

RESEARCH ARTICLE

SPN: A Method of Few-Shot Traffic Classification With Out-of-Distribution Detection Based on Siamese Prototypical Network

GONGXUN MIAO^{1,2}, GUOHUA WU¹, ZHEN ZHANG¹, YONGJIE TONG², AND BING LU²¹School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China²Zhongfu Information Company Ltd., Jinan 250100, China

Corresponding author: Gongxun Miao (miaogx@hdu.edu.cn)


This work was supported in part by the “Pioneer” and “Leading Goose” Research and Development Program of Zhejiang under Grant 2023C03203, Grant 2023C03180, and Grant 2022C03174.

ABSTRACT Traffic classification has always been one of the important research directions in the field of cyber security. Achieving rapid traffic classification and detecting unknown traffic are critical for preventing network attacks, malicious software, transaction fraud, and other types of cyber security threats. However, most existing models are based on large-scale data and are unable to quickly learn and recognize unknown traffic. Some methods based on few-shot learning solve the problem of rapidly learning new types of traffic, but they cannot detect out-of-distribution samples. Based on this, this paper proposes a few-shot traffic multi-classification method that supports out-of-distribution detection, named SPN. It improves the performance by integrating twin networks into the meta-learning framework based on the idea of metric learning, and introduces margin loss to ensure detection performance. We conduct two types of experiments, and compare them with the relevant baseline methods. The results show that SPN has excellent performance in implementing few-shot multi-classification and out-of-distribution detection, and performs well in intrusion detection.

INDEX TERMS Few-shot, intrusion detection, network traffic classification, out-of-distribution, SPN.

I. INTRODUCTION

Network traffic classification has always been a focus of both the cyber security research field and Internet Regulatory Agency. It aims to classify and identify the traffic transmitted over the internet according to different standards. With the extensive application of the internet in areas such as malicious traffic detection, network monitoring, and traffic management, the scale of network traffic data is growing exponentially. This includes various types of data such as video, audio, text, and application data. According to relevant research reports [1], as of the end of 2022, the number of independent mobile users worldwide reached 5.4 billion, with mobile internet users accounting for 4.4 billion. With the surge in traffic, its changes are also accelerating, involving

The associate editor coordinating the review of this manuscript and approving it for publication was Mueen Uddin .

different types of protocols, encryption, obfuscation, and hiding techniques, which lead to the diversity and complexity of the traffic. The development of emerging internet technologies (such as 5G and IoT) brings new requirements for traffic classification. At present, traffic classification faces the following challenges: 1. Difficulty in obtaining large amounts of labeled data; 2. Limited transferability and adaptability of models in constantly emerging new data and scenarios; 3. Limited capability in identifying newly emerging unknown types; 4. Imbalanced data categories and small inter-class differences.

To address the aforementioned challenges, models need to have the ability to learn, transfer, and identify unknown types of samples under few-shot conditions. However, traditional machine learning methods based on statistical features and deep learning methods driven by big data struggle with learning under few-shot conditions. Moreover, existing

few-shot learning approaches based on transfer learning [2] and data augmentation [3], [4] have certain limitations in adapting to new scenarios and dealing with imbalanced data categories, small inter-class differences and the complexity of newly emerging unknown types. Meta-learning [5], [6] has advantages in few-shot traffic classification, such as strong generalization capabilities, low resource overhead, and ease of scenario transfer [7], [8], [9], [10], but these methods lack the ability to detect unknown new types of data. Detection of such unknown new types of data is termed Out-of-Distribution (OOD) detection. OOD detection can provide early warnings and countermeasures against unknown network threats, assist in promptly intercepting malicious traffic, and also reduce the risks of false positives and false negatives. Currently, most OOD detection research is based on large-scale data conditions, and its application in few-shot conditions (such as in the field of natural language processing [11]) is still in its early stages, lacking relatively mature algorithms.

Therefore, in order to overcome these problems in the field of traffic classification mentioned above, and achieve rapid classification and detection of new types of traffic data and unknown types under few-shot conditions, we conduct the following work,

We propose a deep learning framework based on meta-learning, named SPN. This framework accomplishes rapid classification of new types of traffic and detection of unknown traffic simultaneously, demonstrating high performance.

SPN deeply integrates the structure of two siamese twins and the prototype computation module. By utilizing the concept of metric learning, we design a novel metric that endows the network with multi-classification capability. Additionally, it possesses the ability to distinguish between inter-class similar samples and the ability to quickly learn from few samples.

The framework is designed with two novel margin losses that reduce the distance between in-distribution(ID) samples and their corresponding prototypes, and increase the distance between OOD samples and ID samples. This enables the model to achieve better OOD detection performance.

II. RELATED WORKS

In this section, we review existing works on traffic classification, few-shot learning, and OOD detection.

A. TRAFFIC CLASSIFICATION

1) TRADITIONAL MACHINE LEARNING-BASED METHODS

These methods are based on statistical features in network traffic, followed by the use of traditional machine learning algorithms for classification, such as K-Nearest Neighbors(KNN) [12], Support Vector Machines(SVM) [13], Random Forest(RF) [14], eXtreme Gradient Boosting(XGBoost) [15], etc. These methods are straightforward and easy to understand, but the feature extraction requires manual

selection, which can't fully capture the complex patterns and features in network traffic. They no longer meet the requirements for the scale of data today.

2) DEEP LEARNING-BASED METHODS

These methods use deep neural networks to learn feature representations from raw network traffic data, avoiding the drawbacks of manual feature extraction. Since around 2015, many researchers have conducted a lot of researches and attempts in this area. The used models include Convolutional Neural Networks(CNN), Restricted Boltzmann Machines(RBM), Long Short-Term Memory networks(LSTM), and so on. Andresini et al. [16] proposed a model named CLAIRE. The model derives a two-dimensional image representation of network flows by performing a combination of nearest neighbor search and clustering processes, and then uses a convolutional neural network to train the model. Aldwairi et al. [17] proposed an RBM-based method. RBM can be trained without prior information, automatically discovering latent features in the data, and achieving network traffic intrusion detection. References [18], [19], and [20] proposed using combined features constructed by combined methods to improve the accuracy of network intrusion detection. These features include statistical features, spatial features extracted by CNN, and temporal features extracted by Recurrent Neural Networks. Although these methods have their advantages in traffic classification tasks, they all require large amounts of labeled data for training, and they also have high label dependencies and computational complexities. This makes it challenging for them to meet the demands of new scenarios or tasks.

