussion in Section VII, we conclude the paper in Section VIII.

II. Related Work

A. Traffic Classification

1) Unencrypted Traffic Classification: The basic idea of unencrypted traffic classification is to exploit useful information from packet headers and payloads. The port-based methods classify traffic by checking the TCP/UDP port number at the transport layer [7]. Unfortunately, port-hopping techniques [21] make the port-based methods ineffective. The payload-based methods [8], [22] inspect the contents in packet payloads with a predefined rule set, and then use rule-matching results as traffic decisions. The design of the rule set affects the accuracy of the payload-based methods. In addition, such approaches fail in encrypted traffic classification as the payloads are encrypted.

2) Encrypted Traffic Classification: In 2014, Korczynski et al. proposed an encrypted traffic classification method based on the first-order Markov chain [23]. They used message-type sequences of encrypted traffic to build a first-order Markov model for classification. Subsequently, Shen et al. proposed a similar but improved method based on the second-order Markov chain [24] with new features including certificate length and the first packet length. Chang et al. combined both message type sequences and length sequences to build Markov models and got better performance [25]. However, all those Markov-based methods are with small orders (e.g., 1 or 2) and thus they cannot deal with the long-term relationship.

In addition to the Markov chain, many other machine learning algorithms were used in encrypted traffic classification. Liu et al. proposed a semi-supervised method for encrypted traffic classification with carefully selected features which include the maximum, minimum, and average of sent and received bytes [26]. Anderson et al. took flow metadata, packet length distribution, and time distribution into consideration and analyzed encrypted malware traffic with the logistic regression algorithm [27]. However, these ML-based methods are especially dependent on feature selection which relies on sophisticated experience.

In recent years, the research on encrypted traffic classification is evolving towards the direction of deep learning. In 2019, Chang et al. applied the recurrent neural network to the encrypted traffic classification problem and propose the Flow Sequence Network (FS-Net) [12]. The FS-Net is an end-to-end classification model that learns representative features from the raw packet size sequences of encrypted traffic. Unlike FS-Net, TSCRNN [28] learned from raw payloads. In 2020, Wenbo et al. proposed the Flow-Based Relation Network (RBRN), which uses the meta-learning approach to address the problems of encrypted traffic classification including imbalanced network data, model generalization, and overly dependent on data size [17]. In 2021, Shen et al. utilized different graph structures (e.g., spindle-shaped, fish-shaped) as input features for traffic classification [29]. In 2022, Lin et al. proposed a new traffic representation model called ET-BERT, which utilizes pre-training

transformers and the contextualized datagram representation to classify the encrypted traffic [30]. However, these DL-based methods usually lack interpretability and cannot conduct the learning incrementally.

Our work incorporates the idea of learning traffic fingerprints. However, unlike the previous work modeling the fingerprints of traffic with Markov chains, we adopt LSTM to learn the fingerprints of traffic, which can deal with the long-term relationship. Our work is an end-to-end traffic classification network that learns fingerprints from the raw session sequences. Compared with the previous deep-learning-based work in traffic classification [31] and anomaly detection [32], we focus on incremental learning and model interpretability.

B. Incremental Learning

A variety of strategies have been explored to train the model incrementally [33], [34], [35]. Li et al. proposed Learning without Forgetting (LwF), which uses the new data to supervise the learning of the new tasks and to provide unsupervised output guidance on the previous tasks [33]. Further, Dhar et al. introduced Learning without Memorizing, which extends LwF by adding a distillation term based on attention maps [34]. Yanan et al. proposed an incremental method for the instance segmentation task, which uses multi-teacher networks and achieves excellent performance on instance segmentation datasets [35]. However, previous work on incremental learning mainly focuses on computer vision tasks. The data format of network traffic is significantly different from that of images. Therefore, existing incremental learning techniques cannot be applied directly to network traffic classification. It is both challenging and meaningful to explore incremental learning models for network traffic classification.

C. Model Interpretability

The model interpretability can be classified as global interpretability and local interpretability. The global level of interpretability is to give a holistic view of the input features. The output of the global interpretability is usually feature ranking and selection results. Previous feature selection algorithms can be roughly divided into three ways: filter-based, wrapper-based and embedded. Filter-based feature selection algorithms remove features suspected to be irrelevant based on metrics such as mutual information [36]. Wrapper-based feature selection algorithms iteratively select features through retraining the model across different feature subsets [37]. Embedded feature selection algorithms select important features as the model is trained [38]. The local level of interpretability is about understanding the importance of each feature in a single prediction of a model. There are some well-known local interpretability methods like LIME [39], SHAP [40], feature permutation [41], etc. However, previous model interpretability algorithms are mainly applicable to tabular data. The network traffic data, however, is time series. Existing model interpretability methods do not consider the temporal characteristics of data. Thus they may be incapable of interpreting the network traffic classification task.

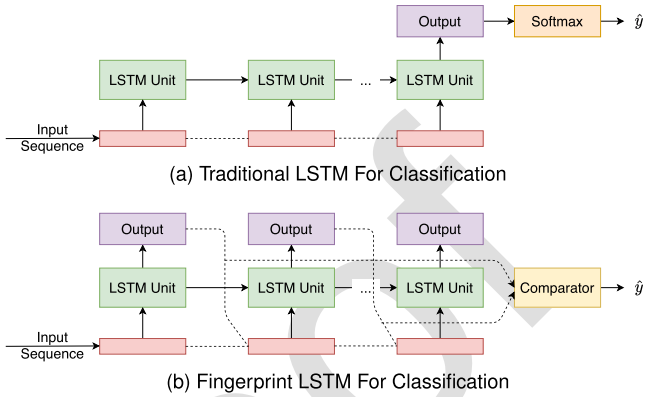


Fig. 1. Comparison between traditional LSTM and fingerprint LSTM.

III. Preliminaries

A. Problem Definition

The traffic classification in this paper refers to classifying the encrypted traffic of sessions into specific traffic types with raw features of packets. Assume that there are N samples and M types of traffic in total. Let the sequence of the p-th sample be $\mathbf{x^{(p)}} = [\mathbf{L_1^{(p)}}, \mathbf{L_2^{(p)}}, \ldots, \mathbf{L_T^{(p)}}]$, where T is the time sequence length of each sample and $\mathbf{L_i^{(p)}} \in \mathbb{R}^d$ is the packet feature at time step i. The traffic type of $\mathbf{x^{(p)}}$ is denoted as $y^{(p)}$, $0 \leq y^{(p)} < M$. We consider an incremental learning scenario and aim to build an end-to-end incremental model $\psi(\mathbf{x^{(p)}})$ to predict a label $\hat{y_p}$ that is exactly the real label y_p. Suppose that there are total S sequential stages in the learning process. We define stage s as introducing the new class C^s using the dataset $D^s = \{(\mathbf{x}, y) | y \in C^s\}$. The training samples and labels are defined as $X^s = \{\mathbf{x} | (\mathbf{x}, y) \in D^s\}$ and $y^s = \{y | (\mathbf{x}, y) \in D^s\}$, respectively. Especially, $y^i \cap y^j = \emptyset$ for $i \neq j$. At stage 0, a classification model $\psi^0(\mathbf{x^{(p)}})$ is trained on dataset X^0 with m^0 classes. For stage j a model $\psi^j(\mathbf{x^{(p)}})$ is trained to classify on accumulated $M = \sum_{q=0}^{j} |m^q|$ classes.

B. Long Short-Term Memory (LSTM)

The LSTM [42] is one of the most popular recurrent neural networks (RNNs) [43] to model sequential data. LSTM can infer the current state based on the previous state and the current input. Therefore, LSTM can remember past information, which is naturally suitable for sequence modeling. The main weak point of the vanilla version of RNN is the vanishing/exploding gradient problem which means cannot retain long-term information [44]. By adding gate units with different functions, the LSTM solves the problem. It can control the information transformation between the hidden states and track the states of the input sequences without using separate memory cells.

