Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.048793

ARTICLE

NFHP-RN: A Method of Few-Shot Network Attack Detection Based on the Network Flow Holographic Picture-ResNet

Tao Yi[1,3], Xingshu Chen[1,2,*], Mingdong Yang[3], Qindong Li[1] and Yi Zhu[1]

[1]School of Cyber Science and Engineering, Sichuan University, Chengdu, 610065, China

[2]CyberScience Research Institute, Sichuan University, Chengdu, 610065, China

[3]Chengdu Fengwei Technology Co., Ltd., Chengdu, 610041, China

*Corresponding Author: Xingshu Chen. Email: chenxsh@scu.edu.cn

Received: 18 December 2023 Accepted: 22 February 2024 Published: 16 April 2024

ABSTRACT

Due to the rapid evolution of Advanced Persistent Threats (APTs) attacks, the emergence of new and rare attack samples, and even those never seen before, make it challenging for traditional rule-based detection methods to extract universal rules for effective detection. With the progress in techniques such as transfer learning and meta-learning, few-shot network attack detection has progressed. However, challenges in few-shot network attack detection arise from the inability of time sequence flow features to adapt to the fixed length input requirement of deep learning, difficulties in capturing rich information from original flow in the case of insufficient samples, and the challenge of high-level abstract representation. To address these challenges, a few-shot network attack detection based on NFHP (Network Flow Holographic Picture)-RN (ResNet) is proposed. Specifically, leveraging inherent properties of images such as translation invariance, rotation invariance, scale invariance, and illumination invariance, network attack traffic features and contextual relationships are intuitively represented in NFHP. In addition, an improved RN network model is employed for high-level abstract feature extraction, ensuring that the extracted high-level abstract features maintain the detailed characteristics of the original traffic behavior, regardless of changes in background traffic. Finally, a meta-learning model based on the self-attention mechanism is constructed, achieving the detection of novel APT few-shot network attacks through the empirical generalization of high-level abstract feature representations of known-class network attack behaviors. Experimental results demonstrate that the proposed method can learn high-level abstract features of network attacks across different traffic detail granularities. Compared with state-of-the-art methods, it achieves favorable accuracy, precision, recall, and F1 scores for the identification of unknown-class network attacks through cross-validation on multiple datasets.

KEYWORDS

APT attacks; spatial pyramid pooling; NFHP (network flow holo-graphic picture); ResNet; self-attention mechanism; meta-learning

1 Introduction

Currently, Advanced Persistent Threat (APT) attacks exploit existing or novel vulnerabilities to infiltrate internal networks and hosts, employing various adversarial techniques that alter their

attack behavior patterns. The vulnerabilities, methods, and pathways used in such attacks are often unprecedented, leading to the emergence of unknown attack characteristics. Limited threat sample quantities and the absence of samples pose challenges in constructing effective attack detection models. According to Trojan analysis reports from APT organizations [1], advanced Trojans such as "NOPEN" employ multiple adversarial techniques, including payload configuration data, resource, and function encryption, to counteract antivirus and sandbox detection. In terms of network communication, these Trojans use asymmetric encryption, waiting for network requests to activate, and other characteristics to counteract network-side detection. Moreover, on the host side, they employ techniques such as memory residency and forensic self-destruction, making sample extraction difficult and threatening features challenging to capture.

Conventional deep learning methods require a large amount of labeled data for learning. For network attack identification, if a particular category of attack samples is scarce, the model struggles to learn useful attack features for classification. With the development of transfer learning and meta-learning techniques, few-shot training and detection have greatly improved. However, challenges persist, such as sample feature loss and insufficient learning from source sample features. Researchers like Xu et al. [2], Snell et al. [3], Xiang et al. [4], Yu et al. [5], Zamir et al. [6], among others, have used transfer learning or meta-learning frameworks to learn initialization experience weights on a large number of similar tasks, effectively enhancing the model's learning ability on new tasks. While meta-learning methods address few-shot recognition to some extent, using average difference values for classification when there is significant fluctuation in intra-class sample features can lead to misjudgments. Transfer learning methods also face issues such as sample-specific feature loss during the transfer process.

Additionally, challenges arise in representing attack features suitable for meta-learning frameworks. These challenges include the inability of time-series sequence traffic features to adapt to the fixed-length input requirement of deep learning and the difficulty in high-dimensional few-shot network attack features' high-level abstract representation. Rong et al. [7] proposed a malicious web request detection method using character-level Convolutional Neural Networks (CNN), converting WEB request parameters into indexed matrix feature vectors and inputting them into a CNN detection model for malicious and benign request determination. Ding et al. [8] introduced an intrusion detection method based on deep convolutional neural networks, which preprocesses network traffic data sequences, converts 121-dimensional features directly into 11×11 black-and-white images, and inputs them into a CNN for feature dimension reduction and recognition. The above methods have taken into account the issue of preprocessing deep learning input data. However, compared to the automatic extraction of detailed features from the original data and the method of constructing fixed-length fine-grained multi-channel color image features, there is a gap in terms of time efficiency and maintaining traffic detail features when considering sequence self-padding transformation and artificial black-and-white image feature construction methods. Long et al. [9–15] and others generally use different encoder methods to extract low-level features from flow sequences to achieve attack classification. The extraction of high-level abstract features in the case of insufficient samples has not been considered, especially the inadequate consideration of the high-level abstract representation of contextual features in attack session traffic.