B. FEW-SHOT LEARNING

Few-shot learning aims to perform classification or learning tasks using a few labeled samples. The model needs to learn from just a few or dozens of labeled samples and be able to generalize to new classes that have no or few labeled samples. Few-shot learning is widely used in image classification, and the proposed models include Siamese Networks(SiameseNet) [21], MatchingNet [22], and ProtoNet [23]. Unlike SiameseNet, which calculate the similarity between any two images, ProtoNet map each category of images to a prototype, and then classifying images based on the distribution generated by their distances to the prototypes. Xu et al. [24] were the first to apply meta-learning to network intrusion detection. They took raw traffic bytes as input and achieved few-shot traffic classification by training on meta-tasks. Meta-learning models can quickly adjust through a small amount of new data to adapt to the new situation. It has advantages in quick learning and adaptation to new tasks and scenarios, such as strong generalization ability, low resource overhead, and easy scene transfer. Yang et al. [25] proposed an improved traffic classification model, FS-IDS, which improved the model's performance in few-shot classification by integrating raw traffic and traffic

statistical features. However, FS-IDS is only applicable for detecting specific malicious samples and is unable to detect unknown attacks. Shi et al. [26] first introduced the Learned to Forget(L2F) mechanism into the Model Agnostic Meta Learning(MAML) [27]. L2F can dynamically control task conflicts during MAML initialization, thus improving the model's convergence speed. Iliyasu et al. [28] proposed a discriminative representation learning method based on supervised auto-encoders to solve the few-shot problem in network intrusion detection. This method requires a large number of malicious samples as training support, and it has a high computational cost. Li et al. [29] proposed an adversarial unsupervised domain-adaptive regularization and an improved Cascade R-CNN to more effectively detect Internet of Things(IoT) attacks. This method improves the model's accuracy and robustness, enabling the detection of new attack samples. But, it requires a significant amount of computational resources and time to train the model, and it is susceptible to the influence of noisy data. Shirekar and Jamali-Rad in [30] proposed Class-Cognizant Contrastive Learning (C³LR), a method that introduces class-level cognition and considers class-level global structure by modifying contrastive loss, thereby improving the performance of few-shot classification. Nonetheless, the performance of this method is highly dependent on the quality and effectiveness of the pre-training phase. For different datasets and tasks, various pre-training strategies and parameter settings are required, resulting in poor transferability of the model. Yang et al. [31] proposed a Multi-task Representation Enhanced Meta-learning model(MetaMRE) to solve the multi-classification task of encrypted traffic. MetaMRE enhances traffic representation differences through the flow discrepancy enhancement module, enabling it to handle version updates and cross-domain issues in encrypted traffic classification well. The aforementioned methods all achieve decent results on few-shot multi-classification tasks, but they are only confined to ID sample classification, and cannot detect OOD samples.

C. OOD DETECTION

Unlike few-shot learning that typically relies on a small amount of labeled data, "OOD detection" is a technique used to determine whether a model has encountered types of data that it has not seen during the training process. In real-time scenarios, OOD detection is of significant importance. For example, in the field of network security, new types of attacks or anomalous behavior may not be included in the model's training data, but they need to be immediately identified and addressed. In this case, the model needs to be able to recognize data distributions that are different from the training data or unseen classes during training, in order to avoid making erroneous predictions on these samples. Hendrycks and Gimpel in [32] proposed a method of Maximum Softmax Probability (MSP), which utilizes the probabilities from the softmax distribution to detect

TABLE 1. Notation.

Notation	Description
N	The num of samples
M	The num of labels
B	The num of tasks
\mathcal{D}	Dataset
\mathcal{D}_S	Support set
\mathcal{D}_Q	Query set
\mathcal{D}_{train}	Meta-training set
\mathcal{D}_{valid}	Meta-validation set
\mathcal{D}_{test}	Meta-testing set
T_i	The i -th task in task set T
$g(\mathbf{x})$	The scoring function of \mathbf{x} for deviation from ID
Z	The number of packages selected for each session
W	The number of bytes captured in each packet
\mathbf{c}_k	The k -th prototype in an episode
S_k	An set of support samples for the k -th class
\mathcal{S}	The set of sample types in the current episode
$d(\bullet, \bullet)$	Distance function between two vectors
\mathbf{h}_L	The L -th layer in a twin of siamese net
$p_\phi(k \mathbf{x}_q)$	The distribution probability of the query vector.
\mathcal{L}_1	The log-likelihood loss
\mathcal{L}_{in}	The margin loss of ID samples
\mathcal{L}_{out}	The margin loss of OOD sample

samples that are misclassified or fall outside the distribution range in the neural network. DeVries and Taylor in [33] added a "confidence estimation branch" to the prediction model and predicted whether the current sample is an OOD sample through the confidence indicator output by the model. Chen et al. [34] proposed the ALOE method to solve the impact of adversarial samples. This method makes the model more robust to input disturbances by introducing an adversarial loss function during training. Wei et al. [35] proposed a very concise and effective LogitNorm loss, which solves the overfitting of the neural network while improving the model's correction ability. These methods have proven to be promising for OOD detection on large-scale data setting, but they are difficult to apply to scenarios with few-shot datasets.

To address the related issues in the field of traffic classification, particularly in few-shot multi-classification and OOD detection, a few-shot traffic classification method based on the meta-learning framework is proposed in this paper. This method possesses the simple and efficient transfer learning ability for new types of samples, the ability to discern subtle features inherent and the ability to detect OOD.

III. PROBLEM DEFINITION

In this section, we introduce some concepts and definitions. The notations used in this article are listed in Table 1.

A. META-LEARNING FRAMEWORK FOR TRAFFIC CLASSIFICATION

Network traffic classification aims to train a classifier to perform the classification and identification of traffic samples. Traffic refers to the data packets or data streams transmitted in a internet that represent specific communication sessions, and can be expressed in the form of

$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathbb{R}^D$, $y_i \in \{1, 2, \dots, M\}$ respectively represent the samples and their corresponding labels. The dataset is divided into three parts, meta-training set $\mathcal{D}_{\text{train}}$, meta-validation set $\mathcal{D}_{\text{valid}}$ and meta-testing set $\mathcal{D}_{\text{test}}$. The meta-learning process consists of two stages, meta-training and meta-testing, each of which comprises multiple episodes. Each episode contains one or more tasks, which represent specific learning modes or tasks and are the basic unit of meta-learning. Each task has its own input data and corresponding targets. The meta-training process requires constructing a task set for training and evaluating the meta-learning model, which can be defined as $T = \{T_i\}_{i=1}^B$, where each task T_i is dynamically sampled from corresponding data. Each task further divides the data into support set \mathcal{D}_S and query set \mathcal{D}_Q , as shown in Algorithm 1.

B. OOD DETECTION

OOD samples refers to samples or data points that do not match the distribution of the model’s training set, i.e., data with a distribution that the model has never encountered before. OOD detection discriminates whether a sample is from ID. OOD detection aims to determine whether a sample comes from the ID or not, and specifically, it can be achieved by learning a scoring function $g(\mathbf{x})$ as Equation (1),

$$y = \begin{cases} in, & g(\mathbf{x}) < \lambda \\ out, & g(\mathbf{x}) \geq \lambda \end{cases} \quad (1)$$

where y represents the predicted label of sample \mathbf{x} . λ is the empirical threshold, and the specific form of $g(\mathbf{x})$ will be shown in Section V-D. In practical applications, if $g(\mathbf{x})$ is greater than lambda, the \mathbf{x} is classified as OOD; otherwise, it is categorized within the ID classes.

IV. THE FRAMEWORK

This section elaborates on the theoretical framework of SPN in detail, including the embedding module, prototype and metric calculation module, and loss function, with a specific structure shown in Figure 1. Section IV-A, IV-B, and IV-C introduce the calculation processes and relevant details of each module in the framework, while Section IV-D describes the workflow of SPN.