Traditionally, as shown in Fig. 1(a), when LSTM is used for classification tasks, the input sequence is first fed into the LSTM unit. Second, the output of the last time step of the LSTM unit is given to the fully connected layer. The classification results can be get after the calculation of the softmax layer. However, in this paper, we use our proposed fingerprint LSTM instead of

traditional LSTM. As shown in Fig 1(b), the fingerprint LSTM tries to capture the fingerprints of different types. It calculates and compares the loss between the output of the current state and the input of the next time step. In the classification stage, the prediction results are determined by the fingerprint LSTM with the minimum loss. There are two main benefits of fingerprint LSTM. First, the traditional LSTM only uses the output of the last time step for the calculation of the fully connected layer. Therefore, the traditional LSTM model tends to learn the features of the subsequent input. It can be easily affected by noise in the later-arrived packets. In the contrast, the learning process of the fingerprint LSTM makes the output of each time step (except the last time step) as close as possible to the input of the next time step. This process makes full use of all packets in a session and thus has local robustness. Second, a fingerprint LSTM module only characterizes the packet sequence of a specific traffic type. The modeling process will not be influenced by other traffic types. In the contrast, traditional LSTM may have a bias towards traffic types with a large number of samples.

IV. Proposed Traffic Classification Method

A. Overview

In Sections I and III, we have discussed the limitations of current encrypted traffic classification methods, which motivate us to carry out further studies. Here we briefly summarize the limitations and motivations as the following two points.

(i) *Current methods for encrypted traffic classification are not suitable for incremental learning.* These methods assume the training data for all categories is available at the time of initial training. However, it will always emerge new attacks in real-world situations. This motivates us to design an incremental model.

(ii) *Current methods for encrypted traffic classification are not interpretable.* The inference process of these methods is under a black-box setting and lacks interpretability. However, we need to understand how the model comes to specific conclusions and what the model learns.

Based on the above limitations and motivations, we propose the I^{2}RNN. The two main design principles are as follows:

(i) *I^{2}RNN learns the fingerprints of traffic and is interpretable.* The training process of I^{2}RNN does not directly implement a traditional classification task. Instead, each LSTM unit learns the *fingerprint* (behavioral characteristics) of the input samples. In other words, at each time step, the current output of each LSTM unit is as close as possible to the next input of the sequence. This learning process is different from the typical classification task which calculates cross-entropy loss at the last time step as described in Section III. We term this training process *fingerprint learning*. By such replacement, our I^{2}RNN has local robustness and is adaptive to imbalanced data. Moreover, we can rank time-series features, identify import features in prediction and calculate the inter-class distance between different traffic types based on the learning process. These capabilities enhance the interpretability of I^{2}RNN and will be demonstrated in Section V.

(ii) *I^{2}RNN is an incremental model.* It contains multiple independent fingerprint modules. Each module maintains a set of parameters and corresponds to the *fingerprint* of a specific traffic type. Therefore, we only need to train a set of additional parameters for the newly introduced traffic type. Such a design indicates that I^{2}RNN could be deployed as an incremental model.

The overview of I^{2}RNN is depicted in Fig. 2. The whole process of I^{2}RNN includes incremental learning and classification. Further, the workflow of incremental learning can be divided into feature input and incremental fingerprint learning. We will discuss each part in detail below.

B. Feature Input

The first part is feature input. In this process, we extract feature tensors from raw traffic data. We first split the raw traffic data of each traffic type into sessions and then perform feature extraction on each packet in sessions.

For formalization, suppose we have raw traffic data D and we split it into N sessions. Each session is a sample and is composed of bi-directional packets with the same IP, port, and transport layer protocol. The feature tensors are extracted from the first T packets in each session and are composed of the packet feature vectors. Each packet feature vector consists of packet length, direction, the interval of arrival time, and values of fields in the packet header, e.g., TTL (Time To Live), TCP flags, and TCP window size. Therefore, the feature tensor of the p-th sample can be denoted as $\mathbf{x}^{(\mathbf{p})} = [\mathbf{L}_\mathbf{1}^{(\mathbf{p})}, \mathbf{L}_\mathbf{2}^{(\mathbf{p})}, \ldots, \mathbf{L}_\mathbf{T}^{(\mathbf{p})}]$, where T is the time sequence length of each sample and $\mathbf{L}_\mathbf{i}^{(\mathbf{p})} \in \mathbb{R}^d$ is the d-dimensional packet feature vector at time step i.

Particularly, the flexibility of our feature input is that we can select different features and different time sequence lengths for each traffic type, i.e., d and T can be different for various traffic types. In other words, the feature input process is not dependent on the fixed number of traffic types. When a new traffic type is added, we can select the appropriate features and time sequence length for this newly added traffic type without affecting those of existing traffic types. This property of the feature input process creates the conditions for incremental fingerprint learning.

C. Incremental Fingerprint Learning

The second part is incremental fingerprint learning. In this process, we learn the fingerprints of different traffic types in an incremental manner.

Different from the model training process of typical classification tasks with LSTM, I^{2}RNN aims to train a model which can learn *fingerprints* (behavioral characteristics) of each traffic type. We call this process *fingerprint learning*. As described in previous sections, we use fingerprint LSTM instead of traditional LSTM to model the packet sequence in sessions of a traffic type, which has local robustness and is suitable for an imbalanced dataset.

In addition, we add an encoding layer before fingerprint LSTM to constitute a fingerprint module. The encoding layer takes the original feature tensors as input and generates the

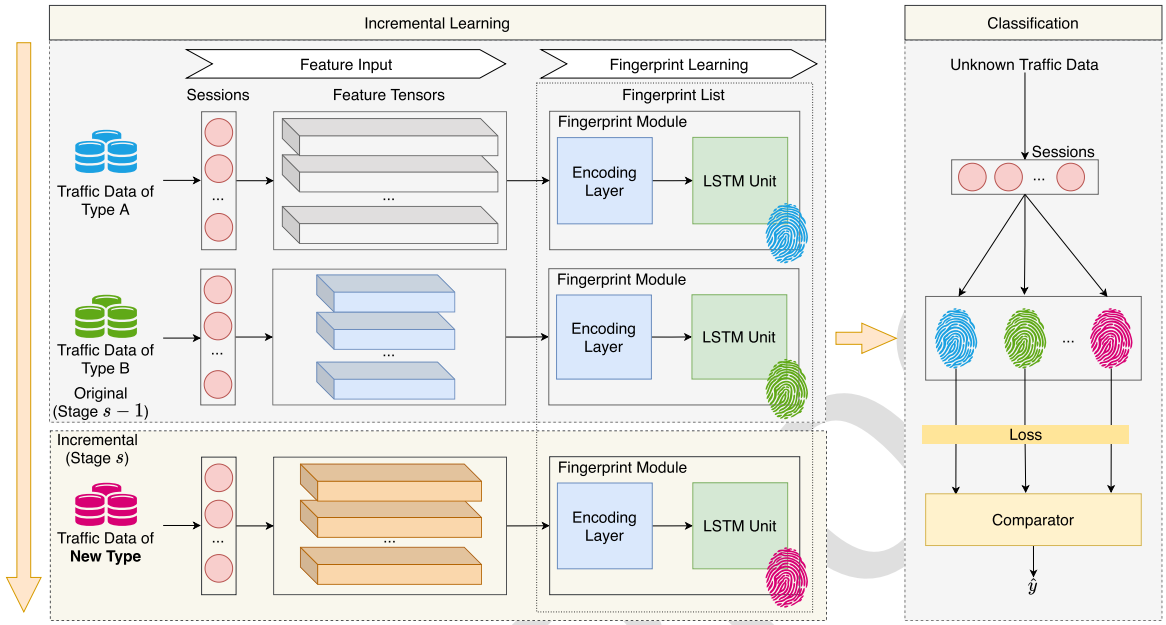


Fig. 2. Overview of I²RNN.

latent representations. It simply uses a linear layer with the $tanh$ activation function. There are two benefits of adding an encoding layer. (i) Some fields in the packet headers are categorical features, e.g., TCP Flags (SYN, ACK, etc). The encoding layer can transform these features into numerical features, which are more convenient for computation. (ii) There may be redundant information in the original features. The encoding layer can reduce the redundant information, which helps the model learn the fingerprints of network traffic better.

