Enhanced Relation Network with Contrastive Pairs for Few-shot Encrypted Traffic Classification

Anonymous Author(s)
Submission Id: 7152

ABSTRACT

Performing encrypted traffic classification in the few-shot scenario is necessary for network management and cyberspace security because of labor-intensive labeling and intrinsically rare samples. Meta-learning methods have been widely used to deal with this problem, which classify new traffic samples via comparing the similarity of all traffic inputs in the meta-learned feature space. However, existing methods may fail to preserve the consistency between the similarity of traffic features and their category information, leading to a sub-optional performance. In this paper, we proposed a novel method, namely Enhanced Relation Network with Contrastive Pairs (ERIN) for few-shot encrypted traffic classification. ERIN integrates contrastive learning in a supervised manner into the meta-learning framework, which enhances the similarity of traffic samples belonging to the same type, improving the performance of existing methods. Moreover, we optimize the conventional relation network by highlighting the informative traffic samples, which helps it generate a more comprehensive class representation for each traffic type, allowing ERIN to fully utilize the limited traffic samples. Furthermore, we evaluate the effectiveness of ERIN on two real-world public encrypted traffic datasets, and the experimental results demonstrate the effectiveness of ERIN.

CCS CONCEPTS

• **Networks** → **Network manageability**; • **Computing methodologies** → **Artificial intelligence**; • **Information systems** → *Data management systems.*

KEYWORDS

Encrypted Traffic Classification, Few-shot Learning, Contrastive Learning, Meta-learning

ACM Reference Format:

Anonymous Author(s). 2022. Enhanced Relation Network with Contrastive Pairs for Few-shot Encrypted Traffic Classification. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17-22, 2022, Atlanta, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

CIKM'22, October 17-22, 2022, Atlanta,USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Encrypted traffic classification aims to associate the encrypted traffic flows with the applications or application types that generated them, which is vital in network management and cyberspace security [16], attracting a lot of attention from both academia and industry[2, 13, 28]. In the face of obfuscated and encrypted traffic which precludes payload analysis, many researchers applied Machine-Learning (ML) based methods to classify traffic type via leveraging the statistical features of the traffic flow. In recent years[1, 10, 22], Deep-Learning (DL) based methods[18] have achieved remarkable successes in encrypted traffic classification tasks due to their automatically feature selection and considerably high capacity of learning under the premise of a large amount of labeled encrypted traffic. In general, various neural networks have been exploited in existing researches, including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), self-attention, Generative Adversarial Networks (GAN) and so on.

In fact, the success of DL-based methods has largely been in the situation where vast quantities of traffic data can be collected[18]. However, network environment is constantly changing and new applications are periodically emerging, making collecting numerous labeled encrypted traffic in time for classifier updating is time-consuming and labor-intensive. Moreover, there are many applications where traffic data is intrinsically rare, such as zero-day attack[33]. Therefore, few-shot encrypted traffic classification approaches have been proposed to address this problem[3, 19, 23, 33, 36], which aim to recognize new traffic types via fully exploiting information contained in limited samples, mitigating the dependence of numerous annotated traffic data.

Data augmentation[34] and meta-learning technologies[3, 7, 23, 33, 35, 36] have been applied in encrypted traffic classification methods to deal with the problem of few-shot learning. The core idea of data augmentation[34] is to increase the number of labeled samples by generating pseudo encrypted traffic. In fact, although the number of labeled samples has increased, whether the generated pseudo encrypted traffic really exists is still questioned. The main idea of meta-learning based methods is to learn universal prior knowledge from numerous encrypted traffic classification tasks, facilitating the quick adaptation to new traffic types with only a few samples. Existing methods leverage different networks (e.g., triplet network or relation network) to train a universal feature extractor. Then, it is transferred to previously unseen new traffic types, and classifying the new traffic types via computing the similarity between learned features. Those methods are required that the similarity between the learned embedding features is consistent with the category relations of traffic inputs. However, the similarity of learned features belonging to the same type may not be preserved in those methods, leading to a sub-optional performance.

In this paper, we proposed a new method namely Enhanced Relation Network with Contrastive Pairs (ERIN) for improving the performance of few-shot encrypted traffic classification, which enhances the similarity of learned features belonging to the same type via learning an intra-class compactness aware representation. Specifically, ERIN contains traffic featurization module, contrastive-aware representation module, prototype rectification module and deep similarity kernel module. Firstly, all traffic inputs are transformed into efficient features in the traffic featurization module. Then, the learned features belonging to the same type are imposed close to each other in the contrastive-aware representation module, where contrastive pairs are used to add constraints to the intra-class distance, avoiding misclassification of few-shot traffic into confusable types. At the same time, prototype rectification generates a more comprehensive class representation by highlighting the informative traffic samples, allowing the deep model to fully utilize the limited samples. At last, the deep similarity kernel gives the new traffic input recognition results by comparing it with the similarity of all class representations produced in the prototype rectification module.

Our contributions can be briefly summarized as follows:

1. We proposed an effective method namely ERIN to improve the performance of few-shot encrypted traffic classification, which learned intra-class compactness aware traffic representations via integrating contrastive learning in a supervised manner into meta-learning based methods.

2. We optimize the class representation of the conventional relation network in the prototype rectification module via assigning different weights for each traffic sample, which highlights the informative traffic samples, allowing the deep model to fully utilize limited traffic samples.

3. We conduct extensive experiments on USTC-TFC2016 and UNB-ISCX two real-world public encrypted traffic datasets to evaluate the superiority of ERIN. Moreover, our ablation analysis experiments demonstrate the effectiveness of the contrastive-aware representation module and prototype rectification module.

The rest of the paper is organized as follows. We first retrospect the most relevant work to our methods in Section 2. Then, the preliminaries are described in Section 3. In Section 4, we provide the details of our proposed ERIN. Then, in Section 5, we present the experimental results and analysis. Finally, the paper is concluded in Section 6.

2 RELATED WORK

Previous works on encrypted traffic classification involve many different techniques, and few-shot encrypted traffic classification is relatively new research to address the problem of inadequate encrypted traffic samples. In this section, We retrospect the most relevant work to our methods in those two parts.