To address these issues, this paper proposes a few-shot attack detection method based on NFHP-RN. Firstly, it introduces a multi-granularity, multi-channel holographic image feature construction method (Network Flow Holographic Picture, NFHP) for network traffic data, concretizing network attacks in the form of visual images. This method overlays multi-granularity detailed traffic context features, constructing a three-dimensional holographic perspective feature image through images. This

approach effectively represents details of traffic context relationship features of different granularity levels while improving the effective utilization of high-dimensional features. Secondly, through an improved ResNet network, it extracts high-level abstract features of network traffic, efficiently utilizing holographic image features to address the challenge of expressing unified attack features for few-shot attacks in the presence of large-scale traffic backgrounds. Thirdly, it constructs a meta-learning detection model, utilizing the experience gained from classifying known malicious samples to learn optimal network initialization weights. An attention matching mechanism is introduced to enhance detection accuracy by calculating the similarity between target samples and known samples. The contributions of this paper are as follows:

1. For the existing detection methods, it is difficult to express the long-period session relationship of APT attacks, difficult to construct features, and difficult to meet the needs of fixed-scale input of deep learning models due to irregular feature sequences. An innovative method based on Spatial Pyramid Pooling (SPP) is proposed to use convolution kernels of different sizes respectively to carry out feature pooling calculations of multi-session behavior, such as average, maximum, minimum, standard deviation and mode. Bilinear interpolation is performed to upsample the obtained multi-level feature images. The feature images of different levels are scaled to the same feature space, and the feature images of all levels are spliced to get the NFHP traffic image features. The final NFHP image contains the original traffic long-period session relationship feature and the original traffic feature, and has the fixed dimension to meet the fixed size input requirements of deep learning.
2. NFHP images preposition part of feature construction, which reduces the size of model network to a certain extent and speeds up the learning speed. In addition, the preposition image feature construction method can also be accelerated through algorithm optimization or parallel computation to improve performance. In order to extract higher-order abstract features of NFHP images, the problems of few-shot attack detection such as few samples, not rich sample features, and not obvious main features can be solved, and the relationship between features in traffic can be better expressed. According to the characteristics of NFHP images, ResNet network is improved to make it perceive the different characteristic eigenvalues calculated in NFHP, and extract the abstract representation of the key behaviors of network attacks.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work, Section 3 details the proposed method, Section 4 presents experimental results and analysis, and finally, Section 5 concludes with summary remarks.

2 Related Work

With the rapid development of network technology, the network has become an indispensable tool in people's lives. But at the same time, such convenient platform has also induced abundant attacks [16]. For better defending, researchers have proposed various corresponding solutions, benefitting the industrial [17,18], vehicle [19], mobile [20] scenarios and so on. This work focuses on the attack detection under the network scenario, which involves two main components. The first component is to extract and represent features from network traffic, and the second component is to identify malicious traffic based on limited data. In the following, we review the existing methods for both components.

2.1 Feature Extraction and Representation on Traffic

Tang et al. [21] and others extracted three decision features from the traffic data, namely the coefficient of variation of TCP traffic, the wavelet packet energy entropy of TCP (Transmission Control Protocol) traffic , and the Pearson correlation coefficient between TCP and total traffic, to distinguish normal traffic from traffic under LDoS attacks. Marques et al. [22] converted the original traffic data into grayscale images used CNN to extract the spatial features of traffic from grayscale images. Meanwhile he used recurrent neural network (RNN) to extract the temporal features of traffic. Demianiuk et al. [23] proposed a contrastive learning-based model to learn an optimized traffic representation function, using positive and negative sample pairs from the traffic graph, thus improving the quality of traffic representation. Ma et al. [24] used the density-based DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to divide the traffic records into different traffic clusters according to the time interval and byte number of the traffic records. Cai et al. [25] represented the features of network traffic by using a Bidirectional Temporal Convolutional Network (BiTCN). BiTCN consists of two independent TCN sub-modules, which perform forward and backward convolution operations on the input network traffic sequence respectively, to capture the bidirectional semantic features of network traffic. Dodia et al. [26] extracted the top three most active Tor connections from each PCAP (Packet CAPture) file and used 150 web fingerprint features to describe the traffic patterns of each connection, while introducing 40 novel host-level features to capture the behavioral characteristics of malware. Finally, the paper combines the connection-level features and the host-level features of each Tor connection to form a 190-dimensional feature vector. Lee et al. [27] selected 21 features from the data stream and converted each feature in the network traffic data to 0 or 1 using a bitmap rule, dividing them into normal traffic or attack according to the range of feature values, thereby reducing the size and dimension of the data. Feature representation methods based on deep learning not only develop rapidly in the field of network attack detection, but also become the main method of feature representation in other research fields [28,29].

2.2 Attack Detection with Limited Data

One solution is autonomous learning, calibrating samples, and expanding the available information space. Tikekar et al. [30], Jiang et al. [31], and others used heuristic feature rule methods to automatically learn the performance of a small number of existing samples for detecting attacks. Subsequently, Wang et al. [32] used graphs to address the shortcomings of individual flow structures, combining similarity detectors and stability-based graph detectors in the flow structure. Li et al. [33] introduced a pipeline framework using random forest feature selection and DBSCAN clustering attribute transformation, converting raw network data into attributes, effectively maintaining data information, and showing excellent results. However, such solutions are ineffective in the case of 0-day vulnerabilities and zero samples. Another solution is an adaptive detection framework. Ouyang et al. [34] introduced a novel IDS (Intrusion Detection Systems) based on small sample learning, named FS-IDS (Few-Shot Intrusion Detection Systems), for detecting network attacks against SCADA (Supervisory Control And Data Acquisition) networks. Zerhoudi et al. [35] used zero-shot learning to compensate for missing examples with semantic knowledge to better estimate unknown user behavior, addressing internal threat detection issues. Zhong et al. [36] proposed a Few-shot Class-Agnostic Self-Adaptive Anomaly Detection (FCAD) framework with Model-Agnostic Meta-Learning (MAML) for cases with few labeled samples of new network anomalies. Also, complex network attacks often hide in a large volume of normal network traffic, sharing similarities with normal network traffic and exhibiting significant changes in attack features. In scenarios with a lack of samples, small sample attack detection fails to achieve effective results. Zhu et al. [37] proposed

a cost penalty layer in the training process to address the unbalance data. Although it can improve the classification accuracy in unbalance data, it is challenging to directly apply to few-shot attack detection. Singh et al. [38] proposed a new meta-learning model to solve the problem of data imbalance by updating weights to reduce errors between target data and measured data.

As shown in Table 1, we compare the above methods in terms of high-level abstraction, importance of features, and limited data. Existing studies mainly discuss the time relationship between traffic in the time dimension, and lack consideration of data load, session attributes, communication object distribution and other characteristics. Especially when a small number of unknown network packets cannot be effectively identified and correlated, the opportunity to analyze and detect malicious behavior may be lost. Therefore, the key of small-sample attack detection under the condition of high-dimensional features is how to extract high-level abstract features to meet the deep learning model's comprehensive learning requirements for unknown small-sample attack features.