A. EMBEDDING MODULE

We first process the traffic data into 3D images (see Section V-B specific data processing) and divide the data into meta-tasks. Next, we use CNN technology to extract features and convert them into an embedding vector. In traffic data, there is a problem of high similarity between samples of different categories, which makes it difficult to distinguish them. To capture subtle differences between samples, we adopt the idea of siameseNet and use two twin networks to extract features from the support and query sets respectively. Taking one of the two siamese twins as an example, we design a 6-layer convolutional neural network to extract and process the sample’s feature, whose specific

structure is shown in Figure 2. The 6 layers of convolution include two types of convolutions. The first includes a 128-filter $2 \times 2 \times 2$ convolution, a BatchNorm3D layer, and a ReLU layer, which we call Conv3D Block I. The second includes a 128-filter $2 \times 2 \times 2$ convolution, a BatchNorm3D layer, a ReLU non-linear layer, and a 128-filter $2 \times 2 \times 2$ convolution with a stride of 2, which we call Conv3D Block II. The first four layers use Conv3D Block I, and the last two layers use Conv3D Block II. After 6 layers of convolution, a feature map is obtained, which is then flattened to obtain a feature vector $\mathbf{h}_L(\mathbf{x})$. Next, the feature vector is fed into a fully connected layer, and a sigmoid operation is performed to obtain the embedding vector $f_\phi(\mathbf{x})$, where f_ϕ represents the sample’s embedding model and ϕ represents the embedding model parameters.

B. PROTOTYPE AND METRIC CALCULATION MODULE

Compared with many meta-learning frameworks such as Relation Networks [36] and MatchingNet, ProtoNet have the advantages of simple structure, strong performance, and strong adaptability. Its core idea is to use a vector, also called a prototype, as a representation of a class. In the support set, the prototype corresponding to each class is the mean vector of all embedding vectors of the samples in the corresponding class set,

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i). \quad (2)$$

\mathbf{c}_k is the prototype of the k -th class sample in the current episode’s support set, and S_k is the set of samples corresponding to the k -th class in the support set. Next, we use Equation (2) to calculate the prototype of each class in the support set of the meta-task. Based on the idea of metric learning, we design a distance measure in a latent space that effectively captures the differences between samples. The distance d between the q -th sample in the query set and the k -th prototype in the support set can be formalized as Equation (3),

$$d(\mathbf{x}_q, \mathbf{c}_k) = \text{ReLU} \left(\sum_{j=1}^{D'} \alpha_j |f_\phi^{(j)}(\mathbf{x}_q) - \mathbf{c}_k^{(j)}| \right). \quad (3)$$

$f_\phi^{(j)}$ represents the j -th component of the vector, D' is the dimensionality of the hidden vector, α_j is a parameter learned by the model during training, which measures the importance of different elements in the vector for distance calculation. After obtaining the distances between a query example and each prototype in the support set, we calculate the softmax probability distribution for the query vector \mathbf{x}_q as Equation (4),

$$p_\phi(y = k | \mathbf{x}_q) = \frac{\exp(-d(f_\phi(\mathbf{x}_q), \mathbf{c}_k))}{\sum_{k'=1}^{|S|} \exp(-d(f_\phi(\mathbf{x}_q), \mathbf{c}_{k'}))}. \quad (4)$$

S represents the set of sample types in the current episode, $d(\bullet, \bullet)$ represents the distance function between two vectors.

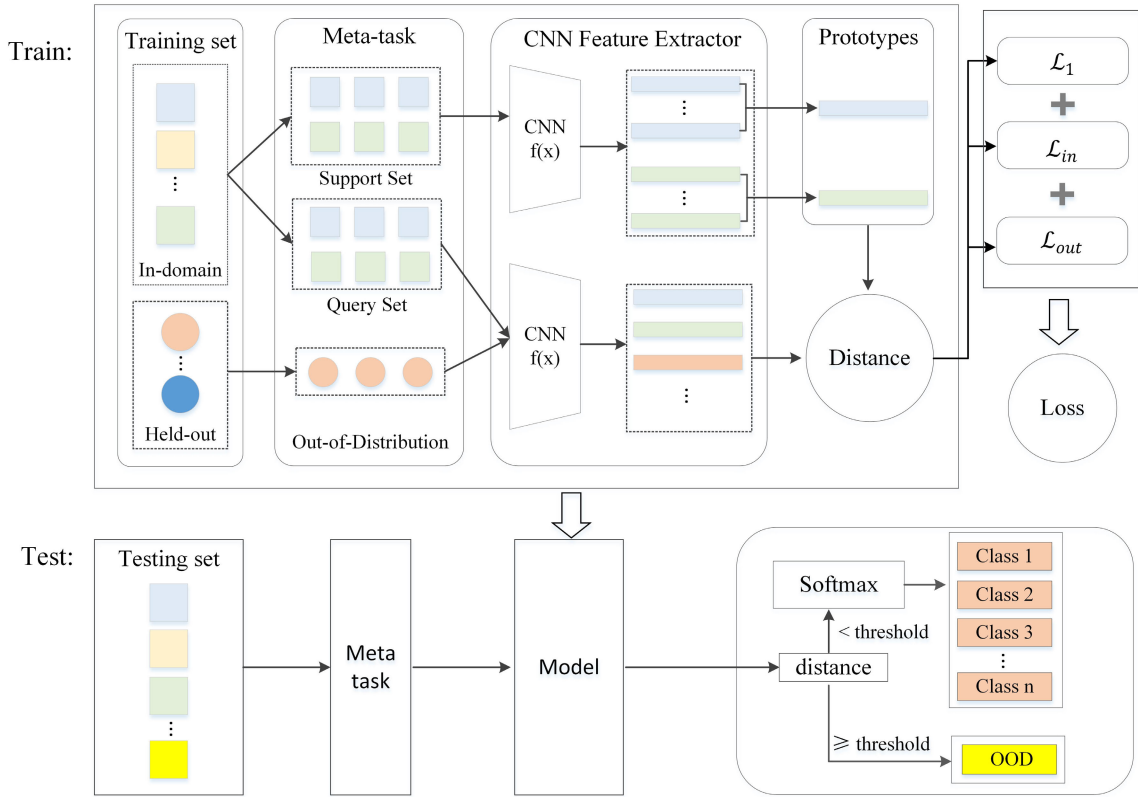


FIGURE 1. The overview of SPN framework.

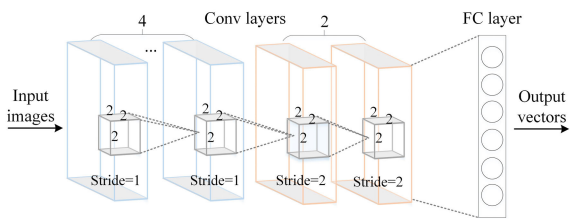


FIGURE 2. The structure of convolutional neural networks (one twin).