2.1 Conventional Encrypted Traffic Classification Approaches

Many researchers are devoted to finding machine learning models that leverage statistical features of the traffic flow to determine its class, i.e., C4.5 decision tree, support vector machine (SVM), Bayes Network, Naive Bayes and Markov model[1, 10, 22]. Recently, many studies have shown that it is possible to design DL algorithms to classify different traffic types[2, 11, 28–32]. Wang et al. make an analogy between traffic packet and image and leveraged the 2D-CNN to learn the representation of raw traffic flow, achieving excellent performance on real-world datasets[30]. On the other hand, study[29] considers original traffic data as text and utilizes 1D-CNN to generate representative features of each traffic example. In order to capture both spatial and temporal features of raw traffic data, study[31] proposed a mixed model which contains CNN and RNN. Recently, many researchers have tended to capture more long-range dependencies of original traffic data to implement encrypted traffic classification by applying self-attention mechanism[2, 32]. In addition, besides original traffic data, raw flow sequences can also be fed to various neural networks as a representation of traffic flow[11, 28]. In study[11], Liu et al. proposed an end-to-end classification model called FS-NET, which adopts an encoder-decoder structure to learn representative features from the raw flow sequences and classifies them. Wang et al. utilize the original traffic data and raw flow sequences at the same time to generate the representation of traffic flows[28]. In general, these deep learning-based methods usually make an analogy between the raw traffic data and images or text, and adopt it to various neural networks.

However, those methods have clear limitations. For instance, the excellent performance achieved by those methods heavily depends on the vast quantities of training data. Moreover, those methods cannot classify traffic types previously unseen in the training dataset. Therefore, there is a need for few-shot encrypted traffic classification methods which is able to classify unseen traffic types with only limited samples.

2.2 Few-shot Encrypted Traffic Classification

Few-shot Learning. Few-shot learning aims to solve classification problems with a small number of examples and make the deep learning model generalize better to test classes[6]. Meta-learning is an effective technique to realize the aim of few-shot learning[5], which has metric-learning based [8, 15, 17, 24–26] based, optimization-based [4, 21] and model-based[20, 27] three forms[6]. Optimization-based techniques explicitly optimize for fast learning, while model-based techniques rely upon an adaptive internal state[6]. In this paper, we mainly focus on metric-based meta-learning methods, in which our method belongs. Koch et al. proposed siamese neural network to compute the representation of input samples and the distance vector between them[9]. Inspired by augmenting neural networks, Vinyals et al.[26] proposed the matching network, which maps a small labeled support set and an unlabelled example to its labels without fine-tuning. Prototypical network[24] are proposed by Snell et al. to reduce the complexity of existing methods for meta-learning, where the model generates a prototype representation of each class and performs classification via computing distance between the query examples to them. Matching network and prototypical network used a fixed metric to measure the similarity of inputs. On the contrary, relation network[25] aims to learn a transferrable deep metric for comparing the similarity between the query examples and support examples. Therefore, we choose relation network as our basic meta-learning framework.

Figure 1: The illustration of the few-shot encrypted traffic classification tasks based on meta-learning framework.

Few-shot Encrypted Traffic Classification. In the encrypted traffic classification field, metric-based and optimization-based meta-learning techniques have been applied to solve the problem of few-shot learning[3, 7, 23, 33, 35, 36]. Feng et al. exploited the Model-Agnostic Meta-Learning (MAML) as the meta-learning framework to learn a few-shot class-adaptive anomaly detection model[3]. Compared to optimization-based methods, metric-based methods are the most well-studied meta-learning forms because they are simple but effective. Triplet Fingerprinting (TF)[23] leverages the triplet network to generate features for new traffic inputs, and then performs Website Fingerprinting (WF) attacks via K-Nearest Neighbors (KNN) classifier with limited samples. Rong et al. proposed a prototype-based few-shot learning model that identifies unseen malware via computing Euclidean distance to prototype representation of each class[19]. FC-Net[33] and RBRN[36] choose relation network[25] as the meta-learning framework to perform few-shot encrypted traffic classification, which measures the similarity of traffic features via a learnable deep metric namely relation module. Moreover, Festic combines multiple metrics to enhance the classification capability of a single metric[35].

However, these methods may not preserve the similarity of the learned features belonging to the same type in the feature space, degrading the performance of similarity-based few-shot tasks.

3 PRELIMINARIES

Meta-learning is an effective approach to realizing the aim of few-shot learning. In this section, we first introduce the basic concept of meta-learning. Then, we define the few-shot encrypted traffic classification problem, which needs to be solved in our study.

3.1 Meta-learning

In conventional machine learning, the deep model requires a large number of training samples to achieve good performance[14]. This requirement severely limits the ability of deep models to learn a new concept quickly, which is one of the defining aspects of human intelligence[6]. Meta-learning has been suggested as one strategy to overcome this challenge, providing an alternative paradigm in which a machine learning model gains meta-knowledge over multiple related tasks and uses this knowledge to improve its future learning performance on new tasks[5, 6], known as "learn to learn". The meta-knowledge is obtained from numerous meta-training tasks used to teach machine learning model how to learn. Each task is formed by randomly sampling N classes and K labeled samples per class from the meta-training data source as the support set, and a fraction of the rest samples from the N classes as the query set. Support set literally supports classification decisions on the query set. Meta-testing tasks also have the support set and the query set. It is worth noting that meta-testing data sources contain previously unseen types with limited samples, which are expected to be identified accurately by the machine learning model with meta-knowledge.

3.2 Problem definition

Assume that we have C base traffic types D^{base} which contains numerous labeled data and N new traffic types with limited samples namely few-shot traffic types D^{new}. As shown in Figure 1, base traffic types are used as the data source to conduct meta-training tasks $\mathcal{T}_1 = \{T_A^{tr}, T_B^{tr}, T_C^{tr}, \cdots\}$ to train the traffic classifier ($D^{tr} = D^{base}$). Then, this classifier is used to perform meta-testing tasks $\mathcal{T}_2 = \{T_A^{test}, T_B^{test}, T_C^{test}, \cdots\}$which formed via randomly sampling from the few-shot traffic types and base traffic types ($D^{test} = D^{base} \cup D^{new}$). Our aim is to obtain the universal meta-knowledge about traffic classification from the meta-training task set $\mathcal{T}_1$, and adopt it to rapidly learn new tasks in $\mathcal{T}_2$ which contains the few-shot traffic types. Therefore, even though the few-shot traffic types are not available during meta-training, the classifier could also assign the class label $\hat{y}$ to the few-shot traffic sample x_i^q with limited labelled samples in the support set when performing meta-testing tasks.

4 ERIN FRAMEWORK

In this section, we give the details of the proposed ERIN. At first, we will present the overall framework of ERIN, and then each component of ERIN will be introduced in detail.

4.1 System Overview

The ERIN architecture is given in Figure 2, consisting of traffic featurization module, contrastive-aware representation module, prototype rectification module and deep similarity kernel module. As illustrated in Section 3, the data source used by each few-shot encrypted traffic classification task contains the support set and query set. The classifier sees the support set and extracts information from it to guide its predictions on the query set. The key issue for ERIN is how to better identify samples in the query set with limited supported samples via leveraging the prior traffic classification knowledge.