Table 1: The comparison of different network attack detection models

Method	Considering high-level abstract features	Considering importance of features	Considering limited data
Marques et al. [22]	×	×	×
Ma et al. [24]	√	×	×
Cai et al. [25]	√	√	×
Li et al. [33]	×	×	√
Ouyang et al. [34]	×	×	√
Zhong et al. [36]	√	√	√
Lan et al. [39]	√	×	×
Agrafiotis et al. [40]	×	√	×

3 Methodology

3.1 Method Framework

In order to achieve few-shot network attack detection, we propose a feature representation and extraction scheme based on NFHP-RN. This approach utilizes a multi-channel image to represent network attack traffic. By enhancing ResNet for feature encoding and employing a meta-learning approach with a self-attention mechanism, we aim to learn a classification method for unknown-category attacks from past limited-known attack recognition tasks. The framework of the detection method is illustrated in Fig. 1.

The methodology workflow is presented as follows:

(1) Traffic IP Grouping: Group traffic by IP addresses and arrange it chronologically. Each group contains communication traffic between two IPs. Calculate N-dimensional statistical features for each traffic entry.

(2) Construction of NFHP Image Features: For multiple session traffic within the same IP group, perform pooling calculations (average, maximum, minimum, standard deviation, and mode) based on the SPP method for each feature dimension, using convolutional kernels of different sizes. Apply bilinear interpolation upsampling to the obtained multi-level feature images, bringing different levels

of feature images to the same feature space. Concatenate all levels of feature images to generate NFHP traffic image features.

(3) RN (ResNet) High-Level Abstract Feature Extraction: Modify the pooling layer of ResNet, changing max pooling to average pooling to enable the convolutional network to capture differences in traffic behavior features. Calculate abstract features for NFHP images.

(4) Meta-Learning Classification: Build a meta-learning model with self-attention for classifying abstract features of traffic. Achieve classification of few-shot network attack traffic.

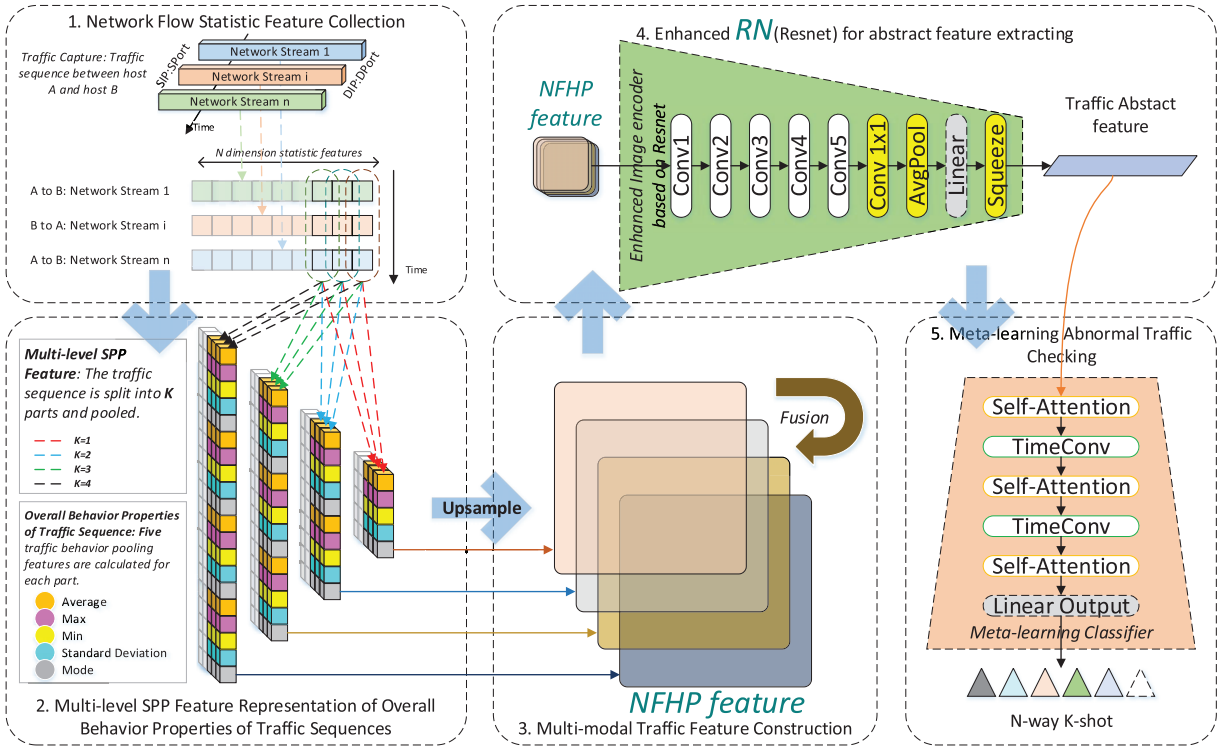

Figure 1: Framework for few-shot attack detection based on NFHP-RN

3.2 Network Attack Background Traffic IP Grouping

In a typical network attack, multiple associated flows are often involved. Many existing works focused on detecting attack traffic only consider a single target flow, neglecting the correlation between behaviors in multiple session flows. Therefore, to characterize the communication behavior from the attacking end to the attacked end in a single network attack and to represent the correlation of network attack traffic features across multiple session flows, we combine multiple normal or abnormal flows generated by the same IP address as background traffic with the target traffic to be detected. A single sample is constructed to simultaneously include the target traffic to be detected and the background traffic generated by the same IP. This enhances the feature representation of correlated traffic. The steps for network attack background traffic IP grouping are as follows:

(1) Group network traffic samples based on the same source IP and destination IP addresses, with each group containing communication traffic between two IP addresses.

(2) Sort the traffic sequences within a group based on time to reconstruct the temporal process of communication between IP addresses.

(3) Save the grouped and sorted traffic sequences for feature construction.