Moreover, the I²RNN maintains a fingerprint list. As depicted in Fig. 2, there are multiple fingerprint modules in the fingerprint list. Each fingerprint module maintains an *independent* set of parameters that corresponds to a traffic type. This means that I²RNN is designed in an incremental learning manner. In other words, the only thing that needs to be done for having the ability to recognize a new traffic type is to introduce an additional fingerprint module to the fingerprint list and to train this module with the new traffic data. This provides I²RNN the capability to classify new traffic types, without retraining existing fingerprint modules.

The entire algorithm of incremental fingerprint learning is exhibited in Algorithm 1.

In an incremental learning scenario, we introduce a new traffic type using training samples X^s. After the incremental fingerprint learning, the previous stage model ψ^{s-1} becomes ψ^s, which has the ability to classify the new traffic type. The first step of incremental fingerprint learning is to create a new fingerprint module, which consists of an encoding layer ρ^s and a fingerprint LSTM Ω^s (line 1). Then we randomly initialize the parameters of each component of the fingerprint module (line 2). In each iteration, we sequentially feed the training samples into the encoding layer and the fingerprint LSTM (lines 4-5). The learning process aims to make the output $\mathbf{a_t^{(p)}}$ as close as possible to the next input $\mathbf{L_{t+1}^{(p)}}$ at t-th time step, where $1 \leq t \leq T-1$ (line 6). Therefore, the loss function of the p-th sample can be set as follow:

$$\mathcal{L}(\mathbf{x^{(p)}}) = \frac{1}{T-1}\sum_{t=1}^{T-1} ||\mathbf{a_t^{(p)}} - \mathbf{L_{t+1}^{(p)}}|| \quad (1)$$

After the calculation of loss, we can obtain the gradient of the parameters of the fingerprint module with backpropagation (line 7). Then we update the parameters of the fingerprint module (line 8). We can get the new model ψ^s after iterations over all training samples (line 10).

Algorithm 1. Incremental Fingerprint Learning

Input: X^s, ψ^{s-1}
1: Create a new fingerprint module $[\rho^s, \Omega^s]$
2: Initialize the parameters of $[\rho^s, \Omega^s]$ randomly
3: **for** $\mathbf{x^{(p)}} \in D^s$ **do**
4: $\quad \mathbf{e^{(p)}} \leftarrow \rho^s(\mathbf{x^{(p)}})$
5: $\quad \mathbf{a^{(p)}} \leftarrow \Omega^s(\mathbf{e^{(p)}})$
6: $\quad \mathcal{L}(\mathbf{x^{(p)}}) \leftarrow \frac{1}{T-1}\sum_{t=1}^{T-1} ||\mathbf{a_t^{(p)}} - \mathbf{L_{t+1}^{(p)}}||$
7: $\quad$ Calculate the gradient of parameters of $[\rho^s, \Omega^s]$
8: $\quad$ Update the parameters of $[\rho^s, \Omega^s]$
9: **end for**
10: $\psi^s \leftarrow$ append $[\rho^s, \Omega^s]$ to the fingerprint list of ψ^{s-1}
Output: ψ^s

D. Traffic Classification

The third part is the traffic classification. When the unknown traffic data arrives, we first process the data with feature input, which converts the raw traffic data into feature tensors. Then we feed feature tensors to each fingerprint module in the fingerprint list that is trained in the incremental fingerprint learning

part. Each fingerprint module calculates the loss defined in (1). Since each fingerprint module has learned the *fingerprint* of a specific traffic type, it will output a small loss for traffic with exactly the same type while giving a relatively large loss for the sample of other traffic types. Consequently, the unknown traffic is classified as the corresponding type of the fingerprint module with the minimum loss. This process is represented as a *comparator* in Fig. 2. To formulate this process, suppose that $\mathcal{L}_i(\mathbf{x^{(p)}})$ represents the output loss of p-th unknown traffic sample calculated by i-th fingerprint module in the fingerprint list. We can classify the sample as $\hat{y_p}$ by determining the specific i which makes $\mathcal{L}_i(\mathbf{x^{(p)}})$ minimum

$$\hat{y_p} = \arg\min_i \mathcal{L}_i(\mathbf{x^{(p)}}) \tag{2}$$

V. Model Interpretability

In the previous section, we have described the design of I^{2}RNN. In this section, we elaborate on the model interpretability of I^{2}RNN from the aspects of feature attribution and inter-class distance between traffic types.

A. Sequence Feature Attribution

Feature attribution is an important part of model interpretability, which can be divided into global-level attribution and local-level attribution. The result of the global level attribution is usually the feature ranking of a specific type, while the local level attribution mainly gives the importance of each feature in the prediction of a single sample.

However, existing feature attribution methods are mainly applicable to tabular data and are not designed for network traffic data, which is time series. If we want these existing feature attribution algorithms to handle the network traffic data, we need to convert the time-series data to tabular data. A common way to accomplish the above transformation is to reshape the time sequence data. For example, assume the time sequence data a has a total of T time steps, and each time step is a d-dimensional feature vector. We can use `b = a.reshape(1, T × d)` to reshape the time sequence data a to tabular data b with *NumPy* [45]. However, this way has two shortages. (i) This transformation considers features at each time step individually and ignores the relationship between the time sequence data. (ii) Current feature ranking methods need a long computation time and have a scalability problem to handle the high-dimensional reshaped data [46].

To address the feature attribution of network traffic data, we propose an effective sequence feature attribution method based on I^{2}RNN. The interpretability of I^{2}RNN focuses on the influence of features on fingerprints in terms of time sequence. Therefore, I^{2}RNN provides a new perspective that illustrates the attribution of sequence features.

Our feature attribution method stems from the decomposition of the internal computational process of I^{2}RNN. Fig. 3 illustrates the internal computational process of I^{2}RNN when feeding traffic sequence data. As mentioned in Section IV, each fingerprint module is trained to learn the *fingerprint* (behavioral characteristics) of a specific traffic type and the classification

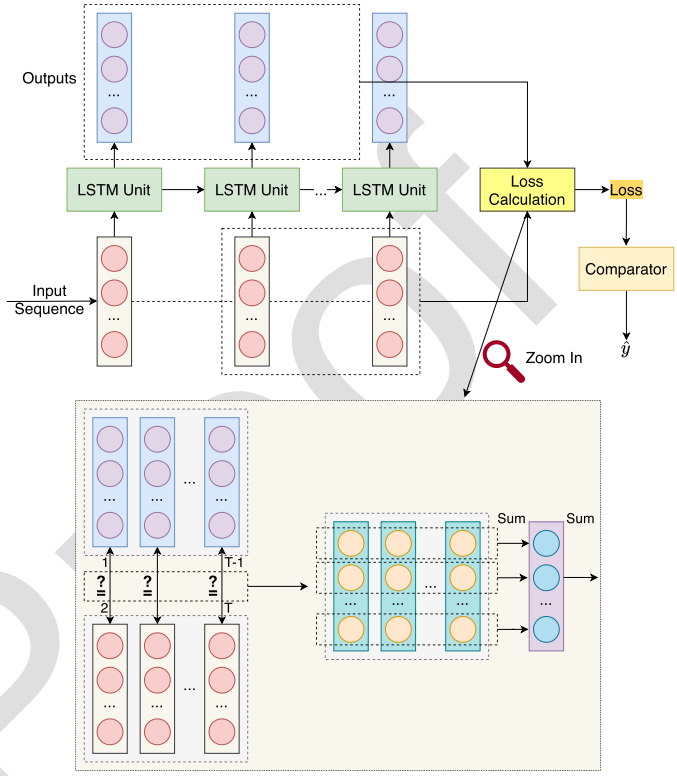


Fig. 3. Internal computational process of I^{2}RNN.