At first, the traffic examples in the support set and the query set are fed into the traffic featurization module, consisting of traffic normalization and embedding generator. With the traffic normalization module, all input traffic samples are transformed into raw traffic payload byte sequences with a uniform length of 1500, which is the required format of ERIN. Then, the embedding generator is used to extract representative features for all payload sequences.

Next, ERIN enhances the similarity between the learned features belonging to the same type via applying supervised contrastive learning in the contrastive-aware representation module. In this module, ERIN employs extra restrictions for the learned traffic features, which compacts intra-class variations while simultaneously

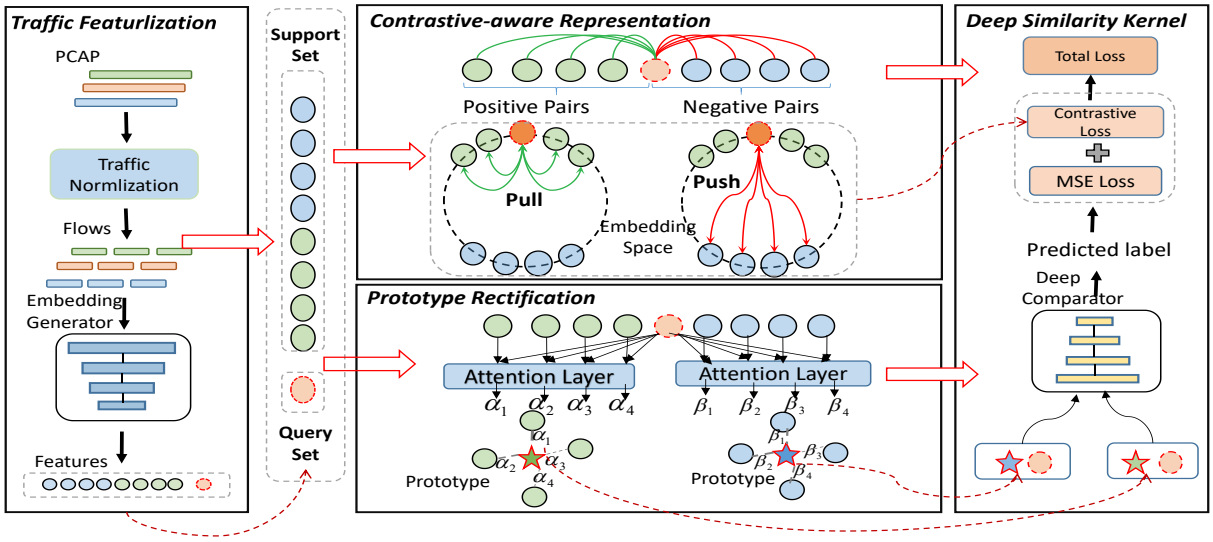


Figure 2: The architecture of ERIN. All traffic inputs are converted to efficient features in the traffic featurization module, and then the contrastive-aware representation module will enhance the similarity of learned features belonging to the same type. At the same time, prototype rectification generated a comprehensive support class representation for computing the similarity with the query sample in the deep similarity kernel.

separating inter-class differences, allowing ERIN to generalize better to few-shot traffic types. Moreover, in order to generate a more comprehensive feature to represent a specific traffic class in support set, prototype rectification module is utilized to determine the contribution of each support traffic example.

At last, the classification results of traffic samples in the query set are given by the deep similarity kernel, which measures the similarity between the learned feature of the query example and prototype of each traffic type in the support set.

4.2 Traffic Featurization

Traffic Featuraization consists of traffic normalization and embedding generator two components. In traffic normalization, raw traffic data (PCAP format) are transformed into payload byte sequences with a uniform length of 1500, which are the required input data format of ERIN. Then, the payload sequences are mapped to another low-level dimension representation in the learned feature space through the embedding generator.

4.2.1 Traffic Normalization. The basic traffic unit we applied in our work is flow, which is defined as packets with the same quintuple [Source IP, Destination IP, Source Port, Destination Port, Protocol]. Following previous works[12, 28, 29], we truncate (or zero-padding) the traffic flows at a fixed length to employ DL-based methods. The details of traffic normalization are described as follows:

(1)Traffic Segmentation: The first step is to divide the raw traffic data into multiply discrete traffic flows. The tool we used to divide the raw traffic into flows is SplitCap, which takes a PCAP format file as input and outputs a series of smaller bin files based on quintuple [Source IP, Destination IP, Source Port, Destination Port, Protocol].

(2)Address Anonymization: The goal of the second step is to randomize the Media Access Control (MAC) address and IP address in the data link layer and the IP layer, respectively. The reason for address anonymization is that IP address and MAC address should not be used for features when the traffic data are collected from different network environments[12, 29].

(3)Byte Sequence Unitization: In the last step, we generate the unified representation for all traffic flow. We choose the first 1500 bytes of payload in the application layer to represent the whole flow. In addition, all bytes in sequence are divided by 255(the maximum value for a byte) to ensure the input values are in the range[0,1].

4.2.2 Embedding Generator. The embedding generator module extracts efficient features for all traffic inputs. The inputs of the embedding generator are also called an episode, which contains $n_s = N \times K$ support traffic samples and n_q query traffic samples. We denote the $\mathcal{S} = \left\{\left(x_j^s, y_j^s\right)\right\}_{j=1}^{n_s}$ as the support set, and we denote S^k as the support subset of the k^{th} class. In addition, the query set can be written as $Q = \left\{\left(x_i^q, y_i^q\right)\right\}_{i=1}^{n_q}$. We stacked four convolution blocks to obtain the more representative features of the traffic samples, which are identical to relation network[25]. More concretely, each convolution block contains 3 × 3 convolutional layer(with 64 filters), batch normalization layer, rectified linear units(ReLu) nonlinear activation function three components. In particular, the first and third convolution block contains a 2×2 max-pooling layer, while the second and last convolution block does not. In addition, we add a dropout layer to help the embedding generator avoid overfitting. We denote Φ as the embedding network, and the output feature of the support traffic samples and a query traffic sample through embedding network Φ can be written as follows:

$$f^{\mathcal{S}} = \Phi(\mathcal{S}),\ f_i^q = \Phi(x_i^q). \tag{1}$$

Through embedding generator, we obtain the learned features of support traffic samples and a query traffic sample, and those learned traffic features will be optimize directly in subsequent contrastive-aware representation module.