3.3 Construction of NFHP Image Features

The grouped background traffic sequences contain statistical features for each traffic entry but are challenging to represent the overall behavioral characteristics of traffic sequences and the correlation between different flows. Additionally, due to varying traffic quantities between different IPs, the directly computed feature lengths differ, making it unsuitable for deep network models. Furthermore, it is necessary to consider the granularity of feature extraction for background traffic of different lengths, with longer background traffic sequences yielding fewer detailed features after compression. To address these issues and enable the feature extractor to fully capture the overall behavioral characteristics of background traffic sequences and the correlation features between flows, we employ the SPP method for multi-scale, multi-attribute feature pooling of background traffic of different lengths. The obtained pooled features are transformed into images of uniform size and fused and concatenated to generate a multi-modal NFHP (Network Flow Holographic Picture). The image features encompass multi-scale behavioral characteristics of background traffic sequences as well as target traffic features, facilitating the feature extractor in capturing relationship features between flows. Meanwhile, the samples have a uniform size. The method for constructing NFHP image features is illustrated in Fig. 2, with the following steps:

(1) Assume the traffic grouping between two IP addresses is represented as $A = \{f_1, f_2, \ldots, f_i, \ldots, f_n\}$, where f_n is the nth traffic record. Let $X_i = \{x_1^i, x_2^i, \ldots, x_m^i, \ldots, x_k^i\}$ be the feature values of traffic f_i, where x_m^i is the mth feature. The feature matrix of this background traffic is given by Eq. (1).

$$A' = \begin{bmatrix} x_1^1 & \cdots & x_m^1 & \cdots & x_k^1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ x_1^i & \cdots & x_m^i & \cdots & x_k^i \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ x_1^n & \cdots & x_m^n & \cdots & x_k^n \end{bmatrix} \tag{1}$$

(2) Suppose all the pooling levels of SPP are denoted by $L = \{1, 2, \ldots, j, \ldots, z\}$. Let $x_m = \{x_m^1, \ldots, x_m^i, \ldots, x_m^n\}$ represent the mth column feature set in A', where the length of x_m is l_{x_m}. When calculating features for level j of x_m, start by slicing the feature values in x_m in sequential order into an average of j portions. If the elements in x_m are less than j, fill the remaining portions with zeros. This process results in a set of feature slices $P_j = \{p_1, p_2, \ldots, p_i, \ldots, p_j\}$, where each slice p_i is a continuous set of feature values in x_m represented as $p_i = \{x_m^a, x_m^{a+1}, \ldots, x_m^{a+c}\}$ with $a \geq 1$. This satisfies the condition specified in Eq. (2).

$$\begin{cases} l_{x_m} \geq j, \quad (j \in \mathbb{N}^+) \\ l_{x_m} - \lfloor l_{x_m}/j \rfloor \times j = 0 \\ c = l_{x_m}/j - 1 \end{cases} \tag{2}$$

(3) Calculate the average, maximum, minimum, standard deviation, and mode for each feature slice p_i. The set of these five statistical features is represented as $\varnothing_{p_i}^m = \{\text{avg}_{p_i}^m, \max_{p_i}^m, \min_{p_i}^m, \text{std}_{p_i}^m, \text{mode}_{p_i}^m\}$, and the calculation methods for each feature are shown in Eq. (3), respectively.

$$\begin{cases} \text{avg}_{p_i}^m & = \mu = \dfrac{\sum_{i=a}^{a+c} x_m^i}{j} \\ \max_{p_i}^m & = \max\{x_m^a, x_m^{a+1}, \ldots, x_m^{a+c}\} \\ \min_{p_i}^m & = \min\{x_m^a, x_m^{a+1}, \ldots, x_m^{a+c}\} \\ \text{std}_{p_i}^m & = \sqrt{\dfrac{\sum_{i=a}^{a+c}(x_m^i - \mu)^2}{c+1}} \\ \text{mode}_{p_i}^m & = \text{mode}(x_m^a, x_m^{a+1}, \ldots, x_m^{a+c}) \end{cases} \tag{3}$$