C. LOSS FUNCTION

To support OOD detection in few-shot traffic classification, we design a margin loss function for OOD based on the existing framework, which enables the model to perceive OOD during training. To improve the OOD discrimination ability of the model, the loss function considers two aspects, the naive classification loss and the margin losses. According to literature [23], we use negative log-likelihood as the naive classification loss,

$$\mathcal{L}_1 = -\log(p_\phi(y = k | \mathbf{x}_q)). \quad (5)$$

Besides, we use the two maximum margin losses to reduce the distance between ID samples and their corresponding prototype, while increase the distance between OOD samples and all prototypes. The margin losses is defined by

Equation (6)-(7),

$$\mathcal{L}_{in} = \max(d(\mathbf{x}_{q_{in}}, \mathbf{c}_k) - m_{in}, 0). \quad (6)$$

$$\mathcal{L}_{out} = \max\left(m_{out} - \min_k(d(\mathbf{x}_{q_{out}}, \mathbf{c}_k)), 0\right). \quad (7)$$

$\mathbf{x}_{q_{in}}$ and $\mathbf{x}_{q_{out}}$ represent the ID samples and OOD samples in the query set respectively. m_{in} and m_{out} are the two margin hyperparameters. In Equation (6), $d(\mathbf{x}_{q_{in}}, \mathbf{c}_k)$ represents the distance measure between the k -th class sample in the query set and the prototype of the k -th class in the support set. \mathcal{L}_{in} will encourage the samples from the same class to concentrate around their corresponding prototypes, thereby reducing the distance between samples and their prototypes within the same class. In Equation (7), $d(\mathbf{x}_{q_{out}}, \mathbf{c}_k)$ represents the distance between an OOD sample and each prototype in the support set. The right-hand side of Equation (7) selects the distance between the OOD sample and its nearest prototype, and increase the distance. The final loss function can be written as Equation (8),

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_{in} + \mathcal{L}_{out}. \quad (8)$$

The pseudocode for loss function computation in training episodes is shown in Algorithm 1. Where $RandomSample(S, n_1)$ denotes a set consisted of n_1 elements sampled randomly and uniformly from set S . $Split(S, n_2)$ denotes an operator that splits the set S into two sets, the

first set with n_2 elements and the second one with $|S| - n_2$ elements. V_{ID} denotes a set of K categories extracted from the category set of \mathcal{D}_{train} , where $K < M_{train}$. \mathcal{D}_{V_k} represents the set of all samples in \mathcal{D}_{train} where $y_i = k$. \mathcal{D}_k is a subset extracted from \mathcal{D}_{V_k} . \mathcal{D}_{S_k} and \mathcal{D}_{Q_k} are the support set and query set obtained by splitting \mathcal{D}_k , with sample sizes of N_S and N_Q respectively. V_{OOD} denotes the set of OOD categories extracted from \mathcal{D}_{train} . $\mathcal{D}_{V_{OOD}}$ represents the set of all samples in which $y_i \in V_{OOD}$. \mathcal{D}_{OOD} is a subset extracted from $\mathcal{D}_{V_{OOD}}$.

Algorithm 1 Training Episode Loss Computation for SPN

Input:

Meta training set $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1}^{M_{train}}$,
 where $y_i \in \{j\}_{j=1}^{M_{train}}$, $M_{train} > 2$.

Output:

The loss \mathcal{L} for a randomly generated training episode.

```

1:  $V_{ID} \leftarrow \text{RandomSample}(\{j\}_{j=1}^{M_{train}}, K)$ .
2: For  $k$  in  $\{1, 2, \dots, K\}$  Do
3:    $\mathcal{D}_k \leftarrow \text{RandomSample}(\mathcal{D}_{V_k}, N_S + N_Q)$ ;
4:    $\mathcal{D}_{S_k}, \mathcal{D}_{Q_k} \leftarrow \text{Split}(\mathcal{D}_k, N_S)$ ;
5:   Calculate  $\mathbf{c}_k$  by using Eqn (2)
6: End For
7:  $V_{OOD} \leftarrow \text{RandomSample}(\{j\}_{j=1}^{M_{train}} \setminus V_{ID}, 1)$ 
8:  $\mathcal{D}_{OOD} \leftarrow \text{RandomSample}(\mathcal{D}_{V_{OOD}}, N_{OOD})$ 
9:  $\mathcal{L} \leftarrow 0$ 
10: For  $k$  in  $\{1, 2, \dots, K\}$  Do
11:   For  $(x_q, y_q)$  in  $\mathcal{D}_{Q_k}$  Do
12:     Calculate  $\mathcal{L}_1$  and  $\mathcal{L}_{in}$  by using Eqn (5)-(6);
13:      $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{KN_Q}(\mathcal{L}_1 + \mathcal{L}_{in})$ ;
14:   End For
15: End For
16: For  $(x_q, y_q)$  in  $\mathcal{D}_{OOD}$  Do
17:   Calculate  $\mathcal{L}_{out}$  by using Eqn(7) on  $\mathcal{D}_{OOD}$ ;
18:    $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{N_{OOD}}\mathcal{L}_{out}$ ;
19: End For

```

D. THE STRUCTURE AND FLOW OF SPN

As shown in Figure 1, the overall structure of SPN consists of two parts: the meta-training phase and the meta-testing phase. In the meta-training phase, a certain number of classes are first selected, and a certain number of samples are chosen for each class to construct the meta-training set. Then, meta-tasks are constructed in Algorithm 1. The data of a meta-task is divided into two parts, namely the support set and the query set. The two sets are input to the two twins of the Siamese network respectively. After extracting features separately, they are formed into embedding vectors. And then, the prototypes are computed for each class in the support set, which serve as representations of the classes. Subsequently, the distance metric is calculated according to Equation (3). The distance metric is used to compute the distance between each ID sample in the query set and its corresponding prototypes in the support set, which is then used to calculate the naive classification loss \mathcal{L}_1 and the

margin loss \mathcal{L}_{in} . The distances between OOD samples in the query set and each prototype are also computed to calculate the margin loss \mathcal{L}_{out} . Finally, the three losses are added together.

In the meta-testing phase, the process of constructing the meta-testing set is the same as that of the meta-training set, but the selected classes are different. Then, meta-tasks are constructed based on the meta-testing set. The data of the meta-task is input to the model, which outputs the distances between each sample in the query set and each prototype. If the distance between a sample and each prototype is greater than a certain threshold, the sample is classified as OOD; otherwise, the probability is calculated according to Equation (4), and the sample is classified to the corresponding class based on the maximum probability.

V. EXPERIMENT

A. DATASETS

To verify the performance of SPN, we implement experiments on three benchmark datasets, ISCX2012, CIC-IDS2017 and USTC-TFC2016. Among them, ISCX2012 and CIC-IDS2017 are widely used standard datasets in network security research, containing common types of network attacks and normal traffic seen in the real world. Malicious traffic of USTC-TFC2016 is collected from public websites in the real world. These three datasets can represent real-world traffic scenarios. The descriptions of the three datasets are as follows,

- ISCX2012 is collected from real-world environments by the Canadian Institute of Cyber Security, with a total of seven days of traffic. Malicious traffic types include Infiltrating the network from inside, HTTP denial of service, DDos using an IRC botnet, and Brute force SSH.
- CIC-IDS2017 is similar to traffic data in a real network environment, and it is composed of normal traffic and the latest common attack traffic. This dataset includes 5 days of network traffic, with attack types including DoS, Heartbleed, Web attacks, Penetration, Botnet, and DDos.
- USTC-TFC2016 is composed of 10 types of malware traffic and 10 types of normal software traffic [37], in which the malware traffic is collected by CTU researchers from the real network environment, and the normal software traffic is collected by the traffic simulation device IXIA BPS.

Table 2-4 lists the number of sessions in different attack categories in the three baseline datasets.

B. DATA PREPROCESSING

In this paper, the method described in literature [24] is adopted for feature extraction. Firstly, the raw traffic is divided into sessions. Secondly, we take the first Z packets of each session and the first W bytes of each packet, and convert them into a matrix of $Z \times W$, where W is the square

TABLE 2. Sample sizes of 5 categories of ISCX2012.