result of the unknown traffic is decided by the fingerprint module with the minimum loss. At each time step, the fingerprint module outputs the prediction of the next time step based on the previous input. The loss is computed according to (1). Actually, (1) is equivalent to first calculating the L2 distance between the output vector and the next input vector at each time step and then summing the distance of each dimension. Specifically, each dimension of $\mathbf{a_t^{(p)}} - \mathbf{L_{t+1}^{(p)}}$ jointly contributes to the overall loss in (1). Therefore, for a specific traffic type, compared the loss at a certain dimension of the corresponding fingerprint module with losses of other fingerprint modules at the same dimension, if the loss at this dimension is smaller and thus has a larger loss difference, we can say this dimension contributes more to the final prediction and thus ranks higher. This is the basic idea of our feature attribution method. Practically, we feed data of a specific type to all the fingerprint modules in the fingerprint list of I^{2}RNN. We calculate the sum of loss difference in each dimension among fingerprint modules and use it to represent the feature importance.

For formalization, we first define the single-dimensional loss of a sample as

$$\mathcal{SL}\left(\mathbf{x^{(p)}}, \psi_J, K\right) = \frac{1}{T-1}\sum_{t=1}^{T-1} |\mathbf{a_{J,t}^{(p)}}[K] - \mathbf{L_{t+1}^{(p)}}[K]| \tag{3}$$

where $\mathcal{SL}(\mathbf{x^{(p)}}, \psi_J, K)$ represents the K-th dimensional loss when feeding sample $\mathbf{x^{(p)}}$ into the J-th fingerprint module in the fingerprint list. Subsequently, we can calculate the loss difference between fingerprint modules in the K-th dimension

of a sample $\mathbf{x}^{(\mathbf{p})}$ (abbreviated as feature loss) as follows:

$$\mathcal{FL}(\mathbf{x}^{(\mathbf{p})}, K) = \frac{1}{M}\sum_{J=0}^{M-1} ReLU\left(\mathcal{SL}(\mathbf{x}^{(\mathbf{p})}, \psi_J, K) - \mathcal{SL}\left(\mathbf{x}^{(\mathbf{p})}, \psi_I, K\right)\right) \tag{4}$$

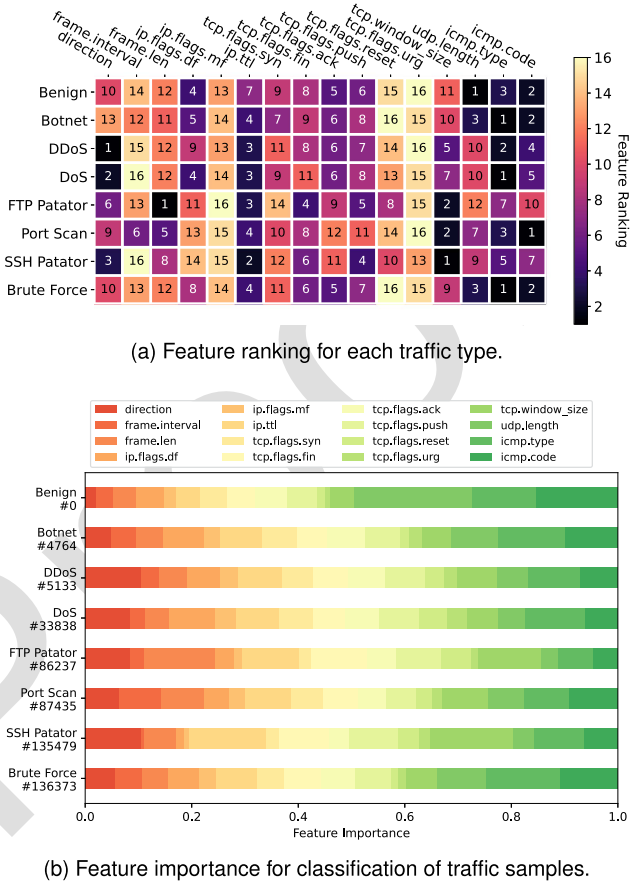
The I-th fingerprint module only learns the *fingerprint* of traffic type I, which is the corresponding type of sample $\mathbf{x}^{(\mathbf{p})}$. Consequently, if $I \neq J$, the overall loss $\mathcal{L}_I(\mathbf{x}^{(\mathbf{p})})$ is supposed to be less than $\mathcal{L}_J(\mathbf{x}^{(\mathbf{p})})$, and K-th dimension loss $\mathcal{SL}(\mathbf{x}^{(\mathbf{p})}, \psi_I, K)$ should be less than $\mathcal{SL}(\mathbf{x}^{(\mathbf{p})}, \psi_J, K)$ correspondingly. Therefore, $\mathcal{FL}(\mathbf{x}^{(\mathbf{p})}, K)$ is usually a positive value. We use ReLU [47] to get the non-negative part. A larger value indicates that the feature at this dimension contributes more to the difference in the overall loss that the traffic classification relies on. Thus we consider the value $\mathcal{FL}(X^I, K)$ as the significance of the feature at the K-th dimension of a sample $\mathbf{x}^{(\mathbf{p})}$.

Eqs. (3) and (4) define the feature importance of a single sample, which offer local interpretability. For the global interpretability of I²RNN, we can sum the feature importance of all the samples belonging to traffic type I

$$\mathcal{SL}(X^I, \psi_J, K) = \sum_{\hat{y^{(p)}}=I} \mathcal{SL}(\mathbf{x}^{(\mathbf{p})}, \psi_J, K) \tag{5}$$

$$\mathcal{FL}(X^I, K) = \sum_{\hat{y^{(p)}}=I} \mathcal{FL}(\mathbf{x}^{(\mathbf{p})}, K) \tag{6}$$

Fig. 4 shows an example of the feature attribution results. Fig. 4(a) depicts the feature ranking results for each traffic type. Each row represents a specific traffic type. Each column stands for a specific feature. The numbers inside the square represent the feature ranking (smaller is higher). These results are interpretable from the perspective of attack principles. For example, from Fig. 4(a), we can see that the *direction* is the significant feature for DDoS and DoS. This is because the communication pattern of these two traffic types is mainly the attacker (the client) sending a flood of packets to the server. Fig. 4(b) shows the feature importance of samples, illustrating which features contribute more to the classification results. The numbers on the y-axis are indexes of the selected sample on the dataset. It facilitates us to understand how the model comes to the final prediction results. Take the FTP Patator #86237 as an example, we can see that the *frame.len* plays an important role in the classification. This is mainly because the purpose of the FTP Patator is to keep trying to decipher the password of the FTP server. Meanwhile, the packet length of the authentication process in FTP is always a fixed value.



(a) Feature ranking for each traffic type.

(b) Feature importance for classification of traffic samples.

Fig. 4. An example of feature attribution results.