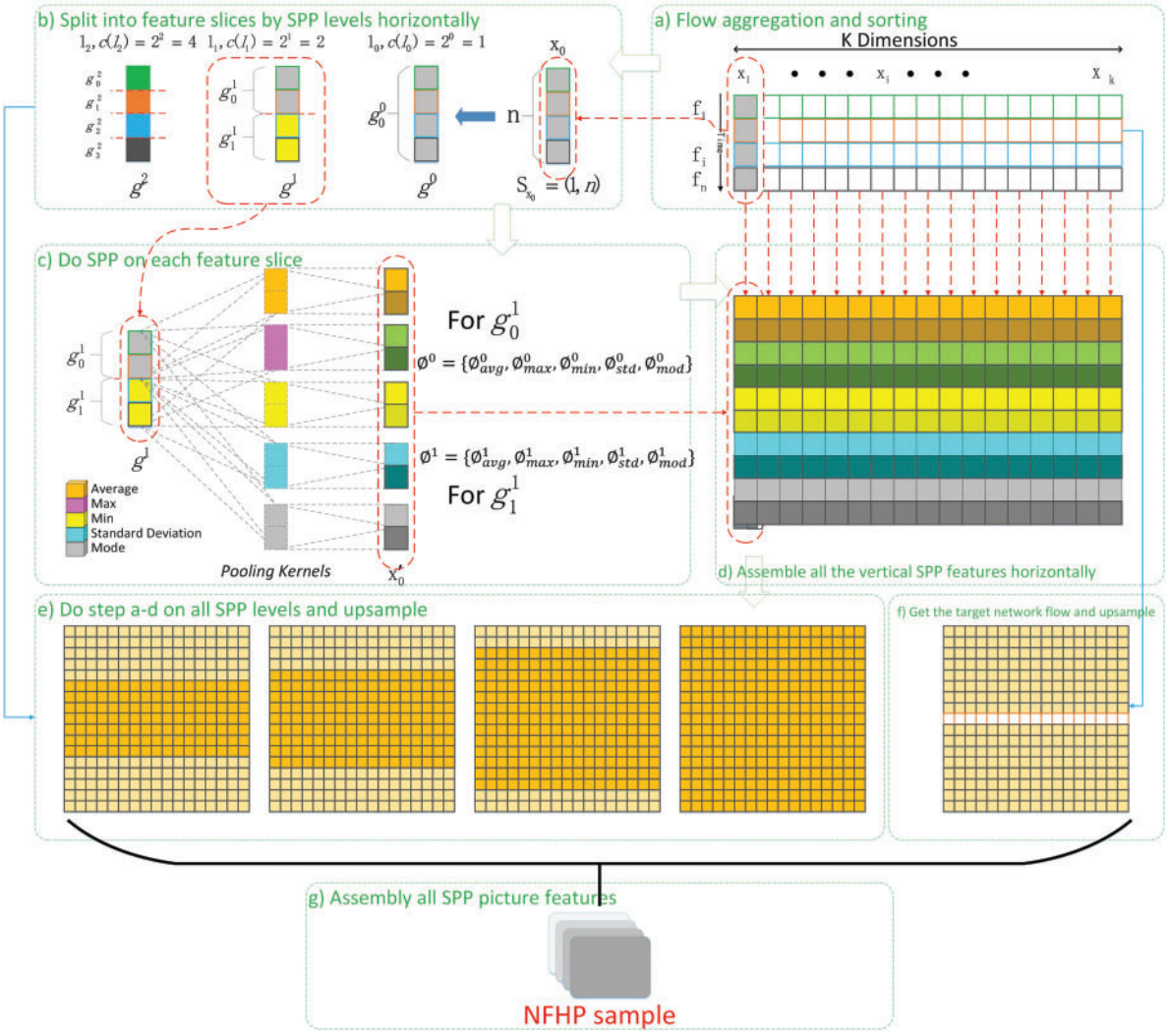

Figure 2: NFHP Image feature construction method

(4) For each p_i in P_j, repeat the previous step to obtain $\Omega^m = \{\Omega_{p_1}^m, \Omega_{p_2}^m, \ldots, \Omega_{p_i}^m, \ldots, \Omega_{p_j}^m\}$. The number of feature values in Ω^m is $j \times 5$. Then, for all feature columns x_m in A', calculate the feature matrix based on the 5 statistical features at the j-th level, as shown in Eq. (4).

$$\Omega_j = \begin{bmatrix} \Omega^1 \\ \Omega^2 \\ \vdots \\ \Omega^m \end{bmatrix} \tag{4}$$

(5) Use all SPP levels $L = \{1, 2, \ldots, j, \ldots, z\}$ to repeat the above steps for A', obtaining the feature set for each level $\varnothing = \{\varnothing_1, \varnothing_2, \ldots, \varnothing_j, \ldots, \varnothing_z\}$, where $\varnothing_j$ is the j^{th} feature with dimensions $(m, 5 \times 1)$. Therefore, the dimensions of each element in $\varnothing$ are given by Eq. (5).

$$S_{\varnothing} = \{(m, 5 \times 1), (m, 5 \times 2), \ldots, (m, 5 \times j), \ldots, (m, 5 \times z)\} \tag{5}$$

(6) Consider each feature matrix in $\varnothing = \{\varnothing_1, \varnothing_2, \ldots, \varnothing_j, \ldots, \varnothing_z\}$ as an individual feature image. Use bilinear interpolation and upsampling to scale each modality image to a uniform size and concatenate them to obtain the multi-modal NFHP background features. Bilinear interpolation oversampling allows flow feature layers with different granularities to be scaled to a uniform size while retaining the corresponding relationship between different layers. This alignment of flow behavior information aids in detecting information of different granularities within the same receptive field, allowing effective extraction of locally abstracted representations of flow behavior.

Bilinear interpolation uses four points (2 × 2) in the original image to compute a new point in the target image. It involves calculating three single linear interpolations in both directions, resulting in the value of the target point. This method achieves a balance between processing speed and effectiveness, as illustrated in Fig. 3.

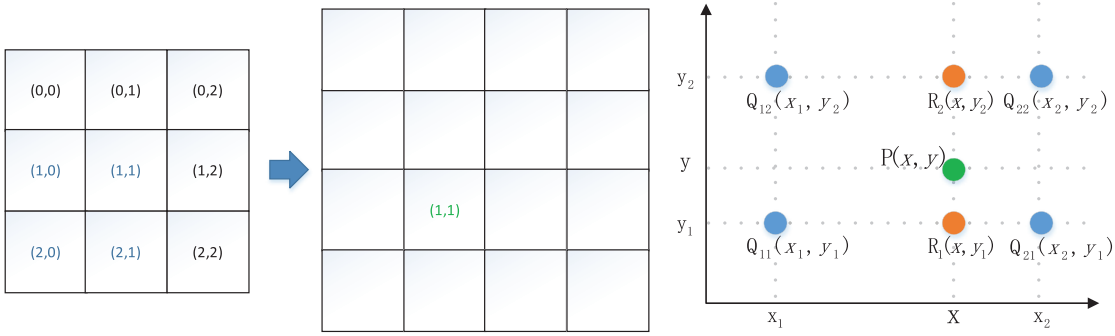

Figure 3: Bilinear interpolation upsampling of image

Let the four adjacent pixels in the image be denoted as $Q_{11}(x_1, y_1)$, $Q_{12}(x_1, y_2)$, $Q_{21}(x_2, y_1)$, and $Q_{22}(x_2, y_2)$. First, calculate the linear interpolation point $R_1(x, y_1)$ using Q_{11} and Q_{21}, as given by Eq. (6).

$$f(R_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{6}$$

Next, compute the linear interpolation point $R_2(x, y_2)$ using Q_{12} and Q_{22}, as expressed in Eq. (7).

$$f(R_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{7}$$

Finally, determine the value of the target point P (x, y) through the two points R_1 and R_2, as given by Eq. (8).