Attack Types	Sizes
Benign	84008
Infiltrating the network from inside	9739
HTTP denial of service	3354
DDos using an IRC botnet	27283
Brute force SSH	4956

TABLE 3. Sample sizes of 14 categories of CIC-IDS2017.

Attack Types	Sizes
Benign	131746
PortScan	158818
DDoS	45135
Bot	1235
FTP-Patator	3943
SSH-Patator	2957
DoS GoldenEye	14856
DoS Hulk	9313
Web Attack-XSS	625
DoS Slowhttptest	8
DoS slowloris	2
Heartbleed	1
Infiltration	6
Web Attack-Sql Injection	12

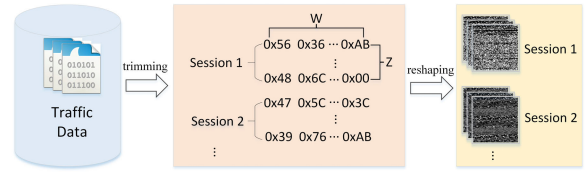
TABLE 4. Sample sizes of 20 categories of USTC-TFC2016.

Attack Types	Sizes	Attack Types	Sizes
Cridex	16386	BitTorrent	7517
Geodo	40947	Facetime	6000
Htbot	6367	FTP	101037
Miuref	13481	Gmail	8629
Neris	33791	MySQL	86089
Nsis-ay	6069	Outlook	7524
Shifu	9634	Skype	6321
Tinba	8504	SMB1	32661
Virut	33103	Weibo1	24953
Zeus	10970	WorldOfWarcraft	7883

number. If the number of sessions and the length of packets are insufficient, the zeros are added. Finally, we reshape the matrix into a $Z \times \sqrt{W} \times \sqrt{W}$ matrix. The details are shown in Figure 3.

C. EVALUATION METRICS

In order to comprehensively evaluate the performance of SPN, macro-precision(macro-P), macro-recall(macro-R),

**FIGURE 3.** The processing of traffic data.

macro-F1, accuracy rate(Acc), detection rate(DR) and F1 are selected as evaluation metrics, which are defined as Equation (9)-(14),

$$\text{macro-P} = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} P_i \quad (9)$$

$$\text{macro-R} = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} R_i \quad (10)$$

$$\text{macro-F1} = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \frac{2P_i \cdot R_i}{P_i + R_i} \quad (11)$$

$$\text{Acc} = \frac{1}{N_{test}} \sum_{i=1}^{M_{test}} TP_i \quad (12)$$

$$\text{DR} = \frac{TP_{Att}}{TP_{Att} + FN_{Att}} \quad (13)$$

$$\text{F1} = \frac{2P_{OOD} \cdot R_{OOD}}{P_{OOD} + R_{OOD}} \quad (14)$$

where $P_i = \frac{TP_i}{TP_i + FP_i}$ and $R_i = \frac{TP_i}{TP_i + FN_i}$ represent the precision and recall of the i -th class sample respectively. FP_i denotes the number of non- i -th class samples classified as the i -th class, FN_i represents the number of i -th class samples classified as the non- i -th class. N_{test} denotes the total number of samples in \mathcal{D}_{test} , and M_{test} is the total number of types in \mathcal{D}_{test} . TP_{Att} represents the number of attack samples correctly classified as attack, and FN_{Att} denotes the number of attack samples incorrectly classified as legal. The macro-P, macro-R, macro-F1, and Acc are used to evaluate the model's multi-classification performance. Acc and F1 are used to measure the model's OOD detection performance. Where P_{OOD} and R_{OOD} represent the accuracy and recall rate of OOD samples respectively. Acc and DR are used to measure the model's performance in attack traffic detection.

D. EXPERIMENTAL SETTINGS

To comprehensively evaluate the performance of the model, two types of experiments are designed in this paper, traffic multi-classification with OOD detection, and intrusion detection.

• Traffic Multi-Classification with OOD Detection.

USTC-TFC2016 and CIC-IDS2017 are selected as benchmark datasets for the experiment. For USTC-TFC2016, we select 10 classes for training, 5 classes for validation, and 5 classes for testing. From the 5 classes in the testing set, we choose one class as the OOD class. Due to the enormous number of possible combinations of all classes ($C_{20}^{10} \cdot C_{10}^5 \cdot C_5^1$), we randomly select 100 combinations. Each combination is experimented

once to obtain a set of metrics, and we take the average of the 100 sets of metrics. CIC-IDS2017 contains 14 classes, with 5 of them having sample sizes less than 12. Due to the extremely small data size, the statistical results of the metrics may not be reliable, so we select the other 9 classes as experimental data. Due to the small number of classes, the evaluation metrics for multi-class classification may not be sufficiently reliable. Therefore, in the experiment, we select the three classes with larger sample sizes (Benign, DDoS, and PortScan) and divide each of them into corresponding training, validation, and testing sets. Then, we select two classes from the remaining six classes for meta-training, two classes for meta-validation, and two classes for meta-testing. In this way, the training, validation, and testing sets each contain 5 classes. With this approach, there are still a considerable number of classes combinations ($C_6^2 \cdot C_4^2 \cdot C_5^1$), and we perform one experiment for each combination to obtain a set of metrics. We take the average of the 100 sets of metrics. We select 5 samples for each class, performing 5-shot 5-way tests. During testing stage, we select the distance to the nearest prototype as the scoring function, $g(\mathbf{x}) = \min_k(d(\mathbf{x}, \mathbf{c}_k))$. Since there is no relatively mature algorithm for implementing multi-classification and OOD detection simultaneously in the field of few-shot traffic classification, we divide the experimental results into two parts for comparison, 1) Comparison of multi-class classification performance (with relevant multi-classification algorithms as baselines, in Section V-E1). 2) Comparison of OOD detection performance (with relevant OOD detection algorithms as baselines, in Section V-E2).

- **Intrusion Detection.** Its purpose is to distinguish between normal traffic and attack traffic, which is essentially a binary classification task. Experiments are conducted on the ISCX2012 and CIC-IDS2017 datasets. As it is a binary classification task, the experiment aims to verify the performance of SPN in binary classification tasks compared with mature binary classification models. It does not involve OOD detection (This framework does not extract data as OOD in the process of generating meta-tasks, and does not consider marginal loss \mathcal{L}_{out}). Since literature [24] achieved traffic intrusion detection for the first time under the meta learning framework, the same data processing approach is adopted in this paper. The ISCX2012 dataset consists of 5 categories of data, including 1 category of normal traffic and 4 categories of attack traffic. During training, we select the normal traffic as one category and randomly combine three categories of attack traffic into another category, based on which the model performed binary classification. During testing, we select the normal traffic as one category and the remaining attack traffic as another category, resulting in a total of C_4^1 combinations. Each combination was evaluated to obtain a set of metrics,

TABLE 5. Comparison of multiple classification results between SPN and baseline methods.