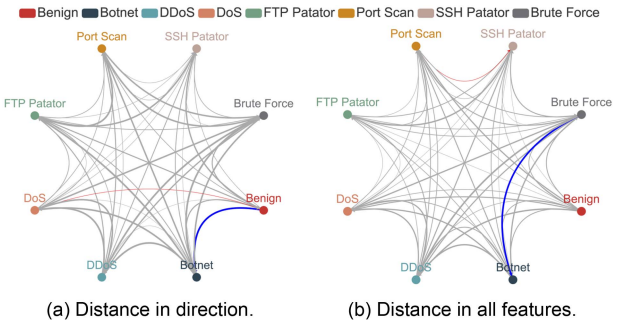
B. Inter-Class Distance Between Traffic Types

According to our design of fingerprint learning, the fingerprint module will output a larger loss if we feed traffic data whose type is not the corresponding type of the fingerprint module. In other words, the loss is the distance between the input traffic and the expected traffic of the fingerprint module. In the previous section, we have defined the single-dimensional loss $\mathcal{SL}(X^I, \psi_J, K)$ in (5), which is the K-th dimension of the output loss when feeding traffic data of type I into the J-th fingerprint module. Based on this, we define the inter-class distance from traffic type I to traffic type J at dimension K as

$$\mathcal{DI}(I, J, K) = ReLU\left(\mathcal{SL}(X^I, \psi_J, K) - \mathcal{SL}(X^I, \psi_I, K)\right) \tag{7}$$

The inter-class distance actually reflects the similarity between traffic types. Considering that the distance is always the non-negative value, we use ReLU to extract the non-negative part. If we take each traffic type as a point and express the distance between the traffic types as an edge between the points, we can draw the distance map on a two-dimensional plane. Note that in general $\mathcal{DI}(I, J, K) \neq \mathcal{DI}(J, I, K)$ and thus the distance map should be a directed graph.

Fig. 5 shows an example of the inter-class distance between traffic types. We use a larger linewidth to represent a smaller inter-class distance. In each subfigure, the red and blue lines indicate the largest and smallest inter-class distances, respectively. Due to the page limitation, we only present the inter-class distance in the dimension of direction and in all features. We find the results are rational. For example, we can see that Port Scan and Brute Force are close in direction. This is mainly because the behavior of these two traffic types is similar, i.e., attempting to access information about the server by sending massive packets



(a) Distance in direction. (b) Distance in all features.

Fig. 5. An example of the inter-class distance between traffic types.

to the target in a short time. DoS and DDoS are close to each other in all features due to their similar attack behaviors. Particularly, Benign and Botnet are the closest in direction because they are both bidirectional communication between the client and servers. In contrast, DoS and Benign are the farthest in direction as the DoS traffic is usually from clients to the target server while Benign contains bidirectional flows. Moreover, we can observe that Botnet and Brute Force are the closest while Port Scan and SSH Patator are the farthest in all features. Inspired by this result, we try to group the traffic types with close distances to a new individual type, which may reduce the struggle of the model to distinguish between similar types and improve the classification performance of the model. The details of this experiment can be found in Section VI-D3.

VI. Evaluation

In this section, we present the experimental settings and the evaluation results of the proposed I^{2}RNN.

A. Experimental Settings

1) Datasets: We evaluate the proposed I^{2}RNN on two public datasets CIC-IDS2017 dataset [48] and ISCXVPN2016 traffic dataset [49], respectively. The first dataset contains the hybrid of encrypted and unencrypted network traffic across five days, which can be divided into 8 classes, Benign, FTP Patator, SSH Patator, DoS, Web Brute Force, Botnet, Port Scan, and DDoS respectively. The second dataset contains pure encrypted traffic, which is composed of 14 applications, e.g., Facebook, Netflix, Skype, etc. The applications are encrypted with various security protocols, including HTTPS, SSH, and proprietary protocols. Both two datasets are imbalanced. For example, in the CIC-IDS2017 dataset, Port Scan accounts for 35.07%, while Botnet only accounts for 0.27%. In the ISCXVPN2016 traffic dataset, Email accounts for 16.89%, while SFTP only accounts for 0.65%.

2) Baselines: We compare I^{2}RNN against various baseline works for encrypted traffic classification, including KNN [50], RF [51], MLP [52], LSTM [6], and FS-Net [12].

We compare our global feature ranking method against the mutual information feature ranking [36], RFE feature ranking [37], and random forest feature ranking [38]. Since these methods cannot be performed on time-series features directly, we first reshape the time-series data into tabular data as mentioned in Section V-A. Then we calculate the feature importance on the reshaped data with these methods respectively. Finally, we *restore* the results of feature importance to the corresponding time-series features to obtain the feature ranking of the original features. The process *restore* is to accumulate the feature importance of corresponding features on the reshaped data, for each feature of the original time-series data.

In terms of local interpretability, we compare the performance of our proposed method against well-known methods, including LIME [39], SHAP [40], and Feature Permutation [41].

3) Implementation Details: The time sequence length of each LSTM unit is set as 32. We set the dimension of the encoding layer as 8. The dimension of hidden states of each fingerprint LSTM is set as 16. The Adam optimizer [53] with a learning rate of 0.001 is used. Our proposed method I^{2}RNN is implemented with *PyTorch*. We use *scikit-learn* [54] to implement other compared methods. The compared local interpretability methods are the implementation from *Captum* [55]. All experiments in this paper are conducted using a 12-core PC with an NVIDIA GeForce GTX 1070 GPU, 128 GB of RAM, and Ubuntu 16.04 LTS.

4) Evaluation Metrics: We evaluate the classification performance of I^{2}RNN with other baselines in terms of Accuracy (AC), Precision (PR), Recall (RC), and F1-Score (F1) [17]. Macro Average [56] is used to avoid biased results due to imbalance between multiple categories of data by calculating the mean value of AC, PR, RC, and F1 of each category.

The performance of local interpretability will be evaluated in terms of stability, robustness, and effectiveness [20].

B. Evaluation of Classification

In this section, we evaluate I^{2}RNN and baseline classification methods with the 5-fold cross-validation.

1) Effectiveness of the Fingerprint Learning: In order to show the effectiveness of fingerprint learning, we plot the distribution of losses output by each fingerprint module when feeding traffic data of different types. Each subfigure in Fig. 6 shows the kernel density estimation (KDE) [57] of the distribution of losses (due to page limitation we do not show all results). The losses of samples whose types are exactly the corresponding type of the fingerprint module (red rectangle part) are significantly lower than that of other types (x-axis). It demonstrates that each fingerprint module in the fingerprint list of I^{2}RNN can learn the *fingerprint* of the corresponding traffic type well. In other words, the core idea of I^{2}RNN described in Section IV is effective.

Tables I and II show the classification results. It is clear that I^{2}RNN achieves significant classification performance and outperforms all other methods in both traffic datasets. The KNN, RF, and MLP are designed to learn from the tabular data. They cannot model the context information in the sequence data and thus perform worse than I^{2}RNN. The I^{2}RNN has a better classification performance than LSTM and FS-Net, especially in F1-Score. It is mainly because both datasets are imbalanced. The I^{2}RNN leverages the advantage of fingerprint learning, i.e., each fingerprint module is only responsible for identifying one traffic

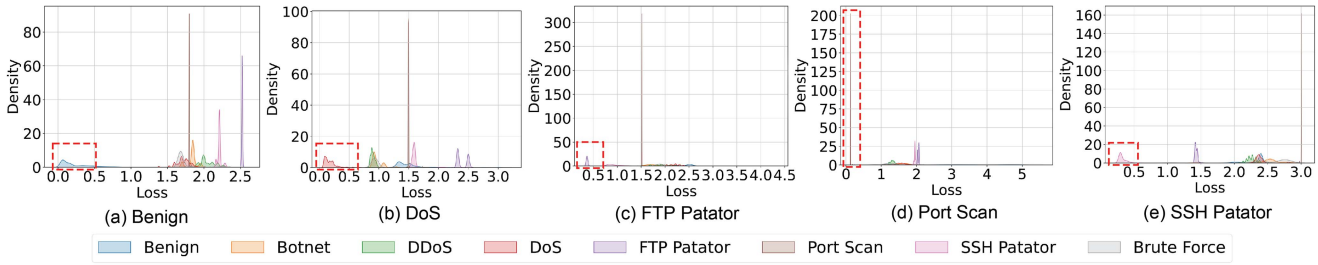


Fig. 6. KDE of losses output by fingerprint modules when feeding different types of traffic data on CIC-IDS2017.