4.3 Contrastive-aware Representation Module

Despite of the efficient features generated by the embedding generator, if we directly used those features for deep similarity kernel in the subsequent step to classify, the performance of few-shot encrypted traffic classification will be sub-optional, since the similarity of the learned features belonging to the same type may not be preserved in the embedding space. Therefore, a compact embedding space with compact intra-class distance is required to improve the performance of the deep similarity kernel.

To deal with this problem, we integrate supervised contrastive loss into existing few-shot encrypted traffic classification methods. We first construct contrastive pairs by using the traffic samples in the support set and query set. Specifically, we take a query sample as an anchor, the support samples with the same label as its positive pairs, and the negative pairs are constructed with the support samples with different labels. We denote the positive pairs and negative pairs as follows:

$$\mathcal{P}: \quad \{(x_i^q, y_i^q), (x_p^s, y_p^s)\} \in \mathcal{P} \quad if\ y_i^q = y_p^s \tag{2}$$

$$\mathcal{N}: \quad \{(x_i^q, y_i^q), (x_n^s, y_n^s)\} \in \mathcal{N} \quad if\ y_i^q \neq y_n^s \tag{3}$$

To minimize the distance between the learned features belonging to the same type, we employ a constraint that enables optimizing those features directly from the embedding space, which can be formalization as following:

$$\min \sum_{\{(x_i^q, y_i^q), (x_p^s, y_p^s)\}) \in \mathcal{P}} d(f_i^q, f_p^s) \tag{4}$$

where d is L_2- norm distance:

$$d(f_i^q, f_p^s) = \|f_i^q - f_p^s\|^2 \tag{5}$$

Moreover, to help the deep model better distinguish the learned features belonging to the different categories, we enlarge the distance between samples in negative pairs simultaneously. Therefore, another constraint is used to ensure the distance of the negative pair $\{(x_i^q, y_i^q), (x_k^s, y_k^s)\}$ be larger than a predefined margin γ:

$$d\left(f_i^q, f_n^s\right) \geq \gamma \tag{6}$$

The supervised contrastive loss is used to implement the constraint of positive pairs and negative pairs during model training. The definition of supervised contrastive loss in our study can be written as follows:

$$\begin{aligned} L_{CL} = & \sum_{\{(x_i^q, y_i^q), (x_p^s, y_p^s)\} \in \mathcal{P}} \underbrace{d\left(f_i^q, f_p^s\right)}_{\text{intra-class constraint}} \\ & + \beta \underbrace{\sum_{(x_i^q, y_i^q), (x_n^s, y_n^s) \in \mathcal{N}} \max\left\{-d\left(f_i^q, f_n^s\right) + \gamma, 0\right\})}_{\text{inter-class constraint}} \end{aligned} \tag{7}$$

where β is a hyperparameter to balance the inter-class and intra-class constraints.

Benefiting from supervised contrastive loss, a robust embedding generator will be trained to generate more discriminative features with compact intra-class compactness and inter-class dispensation, allowing ERIN to generalize better to the few-shot traffic types.

4.4 Prototype Rectification

Prototype is defined as a class feature map representing a class in the support set, and the query traffic sample will have the same label as the nearest class prototype. Original relation network element-wise sums over the learned features of all samples belonging to the same type in the support set to form its class prototype. However, not every traffic sample in the support set is informative. For example, we consider the traffic samples containing numerous padding bytes 0x00 to be less informative. The prototype generated by the original relation network neglect the different contributions of each example in the support set. Therefore, assigning the different weights to each sample in the support set is more reasonable when generating the prototype.

In order to fully utilize the limited samples in the support set, we proposed the prototype rectification module to highlight the important examples while mitigating the adverse effect of useless information. The rectified prototype C_k can be written as the following formula:

$$C_k = \sum_{x_j^s \in \mathcal{S}^k} \alpha_j f_j^s \tag{8}$$

where α_j is the weight of the j^{th} sample in $\mathcal{S}^k$ for prototype computing. The α_j is outcome of e_j though softmax:

$$\alpha_j = \frac{\exp\left(e_j\right)}{\sum_{e_a \in \mathcal{S}^k} \exp\left(e_a\right)} \tag{9}$$

e_j reflect the importance of the j^{th} sample in $\mathcal{S}^k$ to query example x_i^q, which is defined as following:

$$e_j = \text{sum}\left\{\sigma\left(g\left(f_j^s\right) \odot g\left(f_i^q\right)\right)\right\} \tag{10}$$

For computing e_j, we perform a linear transformation $g(\cdot)$ in the first step. Then, the inner production $\odot$ of $g(f_j^s)$ and $g(f_i^q)$ is computed. In addition, σ is an activation function which squeeze the value among [-1,1]. Last, sum operation is used to add all the elements of the vector to gain e_j.

Compared to the conventional relation network, the prototype rectification produces more comprehensive class representations by assigning different weights for each sample in the support set, allowing the deep model to fully utilize the limited samples.

4.5 Deep Similarity Kernel

Deep similarity kernel measures the similarity of support-query feature map pairs to determine if they are matching. In previous studies[24, 26], a fixed metric (e.g., Euclidean) is used to measure similarity between two input vectors. However, a fixed metric lead to critical dependence on the learned embedding network. Therefore, a learnable non-linear similarity metric[25] is used in the deep similarity kernel to mitigate the dependence of the embedding network by learning a transferrable deep metric.

Deep comparator is the key component of the deep similarity kernel, which contains two convolutional blocks and two fully-connected layers. Each convolutional block consist of 3×3 convolution layer with 64 filters, batch normalization, ReLU non-linearity activation function and 2×2 max-pooling layer. The output dimension of the last convolutional block is 64, and then fed the vector to two fully-connected layers which are 8 and 1 dimensional, respectively. The output of the last fully-connected layer is the relation

Table 1: The statistical information of malware in USTC-TFC2016 dataset.

Malware	# of Flows	Malware	# of Flows
Cridex	16023	Nsisay	10917
Geodo	13540	Shifu	15468
Htbot	12425	Tinba	15879
Miuref	9576	Virut	12928
Neris	16708	Zeus	11823

Table 2: The statistical information of regular encrypted traffic characterization in UNB-ISCX dataset.