Method	CIC-IDS2017			USTC-TFC2016		
	macro-F1	macro-R	macro-P	macro-F1	macro-R	macro-P
KNN ^[12]	0.6032	0.5998	0.6031	0.6532	0.6493	0.6501
RF ^[14]	0.9197	0.9196	0.9207	0.9396	0.9401	0.9393
XGBoost ^[15]	0.9045	0.9069	0.9118	0.9283	0.9290	0.9281
MatchingNet ^[22]	0.8914	0.8917	0.8921	0.9650	0.9659	0.9660
ProtoNet ^[23]	0.7622	0.7720	0.7719	0.6558	0.6840	0.7438
SiameseNet ^[21]	0.7387	0.7530	0.7513	0.7008	0.7133	0.7161
C ³ LR ^[30]	0.8886	0.8933	0.9032	0.9025	0.9022	0.9177
MetaMRE ^[31]	0.8942	0.9045	0.9017	0.9103	0.9146	0.9215
SPN	0.9248	0.9244	0.9378	0.9650	0.9652	0.9661

and we calculate the average of these metrics for 4 sets. For CIC-IDS2017, we follow the approach described in reference [24] and select 6 types of traffic, including normal traffic and 5 categories of attack traffic. During training, we randomly combine 3 categories of attack traffic with the normal traffic to form the meta-training set, while the remaining attack traffic and normal traffic formed the meta-testing set. This resulted in a total of C_5^2 possible combinations. Similarly, we take the average of all combination metrics as the final result. In this experiment, we select Acc and DR as the evaluation metrics for intrusion detection.

- **Setting of relevant parameters.** The parameters Z and W are 16 and 256 respectively. The threshold λ is 0.6. The dimension of embedding vector $f_\phi(x)$ is 1024, m_{in} and m_{out} are 0.2 and 0.8 respectively.

E. EXPERIMENTAL RESULTS

This section shows the comparison of traffic multi-classification results, OOD detection and intrusion detection.

1) RESULTS AND ANALYSIS OF TRAFFIC MULTI-CLASSIFICATION

Table 5 shows the multi-classification performance of the proposed method and 8 baseline methods on two datasets. Different from the baseline methods, SPN can achieve both few-shot multi-classification and OOD detection simultaneously, while the baseline methods only achieve few-shot multi-classification.

- **KNN:** The performance is poor on both datasets. The primary reason is that the network traffic data may contain many samples with highly similar features but belong to different classes, making it difficult for KNN to distinguish between them. Additionally, network traffic data usually have multiple feature dimensions. In high-dimensional spaces, distance calculations become more complex, and the “curse of dimensionality” may occur, leading to a decline in KNN’s performance and significant computational overhead.

- RF/XGBoost:** These two methods exhibit strong generalization capabilities and robustness against overfitting. They can effectively capture the complex nonlinear relationships that may exist among features in network traffic data and are also proficient at handling high-dimensional feature data. Therefore, they show stable and promising performance across both datasets. However, both methods require large amounts of labeled data and manual feature extraction for training, and are limited to classifying predefined types. They are not adaptable to classifying new types of samples or recognizing OOD samples, thus falling short in meeting the requirements of real-world network traffic classification scenarios.
- MatchingNet:** It achieves relatively excellent performance on three metrics across the two datasets, especially on USTC-TFC2016. The main reason may be that it uses a bidirectional LSTM mechanism to fuse the features of all samples in the support set when processing data in meta tasks, so that each sample's embedding vector has a global view. At the same time, the attention mechanism is used to make the classification of query examples more accurate. But it suffers from the issues of high resource consumption and slow computation speed.
- ProtoNet:** It performs poorly on both datasets. The main reason may be that it uses a simple method of taking the vector mean to obtain class features. This processing method may not be able to reflect the true characteristics of the traffic. But this method has the characteristics of small overhead and fast computation speed.
- SiameseNet:** It has poor performance on both datasets. It uses ordinary Siamese network, while the network's complexity is insufficient, making it difficult to capture all the relevant features from the input data. In addition, the model itself is suitable for binary classification but not suitable for multi-classification. These may be the reason for the poor performance of the model.
- C³LR:** It adopts a self-supervised pre-training mechanism and introduces a new loss term, which enables the model to obtain representations that are more effective and generalizable, with intra-class similarity and inter-class differences. This may be the main reason why C³LR achieves excellent results.
- MetaMRE:** It is based on the MAML framework, introduces a flow discrepancy enhancement module to enhance the representation difference of encrypted traffic, and fine-tunes the parameters according to a few number of labeled samples during the testing phase to adapt to the current task, which makes the model have excellent performance on both datasets.
- SPN:** The proposed method achieves the best results on the CIC-IDS2017 dataset. On the USTC-TFC2016 dataset, it performs comparably to MatchingNet overall and outperforms other methods significantly. In terms of feature extraction, SPN employs a uniquely designed

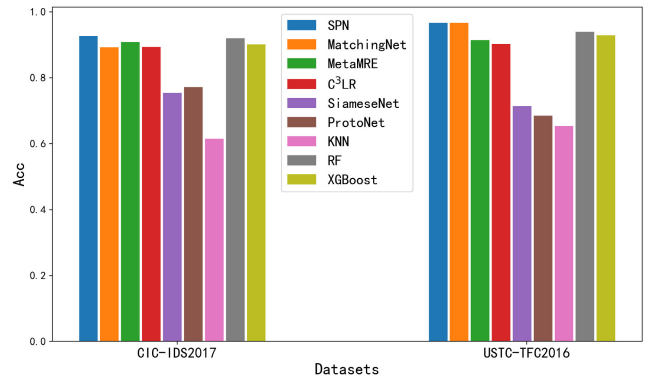


FIGURE 4. Acc comparison of SPN and 8 baselines on CIC-IDS2017 and USTC-TFC2016.

3D convolutional structure called Conv3D Block, which effectively extracts detailed features of traffic data while considering the temporal characteristics of packets within the session. This allows the model to obtain effective representations of traffic data. In terms of network structure, SPN integrates the core structures of the ProtoNet and SiameseNet, giving it the ability of strong discrimination and fast recognition. In addition, SPN introduces margin loss, enabling the model to identify OOD samples (see section V-E3 for detailed effects). These ensure the superior performance of the model in 3 indicators.

In addition, we also calculate another important indicator, Acc. Figure 4 shows the comparison of Acc between SPN and the baseline methods on two datasets. Compared with all baseline methods based on few-shot learning, SPN achieves the best results on CIC-IDS2017 and USTC-TFC2016, improving by 2% and 0.01% respectively compared to the second-best models (MetaMRE and MatchingNet). In all methods based on large-scale data, compared with RF (the second-best model), SPN achieves gains by 0.7% and 2.8% on CIC-IDS2017 and USTC-TFC2016 respectively. In summary, the results show that SPN has superior performance and stronger stability in traffic multi-classification tasks compared to all baseline methods on different indicators across different datasets. Since different sample sizes will have a certain impact on the results, we evaluate the Acc and macro-F1 indicators under different numbers, as shown in Figure 5. As the sample size increases from 1, 3, and 5, the values of Acc and macro-F1 gradually increase on the two datasets, and they tend to stabilize at 10 samples.

2) OOD DETECTION RESULTS ANALYSIS

SPN can also distinguish OOD samples while implementing few-shot multi-classification. To the best of our knowledge, in the field of traffic classification, SPN is the first few-shot multi-classification method that can support OOD detection. To verify the performance of SPN in OOD detection, we compare SPN with some OOD detection methods based on large-scale data. The table 6 shows the experimental

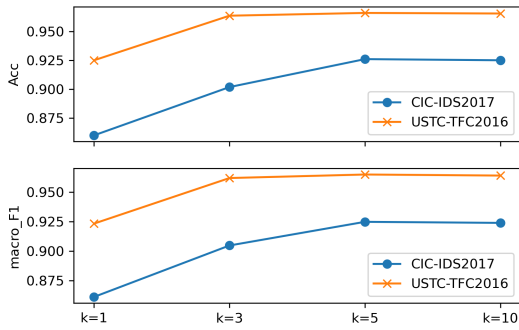


FIGURE 5. Acc and macro-F1 of different k-shots.