TABLE I
COMPARISON RESULTS OF DIFFERENT CLASSIFICATION METHODS ON CIC-IDS2017 DATASET

Method	AC	PR	RC	F1
I2RNN	**0.9897**	**0.9864**	**0.9859**	**0.9861**
I2RNN w/o EL	0.9769	0.9752	0.9729	0.9766
KNN [50]	0.9579	0.8886	0.8891	0.8886
RF [51]	0.9698	0.9692	0.9621	0.9629
MLP [52]	0.9699	0.9699	0.9693	0.9630
LSTM [6]	0.9649	0.7961	0.8560	0.8124
FS-Net [12]	0.9526	0.4721	0.4439	0.4218

TABLE II
COMPARISON RESULTS OF DIFFERENT CLASSIFICATION METHODS ON ISCXVPN2016 DATASET

Method	AC	PR	RC	F1
I2RNN	**0.9393**	**0.9068**	**0.9178**	**0.9013**
I2RNN w/o EL	0.8922	0.8683	0.8693	0.8490
KNN [50]	0.8799	0.7692	0.7094	0.6924
RF [51]	0.8712	0.8742	0.8647	0.8634
MLP [52]	0.8503	0.6992	0.6745	0.5540
LSTM [6]	0.8813	0.5213	0.5132	0.5006
FS-Net [12]	0.5230	0.0924	0.1122	0.0859

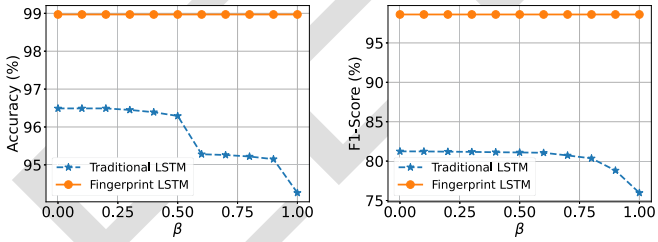


Fig. 7. Comparison results of traditional LSTM and fingerprint LSTM on CIC-IDS2017 with different perturbation ratios β.

type. Thus it will not be confused by different traffic fingerprints and is suitable for imbalanced data.

2) Local Robustness of the Fingerprint Learning: We compare the local robustness of the fingerprint learning of I²RNN and traditional classification with LSTM when having noise in packet sequence. We add a Gaussian noise $\epsilon \tilde{} N(0,1)$ with different perturbation ratios $\beta \in [0,1]$ to the last two packets in the packet sequence of a session. The experiment results are shown in Fig. 7. When gradually increasing the perturbation ratio from 0.0 to 1.0, we find the accuracy and F1-Score of the fingerprint LSTM change slightly while the performance of the traditional LSTM drops sharply. This result indicates our

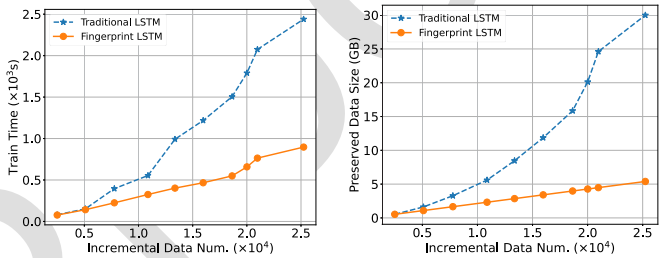


Fig. 8. Comparison of training time and size of data needed to be preserved with different numbers of incremental data.

fingerprint LSTM has local robustness and is more stable in face of noises in sessions.

3) Ablation Study of the Encoding Layer: To verify the effectiveness of the encoding layer, we train a variant version of I²RNN without the encoding layer (EL), which is termed as I²RNN w/o EL in Tables I and II. We can see that the I²RNN with the encoding layer has a better performance than the variant version of I²RNN without the encoding layer. This result indicates that the encoding layer has effects on categorical feature transformation and redundant information reduction.

C. Evaluation of Incremental Learning

1) Comparison of Training Time and Preserved Data Size: Fig. 8 presents the comparison of the training time and the size of data needed to be preserved when giving different numbers of incremental data. Compared to the traditional LSTM, our proposed fingerprint LSTM is more efficient in training time and the size of data needed to be preserved under an incremental learning scenario.

2) Classification Performance in Different Incremental Scenario: We verify the classification performance of I²RNN under different incremental learning scenarios. We consider three incremental scenarios including the addition of one traffic type, the addition of multiple traffic types at once, and the sequential addition of multiple traffic types. We first sort the names of traffic types in alphabetical order. In each incremental scenario, we select the first few traffic types as the incremental traffic types and the remaining traffic types as the initial types. Then we train the model to learn new traffic types according to the incremental scenarios. Table III shows the experiment results. In Table III, the symbol plus (+) means that the corresponding type is added for incremental learning. The *new* means traffic types newly added

TABLE III
CLASSIFICATION PERFORMANCE (F1-SCORE) UNDER DIFFERENT INCREMENTAL LEARNING SCENARIOS

Scenario	Total	New	Old
+Benign	**0.9861**	0.9899	0.9855
+Benign & Botnet	**0.9861**	0.9886	0.9852
+Benign & Botnet →+DDoS & DoS	**0.9861**	0.9891	0.9829

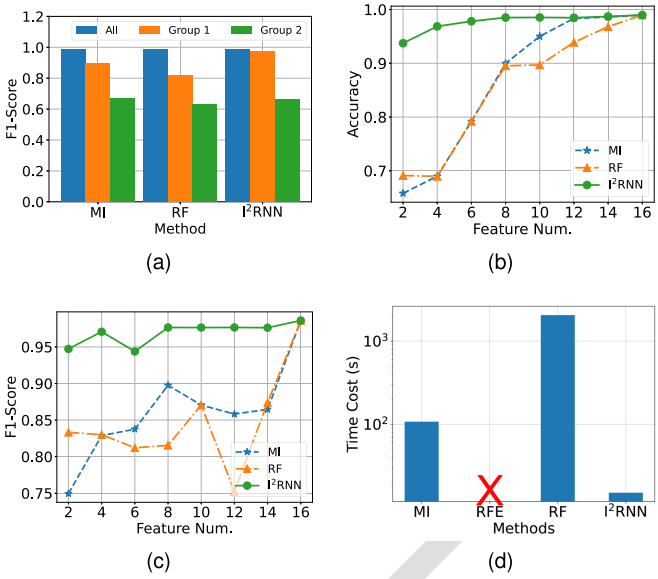


Fig. 9. Experiment results of global model interpretability.

during incremental learning and the *old* means traffic types before incremental learning. We observe that as long as the trained traffic types are the same (including those gradually added for incremental learning), the performance of the I²RNN will be the same. This indicates that incremental learning of I²RNN has the same effect as the training model with full data at once. In other words, our incremental learning algorithm of I²RNN does not cause a degradation in classification performance.

D. Evaluation of Interpretability

For the model interpretability of I²RNN, we first evaluate the global and local interpretability respectively, and then the effectiveness of inter-class distance.