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{8}$$

(7) When considering different flows within the traffic group A as targets for classification, the traffic feature f_i is up-sampled to match the size of the NFHP background image. Subsequently, it is merged into the first color channel of the NFHP image, resulting in the final NFHP visual feature specific to the target traffic. The constructed NFHP visual features effectively capture the overall behavioral characteristics of the background traffic. This facilitates the feature extractor in extracting potential relationships between the target traffic and the background traffic. Furthermore, under the same background traffic conditions, NFHP images accurately express different target traffic scenarios.

3.4 High-Order Abstract Feature Extraction Based on Improved ResNet for NFHP

NFHP visual features consist of multiple layers, where the first layer represents the characteristics of the target traffic, and other layers represent the fusion features of the target traffic's spatiotemporal relationships at different granularities. The higher the level of the layer, the more details it contains about the relationship features, and the overall feature image contains more feature dimensions. To fully utilize these detailed features, an appropriate image feature encoder is needed. ResNet has shown good performance in image feature encoding, especially on small-scale datasets. For optimizing NFHP flow visual features, ResNet18 network is adjusted to precisely perceive different characteristic feature values calculated within NFHP. This adjustment helps in retaining the details of traffic behavior features and extracting abstract representations of critical behaviors in network attacks. The modified ResNet18 network is illustrated in Fig. 4.

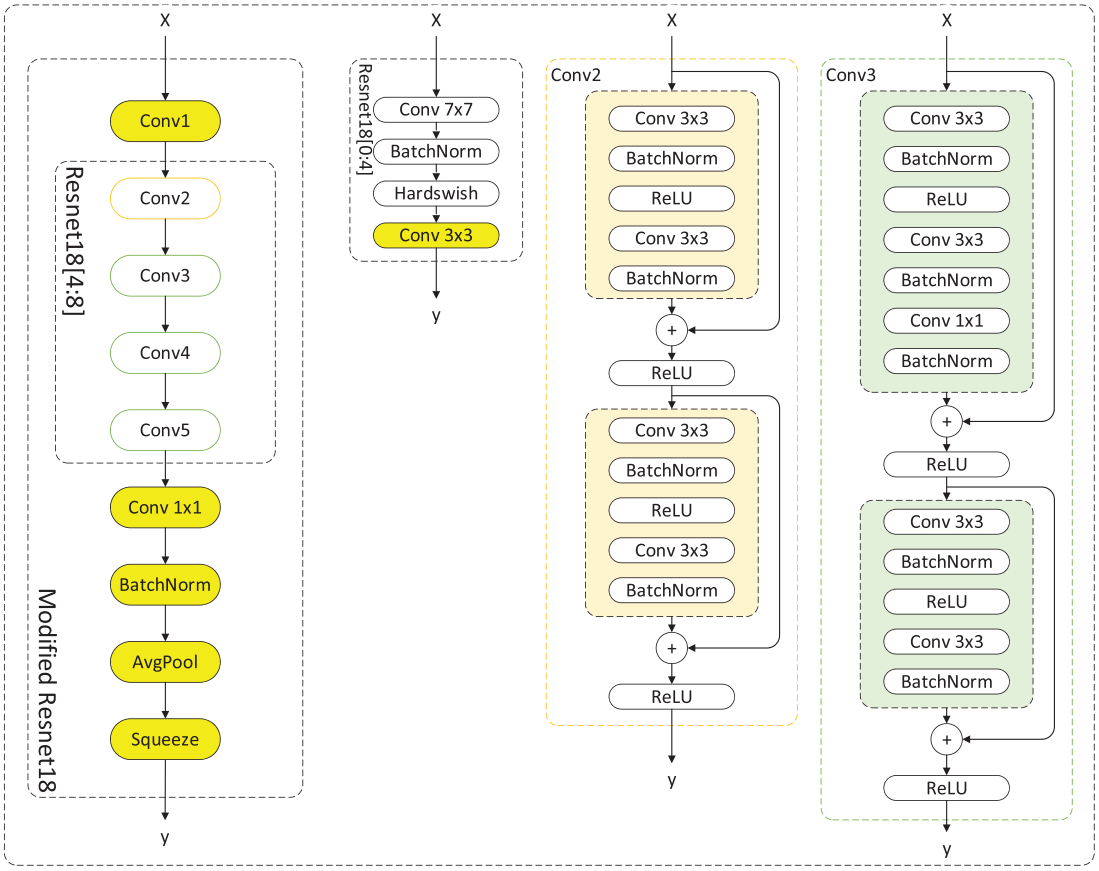

Figure 4: Model architecture of NFHP image feature encoder based on ResNet18

The ResNet18 network has been adjusted by removing the max-pooling in the Conv1 layer and replacing it with a convolutional layer. The Conv2-Conv5 network layers remain unchanged. Simultaneously, the final fully connected layer is replaced with a 1×1 convolutional layer, followed by a flattening operation (Squeeze) on the output to transform the feature dimensions into a 1-dimensional vector suitable for subsequent classification networks. The original max-pooling layer in the ResNet18 network may compromise the effectiveness of average, minimum, and mode characteristics in NFHP samples. The modified ResNet18 can perceive and extract the five NFHP characteristics. This network, acting as an NFHP feature encoder, captures the behavioral patterns of background traffic and the relationships between multiple flows.

3.5 Construction of Meta-Learning Model Based on Self-Attention Mechanism

A meta-learning classifier is constructed based on an attention mechanism. Utilizing the high-level abstract features extracted by ResNet18, past classification experiences are queried to detect and classify new types of few-shot network attacks. This approach significantly reduces the demand for malicious flow samples and enables rapid adaptation to novel variant samples.

Let our meta-learning task for network attack detection have a support set $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{k}$ and a test set $D_{\text{test}} = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{n}$, where $\hat{x}$ is the sample to be classified, and $\hat{y}$ is the predicted label for the sample, as shown in Eq. (9).

$$\hat{y} = \sum_{i=1}^{k} a(\langle \hat{x}, x_i \rangle) \cdot y_i \tag{9}$$

where a can be viewed as an attention mechanism, and y_i can be considered as memory units constrained by x_i. In other words, the attention matching mechanism introduces external memory to compute and learn more optimal network weights. Based on this principle, the constructed meta-learning network model is illustrated in Fig. 5.