TABLE 6. The OOD results of the proposed method were compared with baseline method.

Method	CIC-IDS2017		USTC-TFC2016	
	Acc	F1	Acc	F1
MSP ^[32]	0.8785	0.9347	0.8791	0.9356
ALOE ^[34]	0.8462	0.9167	0.8780	0.9351
LogitNorm ^[35]	0.8760	0.9318	0.9954	0.9974
SPN	0.9512	0.9038	0.9917	0.9833

results of baseline methods and SPN in OOD detection. The result shows that SPN achieves the highest Acc on the CIC-IDS2017 dataset, which is 8.28% higher than MSP(the second-best model), but the F1 is relatively poor. On the USTC-TFC2016 dataset, SPN achieves excellent Acc and F1 scores, both exceeding 98%. All baseline methods have more than 18,000 training samples on IDS2017 and over 36,000 on USTC-TFC2016, while SPN only uses 5 labeled samples for each class. This shows that although SPN is not the best in some indicators, it still has excellent performance in OOD detection.

3) INTRUSION DETECTION RESULTS ANALYSIS

Intrusion detection has mostly been based on large-scale data, but in recent years, the development of intrusion detection methods based on few-shot samples has gradually emerged. In this section, we compare and analyze some intrusion detection methods based on large-scale samples and few-shot samples with SPN.

The comparison of SPN and existing methods on ISCX2012 are shown in Table 7. From the table, we can see that in Acc score, SPN has exceeded the EMD-based method, RBM-based method, RFA, FC-Net, and RFP-CNN based domain-adaptive method by 9.21%, 10.48%, 5.84%, 0.79%, and 1.71%, respectively. Although SPN doesn't surpass HAST-II, TR-IDS and Flow-based deep learning method, the three methods are all based on large-scale data setting. The TR-IDS with the smallest data requirement reaches over 30,000 samples, while the training samples required by SPN are far less than these methods. SPN achieves a

TABLE 7. Comparison of experimental results between the baseline and the proposed method on the ISCX2012 dataset.

Method	Sample Size	Acc(%)	DR(%)
EMD-based method (2015) ^[38]	8720	90.04	90.12
HAST-II (2017) ^[19]	915695	99.89	96.96
TR-IDS (2018) ^[18]	35357	99.13	99.26
RBM-based method (2018) ^[17]	82556	89.00	N/A
Flow-based deep learning method(2018) ^[20]	3320315	99.09	99.08
RFA (2018) ^[26]	500	92.90	89.60
RFA (2018) ^[26]	25	77.60	71.50
FC-Net(2020) ^[24]	5	97.56	98.78
RFP-CNN based domain-adaptive method(2022) ^[29]	5/10	96.68	N/A
SPN	5	98.33	97.12

TABLE 8. Comparison of experimental results between the baseline and the proposed method on the CIC-IDS2017 dataset.

Method	Sample Size	Acc(%)	DR(%)
Flow-based deep learning method(2018) ^[20]	1028007	98.87	98.83
GA-based adaptive method (2018) ^[39]	760056	N/A	92.85
IFSE-AD (2018) ^[40]	24357	97.30	N/A
CLAIRE(2021) ^[16]	30000	98.01	92.55
FC-Net(2020) ^[24]	5	94.33	99.17
RFP-CNN based domain-adaptive method(2022) ^[29]	5/10	94.98	N/A
FS-IDS(2022) ^[25]	5	97.51	99
Supervised Autoencoder ^[28] (2022)	10/20	92.2	89.82
MAML with L2F Method(2023) ^[27]	10	94.66	N/A
SPN	5	99.63	99.62

good performance on the DR metric, with model results consistently above 97%. Although it is not the best, it has also achieved good performance.

The comparison of SPN and existing methods on CIC-IDS2017 are shown in Table 8. It shows that SPN has achieved the best results in terms of Acc and DR. Meanwhile SPN has surpassed the second-best models(Flow-based deep learning method and FC-Net) by 0.77% and 0.8% on the two indicators respectively. Especially in comparison with methods based on few-shot, SPN has surpassed FC-Net, RFP-CNN based domain-adaptive method, FS-IDS, Supervised Autoencoder, MAML with L2F Method by 5.62%, 4.9%, 2.17%, 8.06%, and 5.25% on Acc, respectively. It has surpassed FC-Net, Supervised Autoencoder by 0.45%, 10.91% on DR respectively.

From the results on the ISCX2012, it can be seen that among all few-shot methods, SPN has achieved the best performance on Acc, which is 0.79% higher than the second-best few-shot model. Compared to large-scale dataset based methods, the Acc is only 1.56% lower than the best model, but the data required for training is far less than the best model. The DR indicator is not the best, but it is higher than 97%. On CIC-IDS2017, Acc and DR have exceeded all

baseline methods, which are 0.77% and 0.45% higher than the second-best model, respectively. In summary, on the two evaluation metrics, SPN has shown excellent performance on both datasets.

VI. CONCLUSION

Implementing traffic classification and intrusion detection is of significant importance for detecting and preventing numerous issues in the field of cyber security, such as brute force attacks, data scraping, and transaction fraud. The purpose of this study is to achieve rapid classification of new types of traffic and OOD detection simultaneously under few-shot cases using the proposed SPN framework. For real-world network security applications, SPN's high performance in few-shot multi-classification and OOD detection can simultaneously meet the needs for cost-effective training, discovering new types of traffic, and achieving high-precision classification. SPN is capable of classifying new attack types based on their traffic features and can accurately distinguish known malicious traffic. This reduces the likelihood of false negatives in intrusion detection, thereby enhancing detection accuracy.

Based on the idea of metric learning, this framework effectively integrates the strong discriminative ability of Siamese networks and the few-shot classification capability of Prototypical networks. Additionally, the framework also supports the detection of OOD samples. Firstly, we design a 3D convolution for feature extraction and deeply integrate the prototype computation with the Siamese network, combining the fast classification capability of meta-learning with the powerful discriminative ability. Secondly, ID and OOD margin loss functions are introduced during training, allowing the model to detect OOD samples while performing few-shot traffic classification. Finally, this model possesses the characteristics of low complexity and fast computation speed. For the task of multi-class traffic classification, SPN achieves the best results on the CIC-IDS2017 dataset in all methods based on few-shot learning. On the USTC-TFC2016 dataset, both Acc and macro-P reaches optimal levels, while macro-F1 and macro-R are on par with the second-best model. For OOD sample detection, SPN reaches the optimal Acc indicator on the CIC-IDS2017 dataset, surpassing the second-best model by 7.27%. On the USTCTFC2016 dataset, SPN achieves Acc and F1 scores exceeding 99% and 98%, respectively. For intrusion detection tasks, SPN achieves the best Acc results on the ISCX2012 dataset, surpassing the second-best model in the few-shot methods by 0.79%. On the CIC-IDS2017 dataset, both Acc and DR reaches optimal results, surpassing the second-best model by 0.77% and 0.45% respectively.