Class	# of Flows	Class	# of Flows
Email	6354	File transfer	2730
P2P	4712	Chat	3912
Streaming	10182	VoIP	10324

score of support-query embedding pairs, which ranges from 0 to 1. Relation score r_{ik} represents the similarity between the query sample x_i^q and the $\mathcal{S}^k$, which can be written as follow:

$$r_{ik} = R_\phi(Concat(C_k, f_i^q)) \quad i = 1, 2, ..., N \tag{11}$$

The output of the deep similarity kernel conceptually can be seen as a regression problem because it predicts a relation score ranging from 0 to 1. Therefore, Mean Square Error (MSE) loss is used to train the model.

$$L_{MSE} = \sum_{k=1}^{N} (r_{ik} - 1(y_i == y_k))^2 \tag{12}$$

We adopt both supervised contrastive loss and MSE loss as joint supervision to train our ERIN, the formulation can be written as follows:

$$L_T = \lambda L_{CL} + (1 - \lambda) L_{MSE} \tag{13}$$

where λ is a hyperparameter to balance the two supervision signals.

5 EXPERIMENTAL EVALUATION

Our experiments mainly to answer the following questions:

- **Q1:** Compare to the other SOTA baselines, could ERIN better recognize the few-shot traffic types?
- **Q2:** Could the contrastive-aware representation module and prototype rectification module improve the performance of ERIN?
- **Q3:** How do different choices of parameters affect the performance of ERIN?

To answer those questions, we evaluate ERIN on few-shot malware identification and few-shot traffic characterization two benchmark tasks. In the following, we present the details of datasets, experiment settings, experimental results and analysis.

5.1 Experimental Setup

Datasets. We evaluate ERIN on two public widely-used encrypted traffic datasets, including USTC-TFC2016 and UNB-ISCX. USTC-TFC2016 is explored as the data source of few-shot malware identification, and UNB-ISCX is used to validate the effectiveness of ERIN on few-shot traffic characterization. The detailed information is described as follows:

(1)USTF-TFC2016: USTF-TFC2016 contains ten types of malware collected from CTU researchers and ten types of normal traffic collected by IXIA BPS[30]. We aim to accurately identify the few-shot malware types previously unseen malware with limited traffic samples in meta-testing tasks. Therefore, we select ten malware types as the data source to construct few-shot malware identification tasks, and the statistical information of those malware traffic types is shown in Table 1.

(2)UNB-ISCX: UNB-ISCX dataset contains 7 classes of regular encrypted traffic and 7 classes of protocol encapsulated traffic types. We choose the regular encrypted traffic types as our experimental data source because the encapsulated traffic samples are insufficient to construct base characterization types for obtaining enough prior knowledge. As the "Browser" is hard to label, we choose six categories of the seven regular encrypted traffic types, and the statistical information of regular encrypted traffic types is shown in Table 2.

Few-shot Tasks Settings. *(1)Few-shot Malware Identification:* To evaluate the identification of ERIN for the few-shot malware type, for each experiment, we pick one of the ten malware as the few-shot malware type which is previously unseen in meta-training tasks. For example, we pick Cridex as the few-shot malware type, and the other nine malware forms the base malware types. Base malware types contains abundant samples as the data source to construct the meta-training and meta-testing tasks. In comparison, we add a few Cridex traffic samples only to the meta-testing data source. In this way, the traffic samples of Cridex are limited and not available during the meta-training phase. We aim to identify Cridex accurately in meta-testing tasks.

In few-shot malware identification tasks, we select 5000 traffic samples for each base malware type to construct the meta-training task set. For each meta-training task, we randomly sampled two traffic classes from base traffic types to help ERIN learn how to distinguish between traffic samples belonging to different types. More concretely, each sampled traffic class contains thirty samples, twenty out of them are used as the support set and the left ten samples forms query set. As for meta-testing, 300 traffic samples of all malware (including base malware type and few-shot malware type) are sampled as the data source of the meta-testing task set. The construction of the meta-testing task is similar to the meta-training task. However, note that the few-shot malware type is only included in the meta-testing data source.

(2)Few-shot Traffic Characterization Identification: We iteratively pick one of the encrypted traffic types as the previously unseen few-shot characterization type, which is only included in the meta-testing data source. Similar to few-shot malware identification tasks, the left five encrypted traffic types form the base characterization types. In the few-shot traffic characterization benchmark, we select 2500 traffic samples for each base characterization type as the data source to construct the meta-training task set. The setup of each task in meta-training is consistent with few-shot malware identification. Meta-testing tasks are generated using base characterization types and few-shot characterization type, each of them has 150 samples.

Evaluation Metrics. In order to evaluate the identification of ERIN for the few-shot traffic types, we consider the four evaluation metrics: precision, recall, F1-score and accuracy. In each task of the meta-testing task set, precision measures the ratio of true few-shot

Table 3: Comparison results on ten parallel experiments of few-shot malware identification tasks.

Method	Metric	Few-shot Malware Type										
		Cridex	Geodo	Htbot	Miuref	Neris	Nsisay	Shifu	Tinba	Virut	Zeus	Overall
ERIN	PRE.	**0.9597**	**0.9354**	**0.9202**	0.9748	0.9003	0.9160	**0.9646**	0.9101	0.9168	**0.9267**	**0.9325**
	RE.	0.9322	**0.9570**	**0.9219**	**0.9908**	0.9085	0.9231	**0.9847**	**0.9720**	**0.9329**	**0.9853**	**0.9508**
	F1.	**0.9457**	**0.9460**	**0.9211**	0.9827	0.9044	0.9195	**0.9746**	**0.9400**	**0.9248**	**0.9551**	**0.9414**
	ACC.	**0.9465**	**0.9454**	**0.9210**	0.9826	0.9040	0.9192	**0.9743**	**0.9380**	**0.9241**	**0.9537**	**0.9409**
RBRN	PRE.	0.9354	0.9177	0.8414	0.8557	**0.9528**	**0.9244**	0.7913	0.8763	**0.9469**	0.8858	0.8928
	RE.	0.8941	0.7844	0.9063	0.9565	**0.9767**	**0.9258**	0.9136	0.9567	0.8522	0.9481	0.9114
	F1.	0.9143	0.8458	0.8726	0.9033	**0.9646**	**0.9251**	0.8481	0.9147	0.8970	0.9159	0.9001
	ACC.	0.9162	0.8570	0.8677	0.8976	**0.9642**	**0.9250**	0.8364	0.9108	0.9022	0.9130	0.8990
FC-Net	PRE.	0.9180	0.8864	0.8235	0.8957	0.9208	0.9118	0.8816	0.9244	0.9297	0.9118	0.9004
	RE.	0.9333	0.9750	1.0000	0.9364	0.9300	0.8857	0.9571	0.9167	0.9154	0.9538	0.9403
	F1.	0.9256	0.9286	0.9032	0.9156	0.9254	0.8986	0.9178	0.9205	0.9225	0.9323	0.9190
	ACC.	0.9250	0.9250	0.8929	0.9136	0.9250	0.9000	0.9143	0.9208	0.9231	0.9308	0.9170
TF	PRE.	0.8333	0.7931	0.7826	**1.0000**	0.7826	0.7143	0.9310	0.7391	0.7407	0.7419	0.8059
	RE.	**1.0000**	0.7667	0.6000	0.9667	0.6000	0.6667	0.9000	0.5667	0.6667	0.7667	0.7500
	F1.	0.9091	0.7797	0.6792	**0.9831**	0.6792	0.6897	0.9153	0.6415	0.7018	0.7541	0.7733
	ACC.	0.9333	0.9133	0.8867	**0.9933**	0.8867	0.8800	0.9444	0.9367	0.9433	0.9500	0.9268

samples to traffic samples that are predicted as few-shot samples. Recall measures the ratio of few-shot samples that are successfully discovered by ERIN to all the few-shot traffic samples. F1-score is the harmonic mean of precision and recall: $F_1 = 2\times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. In addition, accuracy measures the ratio of all correctly predicted samples to all traffic samples in the query set.