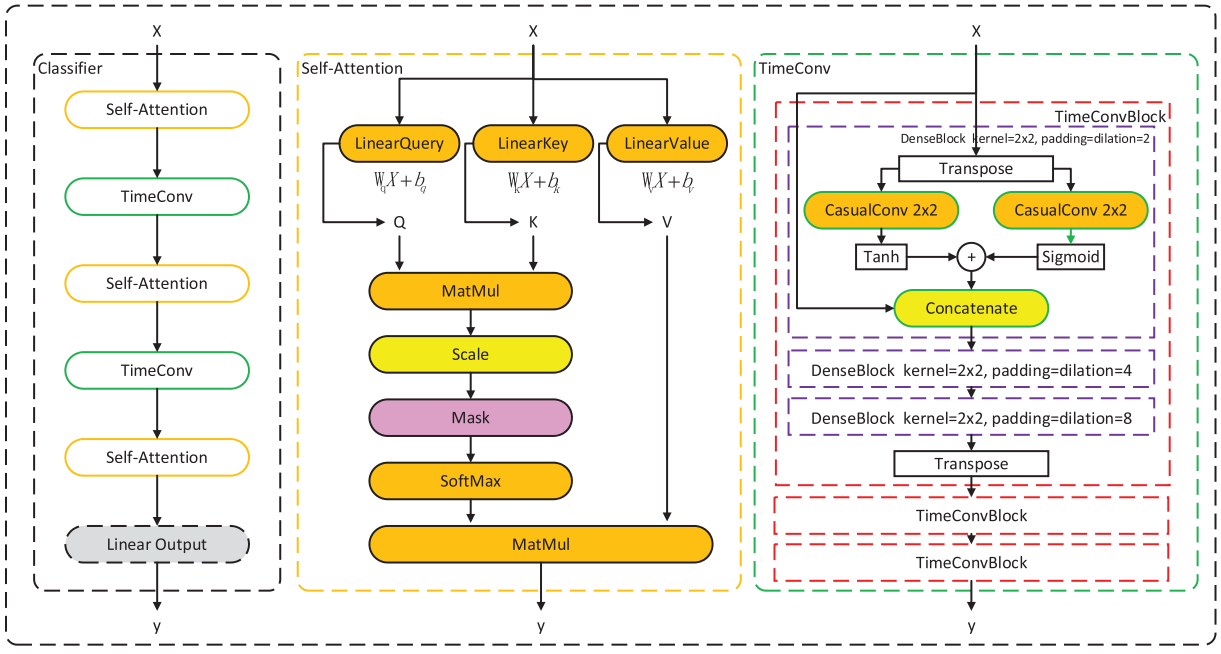

Figure 5: Diagram of meta-learning classification network based on attention mechanism

The classifier network comprises a network structure that alternates between causal convolutional layers and attention layers. In this structure, self-attention layers use the dot-product method to calculate attention weights. A TimeConvBlock consists of three Denseblocks, each containing two CasualConv causal convolution blocks, and each CasualConv containing a 1D convolutional network for receiving the transposed input vector. The size of the convolutional network in the three Denseblocks is 1, and the padding and dilation maintain two-fold growth, that is, the padding and dilation in the first DenseBlock are 2, the second is 4, and the third is 8. The parameters of all TimeConvBlocks are the same, the only difference is that the input tensor size of the next network needs to be based on the output size of the previous layer network, there is no need to manually set the number of channels of the input and output tensors. The internal structure maintains gradients through residual links. This design effectively captures implicit relationships between uplink and downlink traffic samples and their background traffic. Simultaneously, it aids meta-learning methods in leveraging past experiences when classifying samples of different categories.

3.6 Training and Application of the Few-Shot Detection Model

The training and validation process of the NFHP-RN attack traffic representation scheme and meta-learning network is consistent. The entire process is illustrated in Fig. 6.

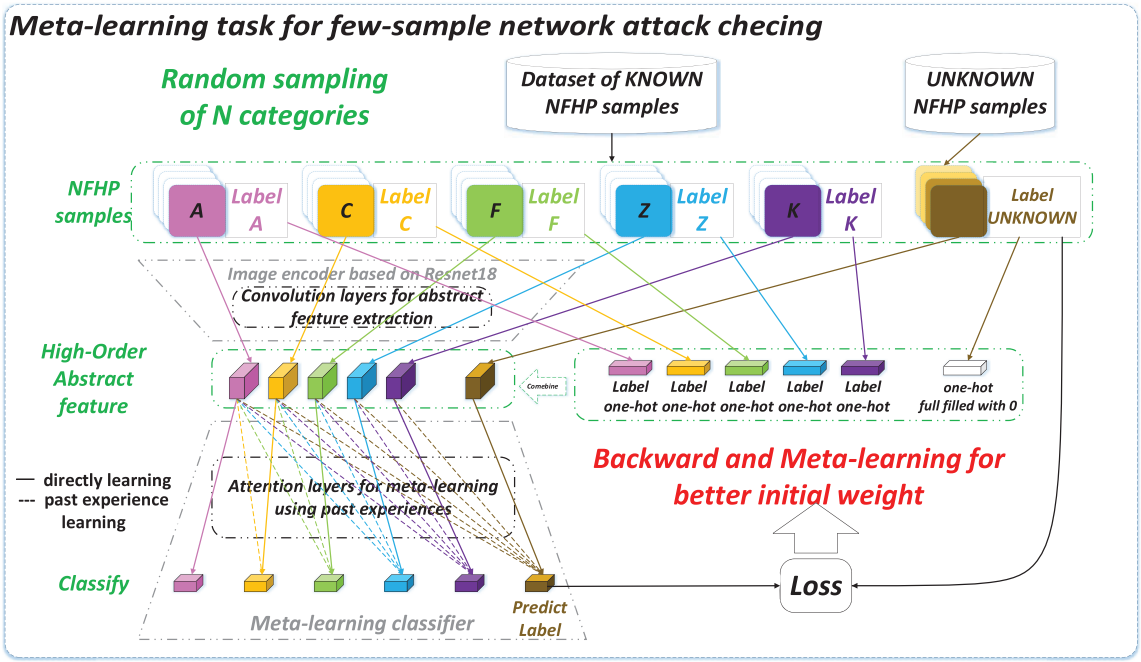

Figure 6: Workflow of meta-learning network attack detection based on NFHP-RN

The process steps during the model training phase are as follows:

(1) Sample from the support set and query set using the N-Way, K-Shot method. Set the predicted target sample category as a one-hot label with all zeros and convert all traffic samples to NFHP feature representation.