1) Effectiveness of Global Feature Ranking: To examine the effectiveness of features ranked by different methods, we first partition the ranked features into two equal groups, that is, each group has 8 features. Then we use features in each group to classify the traffic. The results are shown in Fig. 9(a). We can observe that the first group of I²RNN has a better F1-Score than the second group. This indicates that the feature ranking of I²RNN is effective. Moreover, we validate the effectiveness of feature ranking by selecting a different number of features according to feature ranking results to perform classification tasks. Fig. 9(b) and (c) show the experiment results. Compared with other feature ranking methods, our method can achieve higher classification performance with the same number of features. This suggests that our feature ranking method can choose the important features of time-series data and is more effective than other feature ranking methods. Particularly, using our feature ranking method can achieve classification performance similar to using all features with only half the number of original features. Finally, we also compare the time cost of different feature ranking methods and the results are shown in Fig. 9(a). We can see that I²RNN uses the shortest time to rank features while other methods need a longer time to handle or are even impossible to get results in a tolerable time (see a red cross in RFE). We argue that the efficiency of the feature ranking method is also a part of the interpretable evaluation, which allows users to have quick feedback and a good user experience.

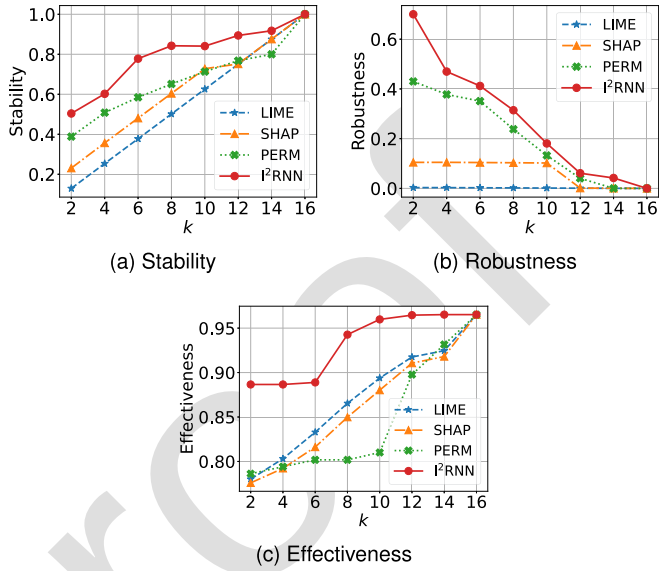


Fig. 10. Performance comparison of local interpretability.

2) Performance Comparison of Local Interpretability: We compare the performance of the local interpretability of I²RNN with LIME [39], SHAP [40], and feature permutation (PERM) [41] in terms of stability, robustness, and effectiveness [20]. Stability is the fundamental metric of local interpretability. The explanation results generated in a critical security system should be stable. That is, the feature importance must not be seriously influenced by the slight fluctuation of the model (e.g., n_epochs = 95, n_epochs = 100, and n_epochs = 105). Robustness is used to measure how similar the explanation results are for similar instances, i.e., the feature importance of samples of the same class should be similar, while the feature importance of samples of different classes should be as distinct as possible. Effectiveness is the measure of whether the explanation results are important to the decision-making. In other words, if the explanation results are really the decision basis for an individual prediction, then after the elimination of such features, the classification result would be changed.

Fig. 10 illustrates the performance comparison of local interpretability in terms of stability, robustness, and effectiveness under different top-k features. From the figure, we can observe that:

TABLE IV
EXPERIMENT RESULTS OF DIFFERENT GROUPING STRATEGIES

Grouping Strategy	Distance	AC (%)	F1 (%)
Botnet & Brute Force	**0.65**	**99.95**	**98.90**
DDoS & DoS	**1.28**	**99.64**	**98.81**
Benign & DDoS	2.67	91.24	85.31
Port Scan & SSH Patator	7.49	79.71	80.84

- The I²RNN shows the best stability, robustness, and effectiveness compared to other methods under different top-k features.
- The ranking of the four local interpretability approaches in term of the stability metric is I²RNN > PERM > SHAP > LIME. The stability of I²RNN is above 0.8 when $k \geq 6$, indicating that explanation features hardly change when slightly modifying the model.
- The ranking of the four local interpretability approaches in term of the robustness metric is I²RNN > PERM > SHAP > LIME. With the increase of k, the robustness of the four methods decreases. For SHAP and LIME, their robustness scores are lower than 0.2, and they hardly change under various k values. The robustness of PERM is less than 0.5. These indicate that PERM, SHAP, and LIME cannot capture the core differences between traffic types especially when k is small.
- The ranking of the four local interpretability approaches in term of the effectiveness metric is I²RNN > LIME ≥ SHAP > PERM. The effectiveness of LIME and SHAP is very similar. For I²RNN, the effectiveness starts to be stable when $k > 8$. This indicates important features of I²RNN are in the top half of the ranked features, which results in little change of predicted label with higher k values.

3) Effectiveness of Inter-Class Distance: To validate whether inter-class distance effectively characterizes the similarity between traffic types, we group traffic types with close distance as an individual traffic type. As depicted in Fig. 5, Botnet and Brute Force are the closest in all features and thus we group them into an individual type. We also group DDoS and DoS which are close in all features. For a fair comparison (i.e., taking out the factor of category reduction), we also group traffic types with large distances. We group Port Scan and SSH Patator which are the farthest types in all features. Benign and DDoS are also grouped as an experiment for large distances. The results are shown in Table 4. We can observe that the performance is improved if we group traffic types that have small inter-class distances (i.e., they are close in some dimensions or in all features), while the performance significantly decreases if we group traffic types with large distances. These results indicate that our inter-class distance is rational, which captures the relationships between traffic types.

VII. DISCUSSION

I²RNN with fingerprint learning has shown a great advantage in encrypted traffic classification. It can be deployed as a model capable of efficient incremental learning to cope with the ever-increasing types of traffic in the real world. The global level feature ranking and selection help eliminate unnecessary features and alleviate the computational overhead. Moreover, researchers and users can understand the deep insights for the model's predictions from the local feature interpretation with high stability, robustness, and effectiveness. Besides, I²RNN profiles the differences and relations between traffic types with the inter-class distance portrait. Users can group similar traffic types to improve classification performance in practice. Below, we discuss the identification of new traffic types, model scalability, and limitations.

Where do the new traffic types come from? Though I²RNN focuses on coping with the incremental learning scenario, with the ability to incrementally learn new types without accessing past data and retraining the model, it is natural to think about where the new traffic types come from. Currently, the newly labeled data for updating the model are manually proposed by the same party who uses the model for detection. We can obtain new application types by collecting relevant information in the application markets such as App Store and Google Play. New types of attacks may be reported by researchers after analyzing system logs and traffic data. Some arts [58], [59] also propose open-set recognition for encrypted traffic classification, which focus on automatically recognizing and labeling new traffic types. I²RNN could leverage these methods to improve the identification of new traffic types and build a more automatic incremental learning process.

What is the scalability of the model? As the number of fingerprint modules increases, the scalability of the I²RNN may become a concern for users. Here we clarify the scalability of I²RNN in terms of space and time. In terms of space, each fingerprint module is lightweight and does not take up too much disk space. For example, the number of parameters of each fingerprint module trained on ISCXVPN2016 is only 5334. The size of the hard disk space occupied by each fingerprint module is only 23.17 KB. Therefore, even if the fingerprint module continues to increase, there will be no obvious scalability problem in space. In terms of time, since each fingerprint module is independent, the network traffic data can be simultaneously fed into each fingerprint module for parallel computation. Therefore, the computing capacity practically depends on the maximum number of parallel threads the CPU allowed. Furthermore, this computational process can be extended to distributed computing clusters using the idea of MapReduce [60]. Fingerprint modules can be deployed on multiple computing nodes. When new traffic data arrives, each computing node can calculate the loss for the input traffic in parallel. After waiting for all computing nodes to complete the calculation, we can aggregate all losses and give the final prediction results by comparing the outputs of each fingerprint module. Therefore, even if the fingerprint module keeps increasing, the model will also maintain good scalability in time.