Baseline Methods. We choose three SOTA metric-based few-shot encrypted traffic classification methods as our baselines, including FC-Net[33], RBRN[36] and TF[23]. The details of those methods can be briefly described as follows:

(1)FC-Net: FC-Net consists of F-Net and C-Net. F-Net is used to extract the features of all traffic samples, and the C-Net gives the recognition results via comparing the similarity of the sample feature pairs.

(2)RBRN: RBRN utilizes the relation network as the meta-learning framework, which also contains feature extraction and similarity comparison. Note that the similarity comparison in RBRN is between types and samples instead of sample feature pairs.

(3)TF: TF leverages the triplet network for few-shot learning, including pre-training and attack two phases. The outcome of the pre-training phase is a feature extractor that is used to help the classifier in attack phase to perform traffic classification.

Implementation. Our model is implemented with Tensorfolw 1.15 and trained on a Tesla P100. To develop ERIN, we carefully select the hyperparameters to achieve the best performance. We adopt the episode training mechanism to generate meta-training tasks[26]. We use a query sample as an anchor, and use all traffic samples in the support set to construct contrastive pairs according to the type relationship. We set the balance factor λ in total loss to 0.6, set the proportion of inter-class distance β in contrastive loss to 0.02, and set the number of convolution kernels in the embedding generator and deep similarity kernel to 64. In addition, we employ 0.8 dropout rate in the first and third convolution layer of the embedding generator to prevent overfitting. We train ERIN by Adam Optimizer with 0.001 learning rate and 500 epochs, containing 10000 $\times$ 500 episodes in total.

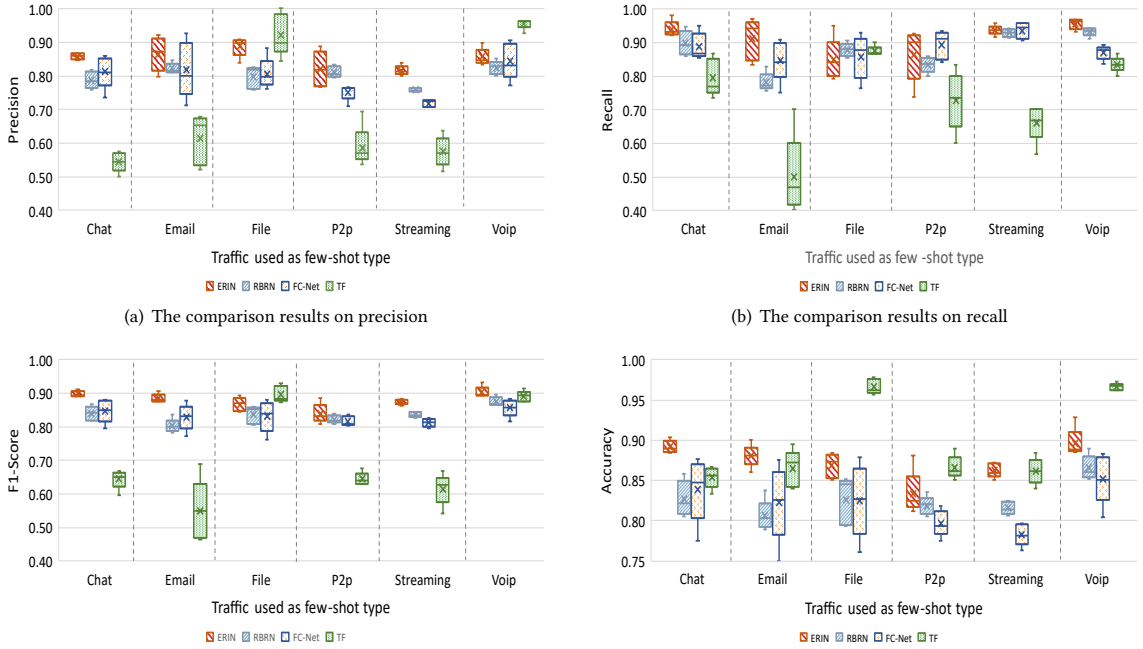
5.2 Comparison Study (RQ1)

In this section, we aim to compare ERIN to three SOTA methods for few-shot encrypted traffic classification on two benchmark tasks. For each benchmark task, we iteratively pick one of the traffic types in the dataset as the few-shot type to demonstrate the generalization of the results. Therefore, ten parallel experiments on few-shot malware identification and six parallel experiments on few-shot traffic characterization are conducted. Moreover, to generate the rigour of experiments in the few-shot scenario, we generate 3000 tasks in the meta-testing phase and repeat each experiment multiple times for evaluation in each parallel experiment. The average performance is reported in Table 3 and Figure 3. Based on the results, we can make the following observations:

Results on Few-shot Malware Identification: 1. ERIN can identify few-shot malware type accurately and achieves the best performance on overall accuracy and F1-score. As shown in Table 3, the average accuracy across ten experimental settings of ERIN is 94.09%, which is slightly higher than the second-best method TF. However, the overall F1-Score of ERIN is obviously superior to TF. It is also notable that ERIN outperforms all baseline methods on 7 out of 10 experimental settings in terms of accuracy and F1-score. Such good results strongly demonstrated the effectiveness of ERIN for few-shot malware identification.

2. Compared to TF, ERIN achieves a more consistent performance across different experimental settings. For example, the F1-score of TF is 98.31% when the Miuref is set to the few-shot malware type in Table 3. However, the performance of TF is dramatically decreased when the few-shot type is set to Neris. In comparison, ERIN achieves more than 90% in terms of F1-score in all experimental settings. The consistent performance of ERIN benefits from the combination of contrastive loss and MSE loss, which optimizes the deep model via utilizing information from learned feature space and output space simultaneously.