In addition, in this study, the selection of an appropriate length for traffic data relies on prior knowledge. Improving the representation of samples or imposing more reasonable constraints on samples in the latent space may enhance the performance of the model. In future work, we will explore an adaptive data trimming method and also consider

reducing the reliance on data labels through semi-supervised or unsupervised approaches. The utilization of pre-training and contrastive learning is a promising direction for models to acquire more effective representations or normalizing the vectors in the latent space and imposing angle constraints, which will also be an area of our exploration in the future.

REFERENCES

- [1] 2023 *Imperva Bad Bot Report*. Accessed: 2023. [Online]. Available: <https://www.imperva.com/>
- [2] L. Zhao, Z. Ou, L. Zhang, and S. Li, "Hybrid fine-tuning strategy for few-shot classification," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–12, Oct. 2022.
- [3] H. H. Kim, D. Woo, S. J. Oh, J.-W. Cha, and Y.-S. Han, "ALP: Data augmentation using lexicalized PCFGs for few-shot text classification," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 10, pp. 10894–10902.
- [4] H. Wang, S. Tian, Y. Fu, J. Zhou, J. Liu, and D. Chen, "Feature augmentation based on information fusion rectification for few-shot image classification," *Sci. Rep.*, vol. 13, no. 1, p. 3607, Mar. 2023.
- [5] H.-J. Ye, L. Han, and D.-C. Zhan, "Revisiting unsupervised meta-learning via the characteristics of few-shot tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3721–3737, Mar. 2023.
- [6] S. Rao, J. Huang, and Z. Tang, "Adaptive regularized warped gradient descent enhances model generalization and meta-learning for few-shot learning," *Neurocomputing*, vol. 537, pp. 271–281, Jun. 2023.
- [7] H. Guo, X. Zhang, Y. Wang, B. Adebisi, H. Gacanin, and G. Gui, "Few-shot malware traffic classification method using network traffic and meta transfer learning," in *Proc. IEEE 96th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2022, pp. 1–5.
- [8] H. Sun, L. Wan, M. Liu, and B. Wang, "Few-shot network intrusion detection based on prototypical capsule network with attention mechanism," *PLoS ONE*, vol. 18, no. 4, Apr. 2023, Art. no. e0284632.
- [9] Q. Zhou, L. Wang, H. Zhu, and T. Lu, "Few-shot website fingerprinting attack with cluster adaptation," *Comput. Netw.*, vol. 229, Jun. 2023, Art. no. 109780.
- [10] J. Li, C. Gu, L. Luan, F. Wei, and W. Liu, "Few-shot open-set traffic classification based on self-supervised learning," in *Proc. IEEE 47th Conf. Local Comput. Netw. (LCN)*, Sep. 2022, pp. 371–374.
- [11] H. Lin, Y. Yan, and G. Chen, "Boosting low-resource intent detection with in-scope prototypical networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 7623–7627.
- [12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
- [13] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Jul. 1995.
- [14] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [15] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
- [16] G. Andresini, A. Appice, and D. Malerba, "Nearest cluster-based intrusion detection through convolutional neural networks," *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106798.
- [17] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection," *Comput. Netw.*, vol. 144, pp. 111–119, Oct. 2018.
- [18] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Secur. Commun. Netw.*, vol. 2018, pp. 1–9, Jul. 2018.
- [19] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
- [20] A. Pektaş and T. Acarman, "A deep learning method to detect network intrusion through flow-based features," *Int. J. Netw. Manage.*, vol. 29, no. 3, p. e2050, 2019.
- [21] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, vol. 2, no. 1, 2015, pp. 1–30.

- [22] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.
- [23] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
- [24] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.
- [25] J. Yang, H. Li, S. Shao, F. Zou, and Y. Wu, "FS-IDS: A framework for intrusion detection based on few-shot learning," *Comput. Secur.*, vol. 122, Nov. 2022, Art. no. 102899.
- [26] Z. Shi, M. Xing, J. Zhang, and B. H. Wu, "Few-shot network intrusion detection based on model-agnostic meta-learning with 12f method," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Jul. 2023, pp. 1–6.
- [27] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
- [28] A. S. Iliyasa, U. A. Abdurrahman, and L. Zheng, "Few-shot network intrusion detection using discriminative representation learning with supervised autoencoder," *Appl. Sci.*, vol. 12, no. 5, p. 2351, Feb. 2022.
- [29] K. Li, W. Ma, H. Duan, H. Xie, and J. Zhu, "Few-shot IoT attack detection based on RFP-CNN and adversarial unsupervised domain-adaptive regularization," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102856.
- [30] O. K. Shirekar and H. Jamali-Rad, "Self-supervised class-cognizant few-shot classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2022, pp. 976–980.
- [31] C. Yang, G. Xiong, Q. Zhang, J. Shi, G. Gou, Z. Li, and C. Liu, "Few-shot encrypted traffic classification via multi-task representation enhanced meta-learning," *Comput. Netw.*, vol. 228, Jun. 2023, Art. no. 109731.
- [32] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," 2016, *arXiv:1610.02136*.
- [33] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," 2018, *arXiv:1802.04865*.
- [34] J. Chen, Y. Li, X. Wu, Y. Liang, and S. Jha, "Robust out-of-distribution detection for neural networks," 2020, *arXiv:2003.09711*.
- [35] H. Wei, R. Xie, H. Cheng, L. Feng, B. An, and Y. Li, "Mitigating neural network overconfidence with Logit normalization," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 23631–23644.
- [36] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1199–1208.
- [37] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.
- [38] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2519–2533, Sep. 2015.
- [39] P. A. A. Resende and A. C. Drummond, "Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling," *Secur. Privacy*, vol. 1, no. 4, p. e36, 2018.
- [40] J. Wang, H. Zhao, J. Xu, H. Li, H. Zhu, S. Chao, and C. Zheng, "Using intuitionistic fuzzy set for anomaly detection of network traffic from flow interaction," *IEEE Access*, vol. 6, pp. 64801–64816, 2018.



GUOHUA WU received the B.S. degree from the Shandong University of Technology, Jinan, China, in 1992, the M.S. degree from the National Institute of Metrology, Beijing, China, in 1995, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1998. He is currently a Professor with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou. He is also the Executive Director of the Information Security Laboratory. His current research interests include

information systems, model-driven architecture, data mining, and digital health.



ZHEN ZHANG received the degree in computer science from Zhejiang University, in 2005. He is currently an Associate Professor with the School of Cyberspace, Hangzhou Dianzi University. His current research interests include information visualization, information security, information hiding, and confidential technology.



YONGJIE TONG received the M.S. degree from the School of Physics, Nankai University, Tianjin, China, in 2019. Since 2019, he has been engaged in the field of data mining. He is currently a Data-Mining Engineer with Zhongfu Information Company Ltd., Jinan, China. His research interests include machine learning and optimization, graph mining, and cyber security.



GONGXUN MIAO received the M.S. degree from the School of Computer Science and Technology, Shandong University of Technology, Zibo, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. His current research interests include computer cyber security, information systems, data mining, and information security.



BING LU received the B.Eng. and M.S. degrees from the College of Computer Science and Technology, Wuhan University of Technology, China, in 2015 and 2017, respectively. He is currently a Data-Mining Engineer with Zhongfu Information Company Ltd., Jinan, China. His research interests include machine learning and its application, data mining, and information security.

...