Limitations and Future Work. There are still two limitations of I²RNN. The first limitation is that the fingerprint may be outdated over time as the behavior of the corresponding traffic type may vary at different times. Future work can develop some systematic strategies to periodically update the fingerprint modules in I²RNN. The second limitation is that hyper-parameters of I²RNN are configured empirically. Future work can investigate

the sensitivity of hyper-parameters, and provide brief guidelines about how to configure them.

VIII. Conclusion

In this paper, we have presented I^{2}RNN, an incremental and interpretable recurrent neural network model for encrypted traffic classification. The I^{2}RNN learns fingerprints from the raw session sequences, which has local robustness. The I^{2}RNN considers an incremental learning scenario. It only needs to train an additional set of parameters for the newly added traffic type rather than retraining the whole model. Moreover, the I^{2}RNN is an interpretable model, which provides the feature attribution globally and locally. It can also depict the similarity between traffic types through inter-class distance. The evaluation results on two public datasets demonstrate that our I^{2}RNN can achieve remarkable performance in terms of encrypted traffic classification, incremental learning, and model interpretability. In performance comparison with well-known local interpretability methods, our I^{2}RNN exhibits excellent stability, robustness, and effectiveness in the interpretation of network traffic data. In future work, we plan to investigate effective techniques for identifying new types of traffic automatically and updating the learned fingerprints to keep pace with the time evolution of traffic.

References

[1] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, "An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification," *Comput. Netw.*, vol. 132, pp. 81–98, 2018.

[2] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Comput. Netw.*, vol. 76, pp. 75–89, 2015.

[3] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "AppScanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 439–454.

[4] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," 2008.

[5] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (SSL) protocol version 3.0" RFC 6101, Tech. Rep., 2011.

[6] L. Vu et al., "Time series analysis for encrypted traffic classification: A deep learning approach," in *Proc. IEEE 18th Int. Symp. Commun. Inf. Technol.*, 2018, pp. 121–126.

[7] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "PortVis: A tool for port-based detection of security events," in *Proc. ACM Workshop Visual. Data Mining Comput. Secur.*, 2004, pp. 73–81.

[8] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, Second Quarter 2014.

[9] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. Int. Workshop Passive Act. Netw. Meas.*, Springer, 2005, pp. 41–54.

[10] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 3366–3383.

[11] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 1, pp. 63–78, Jan. 2018.

[12] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1171–1179.

[13] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, pp. 1999–2020, 2017.

[14] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *Int. J. Netw. Manage.*, vol. 25, no. 5, pp. 355–374, 2015.

[15] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat.*, 2017, pp. 43–48.

[16] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[17] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proc. Web Conf.*, 2020, pp. 13–22.

[18] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 1271–1276.

[19] B. Cui, G. Hu, and S. Yu, "DeepCollaboration: Collaborative generative and discriminative models for class incremental learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 1175–1183.

[20] M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan, and T. Liu, "Can we trust your explanations? Sanity checks for interpreters in android malware analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 838–853, 2021.

[21] F. Constantinou and P. Mavrommatis, "Identifying known and unknown peer-to-peer traffic," in *Proc. IEEE 5th Int. Symp. Netw. Comput. Appl.*, 2006, pp. 93–102.

[22] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," *J. Comput. Syst. Sci.*, vol. 79, no. 5, pp. 573–585, 2013.

[23] M. Korczyński and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 781–789.

[24] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1830–1843, Aug. 2017.

[25] C. Liu et al., "MaMPF: Encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Serv.*, 2018, pp. 1–10.

[26] H. Liu, Z. Wang, and Y. Wang, "Semi-supervised encrypted traffic classification using composite features set," *J. Netw.*, vol. 7, no. 8, 2012, Art. no. 1195.

[27] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of TLS (without decryption)," *J. Comput. Virol. Hacking Techn.*, vol. 14, no. 3, pp. 195–211, 2018.

[28] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Comput. Netw.*, vol. 190, 2021, Art. no. 107974.

[29] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 2367–2380, 2021.

[30] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf.*, 2022, pp. 633–642.

[31] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 3431–3446.

[32] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1285–1298.

[33] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[34] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5138–5146.

[35] Y. Gu, C. Deng, and K. Wei, "Class-incremental instance segmentation via multi-teacher networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 1478–1486.

[36] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E*, vol. 69, no. 6, 2004, Art. no. 066138.

[37] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, no. 1, pp. 389–422, 2002.

[38] G. Louppe, "Understanding random forests: From theory to practice," 2014, *arXiv:1407.7502*.

[39] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144.

[40] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.

[41] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[43] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Hoboken, NJ, USA: Wiley, 2001.
[44] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[45] C. R. Harris et al., "Array programming with numPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
[46] Q. Zou, J. Zeng, L. Cao, and R. Ji, "A novel features ranking metric with application to scalable visual and bioinformatics data classification," *Neurocomputing*, vol. 173, pp. 346–354, 2016.
[47] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[48] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[49] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.
[50] D. McGaughey, T. Semeniuk, R. Smith, and S. Knight, "A systematic approach of feature selection for encrypted network traffic classification," in *Proc. IEEE Annu. Int. Syst. Conf.*, 2018, pp. 1–8.
[51] Y. Zhai and X. Zheng, "Random forest based traffic classification method in SDN," in *Proc. IEEE Int. Conf. Cloud Comput., Big Data Blockchain*, 2018, pp. 1–5.
[52] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *Proc. Netw. Traffic Meas. Anal. Conf.*, 2018, pp. 1–8.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[54] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[55] N. Kokhlikyan et al., "Captum: A unified and generic model interpretability library for pytorch," 2020, *arXiv: 2009.07896*.
[56] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance selection algorithm to reconstruct training set for support vector machine," *Knowl.-Based Syst.*, vol. 116, pp. 58–73, 2017.
[57] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Evanston, IL, USA: Routledge, 2018.
[58] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Serv.*, 2019, pp. 1–10.
[59] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for DL-based traffic classifier update," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 397–405.
[60] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

Zhuoxue Song received the BS degree in computer science and technology from Hangzhou Dianzi University, in 2021. He is currently working toward the MS degree with the security of cyberspace, Zhejiang University. His main research interests include encrypted traffic classification and intrusion detection.

Ziming Zhao received the BS degree in computer science and technology from Shijiazhuang Tiedao University, in 2019. He is currently working toward the PhD degree with the security of cyberspace at Zhejiang University. His main research interests include network security and intrusion detection.

Fan Zhang (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, University of Connecticut, Mansfield, CT, USA, in 2011. He is currently a full professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, and also with the Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou. His research interests include system security, hardware security, network security, and cryptography.

Gang Xiong is currently a full professor and PhD supervisor with the Institute of Information Engineering, Chinese Academy of Sciences, China. He has authored more than 60 papers in refereed journals and conference proceedings. His research interests include network and information security. He is a member of the 3rd Communication Security Technical Committee of China Institute of Communications.

Guang Cheng (Member, IEEE) received the BS degree in traffic engineering from Southeast University, Nanjing, China, in 1994, the MS degree in computer application from the Hefei University of Technology, Hefei, China, in 2000, and the PhD degree in computer network from Southeast University, in 2003. He is currently a full professor with the School of Cyber Science and Engineering, Southeast University. He has authored or coauthored seven monographs and more than 100 technical papers, including top journals and top conferences. His research interests include network security, network measurement, and traffic behavior analysis.

Xinjie Zhao received the BS, MS, and PhD degrees from Ordnance Engineering College, Shijiazhuang, China, in 2006, 2009, and 2012, respectively. He is currently with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His main research interests include side channel analysis, fault analysis, and combined analysis in cryptography. He won the best paper award in COSADE 2012.

Shize Guo received the BS and MS degrees from Ordnance Engineering College, China, in 1988 and 1991, respectively, and the PhD degree from the Harbin Institute of Technology, in 1994. He is currently with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His main research interests include information technology and information security.