3. Compared to RBRN and FC-Net, ERIN is prone to perform well on both precision and recall simultaneously. RBRN and FC-Net are conventional relation network based methods. As can be



(a) The comparison results on precision

(b) The comparison results on recall

(c) The comparison results on F1-score

(d) The comparison results on accuracy

Figure 3: The distribution of comparison results on six parallel experiments of few-shot traffic characterization tasks.

Table 4: Performance comparison between ERIN and three variants.

Few-shot Task	Method	Component		Metric			
		CAR	PR	PRE.	RE.	F1.	ACC.
Malware Identification	RN	×	×	**0.9399**	0.8500	0.8925	0.8979
	RN w/o PR	×	✓	0.9117	0.8942	0.9029	0.9040
	RN w/o CAR	✓	×	0.9300	0.9495	0.9397	0.9390
	ERIN	✓	✓	0.9325	**0.9508**	**0.9414**	**0.9409**
Traffic Characterization	RN	×	×	0.8710	0.8100	0.8394	0.8450
	RN w/o PR	×	✓	0.8571	**0.8400**	0.8485	0.8500
	RN w/o CAR	✓	×	0.9145	0.8176	0.8634	0.8706
	ERIN	✓	✓	**0.9690**	0.8333	**0.8961**	**0.9033**

seen in Table 3, there is a margin between precision and recall in some cases of those methods. For instance, when the Shifu is set to the few-shot type, the difference between precision and recall achieves 12.2% and 7.6% respectively on RBRN and FC-Net methods. On the other hand, ERIN is able to achieve high precision and recall across different experimental settings simultaneously. One possible reason to explain this phenomenon is that ERIN is able to select more informative samples through the prototype rectification module, allowing the deep model to fully utilize the traffic samples with limited numbers.

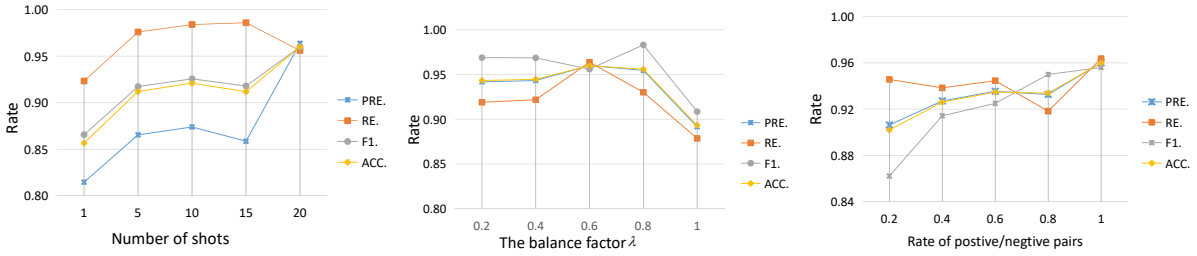
Results on Few-shot Traffic Characterization:1. Overall, ERIN can identify few-shot characterization types with a limited number and outperforms the other SOTA baseline methods. As can be seen in Figure 3, ERIN performs well in terms of accuracy, beating the FC-Net and RBRN baselines in all experimental settings. In particular, ERIN 3% lose than TF on accuracy when P2p is the few-shot characterization type. However, TF is hard to simultaneously perform well on both precision and recall. Moreover, results show that our ERIN achieves the best F1-score on all experiments with different settings, demonstrating the superiority of our ERIN on few-shot traffic characterization.

2. Whether the experimental setting changes or not, ERIN has a more stable performance than other baselines. Compared to RBRN and FC-Net, our methods have a smaller standard deviation in terms of F1-Score in the experiments with the same setting, which is reflected in the length of the box in Figure 3(c). In addition, ERIN is less insensitive to the type of few-shot characterization. In comparison, the performance of TF may dramatically decrease when the few-shot characterization type is changed. This is mainly because that ERIN could learn universal prior knowledge through applying the contrastive-aware representation module, improving the generalization of ERIN for few-shot traffic types.

5.3 Ablation Analysis (RQ2)

In this section, we conduct ablation experiments to evaluate the effectiveness of the contrastive-aware representation module and prototype rectification module. We consider RN, RN w/o CAR and RN w/o PR three variants. RN is the original relation network proposed in study[25]. RN w/o CAR adds the contrastive-aware representation module to RN, and the prototype rectification module is added in RN w/o PR.

Effectiveness of Prototype Rectification: As shown in Table 4, we note that the prototype rectification module improves the performance of RN on recall, F1-score and accuracy three metrics by



(a) Impact of the number of support samples (b) Impact of the balance factor (c) Impact of the contrastive pairs

Figure 4: The performances of ERIN as hyperparameters vary.

0.7% to 4.4% on the USTC-TFC dataset. A similar conclusion can be drawn for the UNB-ISCX dataset. The information carried by each traffic example in the support set is different. Prototype rectification module selects the more informative traffic samples and boosts the weights of those samples when generating the prototype for each class in the support set. As we expected, the results in Table 4 demonstrate the effectiveness of the prototype rectification module.

Effectiveness of Contrastive-aware Representation Module: As shown in Table 4, the contrastive-aware representation module improves the performance of RN on the USTC-TFC dataset with F1-score of 4.7% and accuracy of 4.1%, respectively. As for the UNB-ISCX dataset, the contrastive-aware representation module improves all metrics by 2.3% to 9.8%. Compared to RN, the contrastive-aware representation module improves the discriminative ability of RN w/o CAR via enhancing the similarity between the learned features belonging to the same type. ERIN generated a more compact feature space where the learned features belonging to the same type have a short distance, allowing the deep model to generalize better for unseen traffic types.

Moreover, full ERIN achieves the best performance on two benchmarks in Table 4, demonstrating that the combination of the contrastive aware representation module and prototype rectification module improves the performance of RN to the greatest extent.

5.4 Sensitivity Analysis (RQ3)

In this part, we investigate the imparts of three parameters, including the number of samples in the support set, the balance factor λ and the number of contrastive pairs. The experimental results are reported in Figure 4.

Impact of the Number of Samples in the Support Set: We test the performance of different $k \in [1, 5, 10, 15, 20]$ which indicates the number of samples per class in the support set. Figure 4(a) shows that as we increase more traffic samples in the support set, the performance of ERIN will be better. This phenomenon is consistent with our expectation because more traffic samples will provide more abundant information when performing few-shot encrypted traffic classification tasks. On the other hand, the few-shot traffic classification model is less sensitive than the conventional traffic classification model. For example, when $k \in [5, 15]$, the improvement of ERIN is not obvious even if the k is doubled.

Impact of Balance Factor λ: λ is a hyperparameter to balance the proportion of contrastive loss and MSE loss in total loss. The larger λ indicates that the proportion of contrastive loss is higher. As shown in Figure 4(b), we can see that ERIN achieves the best performance in terms of F1-score and accuracy when $\lambda = 0.6$. It is also notable that the performance of ERIN dramatically decreases when the $\lambda = 1.0$. This phenomenon indicates that fully replacing the MSE loss with contrastive loss will damage the performance of ERIN, and the combination of them is necessary for the few-shot encrypted traffic classification.

Impact of the Number of Contrastive Pairs: We denote the p as the ratio of contrastive pairs participating in model training to the total number of contrastive pairs. As shown in Figure 4(c), we can see that as more contrastive pairs are involved in the meta-training phase, the performance of ERIN will be better. In addition, we found that ERIN reaches nearly 93% on F1-score even when $p = 0.4$, which is only 2% worse than using all contrastive pairs ($p = 1.0$), indicating that even portion of contrastive pairs contains sufficiently rich information for ERIN optimization.

6 CONCLUSION

In this paper, we proposed a novel method namely ERIN for the few-shot encrypted traffic classification. ERIN contains traffic featurization module, contrastive-aware representation module, prototype rectification module and deep similarity kernel module. Traffic featurization module generates the features for all traffic inputs. Then, the contrastive-aware representation module enhances the similarity of the learned features belonging to the same type via integrating contrastive loss into the meta-learning framework, allowing ERIN to generalize better for few-shot traffic types. Moreover, we apply prototype rectification to generate a more comprehensive prototype for each class in the support set, improving the performance of ERIN. At last, the deep similarity kernel module will give the recognization results by computing the similarity of query sample and prototypes in the support set. Furthermore, we evaluate the effectiveness of ERIN on two real-world public encrypted traffic datasets, and the experimental results demonstrate the effectiveness of ERIN on few-shot encrypted traffic classification tasks.

REFERENCES

[1] Blake Anderson and David McGrew. 2017. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*. 1723–1732.

[2] Jin Cheng, Yulei Wu, E Yuepeng, Junling You, Tong Li, Hui Li, and Jingguo Ge. 2021. MATEC: A lightweight neural network for online encrypted traffic

classification. *Computer Networks* 199 (2021), 108472.
[3] Tongtong Feng, Qi Qi, Jingyu Wang, and Jianxin Liao. 2021. Few-Shot Class-Adaptive Anomaly Detection with Model-Agnostic Meta-Learning. In *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–9.
[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.
[5] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. 2020. Meta-Learning in Neural Networks: A Survey. *arXiv* (2020).
[6] Mike Huisman, Jan N Van Rijn, and Aske Plaat. 2021. A survey of deep meta-learning. *Artificial Intelligence Review* 54, 6 (2021), 4483–4541.
[7] Zijian Jia, Yepeng Yao, Qiuyun Wang, Xuren Wang, Baoxu Liu, and Zhengwei Jiang. 2021. Trojan Traffic Detection Based on Meta-learning. In *International Conference on Computational Science*. Springer, 167–180.
[8] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M Bronstein. 2019. Repmet: Representative-based metric learning for classification and few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5197–5206.
[9] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille, 0.
[10] Chang Liu, Zigang Cao, Gang Xiong, Gaopeng Gou, Siu-Ming Yiu, and Longtao He. 2018. Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.
[11] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li. 2019. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*.
[12] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.
[13] Kelong Mao, Xi Xiao, Guangwu Hu, Xiapu Luo, Bin Zhang, and Shutao Xia. 2021. Byte-Label Joint Attention Learning for Packet-grained Network Traffic Classification. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.
[14] Matiur Rahman Minar and Jibon Naher. 2018. Recent advances in deep learning: An overview. *arXiv preprint arXiv:1807.08169* (2018).
[15] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. 2019. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2239–2247.
[16] Pescape, A., Dainotti, Claffy, and C. K. 2012. Issues and future directions in traffic classification. *IEEE Network: The Magazine of Computer Communications* 26, 1 (2012), 35–40.
[17] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676* (2018).
[18] Shahbaz Rezaei and Xin Liu. 2019. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine* 57, 5 (2019), 76–81.
[19] Candong Rong, Gaopeng Gou, Chengshang Hou, Zhen Li, Gang Xiong, and Li Guo. 2021. UMVD-FSL: Unseen Malware Variants Detection Using Few-Shot Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[20] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*. PMLR, 1842–1850.
[21] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065* (2016).
[22] Muhammad Shafiq, Xiangzhan Yu, Asif Ali Laghari, Lu Yao, Nabin Kumar Karn, and Foudil Abdessamia. 2016. Network traffic classification techniques and comparative analysis using machine learning algorithms. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2451–2455.
[23] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew K Wright. 2019. Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with N-shot Learning. (2019).
[24] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4080–4090.
[25] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1199–1208.
[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems* 29 (2016), 3630–3638.
[27] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016).
[28] Xin Wang, Shuhui Chen, and Jinshu Su. 2020. App-Net: a hybrid neural network for encrypted mobile traffic classification. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 424–429.
[29] W. Wei, Z. Ming, J. Wang, X. Zeng, and Z. Yang. 2017. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*.
[30] W. Wei, Z. Ming, X. Zeng, X. Ye, and Y. Sheng. 2017. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*.
[31] W. Wei, Y. Sheng, J. Wang, X. Zeng, and Z. Ming. 2018. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* 6, 99 (2018), 1792–1806.
[32] Guorui Xie, Qing Li, and Yong Jiang. 2021. Self-attentive deep learning method for online traffic classification and its interpretability. *Computer Networks* 196 (2021), 108267.
[33] C. Xu, Shen J, and X. Du. 2020. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Transactions on Information Forensics and Security* PP, 99 (2020), 1–1.
[34] Tianpeng Ye, Gaolei Li, Ijaz Ahmad, Chaofeng Zhang, Xiang Lin, and Jianhua Li. 2021. FLAG: Few-shot Latent Dirichlet Generative Learning for Semantic-aware Traffic Detection. *IEEE Transactions on Network and Service Management* (2021).
[35] Zijian Zhao, Yingxu Lai, Yipeng Wang, Wenxu Jia, and Huijie He. 2021. A Few-shot Learning Based Approach to IoT Traffic Classification. *IEEE Communications Letters* (2021).
[36] Wenbo Zheng, Chao Gou, Lan Yan, and Shaocong Mo. 2020. Learning to Classify: A Flow-Based Relation Network for Encrypted Traffic Classification. In *WWW '20: The Web Conference 2020*.