(2) Encode NFHP samples based on ResNet18.

(3) Concatenate the abstract features of the encoded samples with their corresponding one-hot vector labels.

(4) Input the concatenated samples into the meta-learning classifier to predict the labels of the target samples.

(5) Calculate the loss between the predicted labels of the target samples and the actual labels for backpropagation, achieving model network updates. Cross-entropy is used to characterize the distance between actual and expected output probabilities. The smaller the cross-entropy value, the closer the two probability distributions. Let p be the expected output probability distribution and q be the actual output probability distribution. $H(p, q)$ represents cross-entropy, and the loss function is shown in Eq. (10).

$$H(p,q) = -\sum_{x} (p(x)\log q(x) + (1-p(x))\log(1-q(x))) \tag{10}$$

The process during the model testing and usage phase is consistent with the training process. The model ultimately outputs predicted labels for unknown class target samples, achieving the prediction classification for few-shot network attacks.

4 Experimental Results and Analysis

4.1 Experimental Setups

The experiment was conducted on 6 data sets. These datasets contain different feature dimensions, different number of label categories, different attack traffic distribution proportions, different feature formats, and different industries. We will use these datasets to fully validate our methods, including accuracy, generalization, extensibility, and robustness.

(1) CICIDS-2017: CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack (CSV files).

(2) CICIDS-2018: CICIDS2018 includes seven different attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The attacking infrastructure includes 50 machines and the victim organization has 5 departments and includes 420 machines and 30 servers. The dataset includes the captures network traffic and system logs of each machine, along with 80 features extracted from the captured traffic using CICFlowMeter-V3.

(3) CIC-ToN-IoT: The BoT-IoT dataset was created by designing a realistic network environment in the Cyber Range Lab of UNSW Canberra. The network environment incorporated a combination of normal and botnet traffic.

(4) CIC-BoT-IoT: The datasets were collected from a realistic and large-scale network designed at the Cyber Range and IoT Labs, the School of Engineering and Information technology (SEIT), UNSW Canberra @ the Australian Defence Force Academy (ADFA). Including various attacking techniques, such as DoS, DDoS and ransomware, against web applications, IoT gateways and computer systems across the IoT/IIoT network.

(5) NF-UNSW-NB15-v2: The NetFlow-based format of the UNSW-NB15 dataset, named NF-UNSW-NB15, has been expanded with additional NetFlow features and labelled with its respective attack categories. The total number of data flows is 2,390,275 out of which 95,053 (3.98%) are attack

examples and 2,295,222 (96.02%) are benign. The attack examples are further classified into nine subcategories, the table below represents the NF-UNSW-NB15-v2 dataset's distribution of all flows.

(6) NF-ToN-IoT-v2: The publicly available pcaps of the ToN-IoT dataset are utilised to generate its NetFlow records, leading to a NetFlow-based IoT network dataset called NF-ToN-IoT. The total number of data flows is 16,940,496 out of which 10,841,027 (63.99%) are attack examples and 6,099,469 (36.01%), the table below lists and defines the distribution of the NF-ToN-IoT-v2 dataset.

The number of label categories in each dataset as Table 2.

Table 2: Class count and sample count of four datasets

Dataset	NFeature	NClass	NSample	Attack (%)	Benign (%)	Format	Industry
CICIDS-2017	82	15	2,830,743	19.37%	80.63%	CICFlowMeter-v4	IDS/IPS
CICIDS-2018	82	15	16,232,943	83.07%	16.93%	CICFlowMeter-v4	IDS/IPS
CIC-ToN-IoT	82	10	5,351,760	53.00%	47.00%	CICFlowMeter-v4	IoT
CIC-BoT-IoT	81	5	13,428,602	99.34%	00.66%	CICFlowMeter-v4	IoT
NF-UNSW-NB15-v2	43	9	2,390,275	3.98%	96.02%	NetFlow-based	IDS/IPS
NF-ToN-IoT-v2	43	10	16,940,496	63.99%	36.01%	NetFlow-based	IoT

Experimental environment as Table 3.

Table 3: Experimental environment

Items	Performance
CPU	Intel® Xeon® Silver 4210 CPU @ 2.20 GHz 2.20 GHz (2nd Processor)
GPU	Tesla V100 32GB $*$ 2
RAM	128GB DDR4 RAM
OS	Ubuntu22.04.1 LTS
Softwares	Anaconda 4.12.0, Python 3.9.12, pytorch 1.13.0, scikit-learn 1.1.1

For analysis, we use four following common Information retrieval evaluation metrics.

(1) Accuracy (A_c): It is the ratio of correctly classified instances (TP + TN) in front of all instances (TP + TN + FP + FN).

$$A_c = (TP + TN)/(TP + FP + TN + FN) \tag{11}$$

(2) Precision (P_r): It is the ratio of correctly classified attack flows (TP), in front of all the classified flows ($TP + FP$).

$$P_r = TP/(TP + FP) \tag{12}$$

(3) Recall (R_c): It is the ratio of correctly classified attack flows (TP), in front of all generated flows ($TP + FN$).

$$Rc = TP/(TP + FN) \tag{13}$$

(4) F-Measure (F_1): It is a harmonic combination of the precision and recall into a single measure.

$$F1 = 2/(1/Pr + 1/Rc) \tag{14}$$

For model parameter settings, we used consistent model structure and hyperparameters across all experiments except for the input NFHP dimensions. The hyperparameters are set in the following Table 4.

Table 4: Experimental hyperparameters

Hyperparameters	Value
Bs (batch_size)	32
Loss function	Cross-entropy
Optimizer	adamW
Lr (learning rate)	0.00001
N-Way (Number of meta-learning sampling categories)	5
K-Shot (Meta-learning predicts the number of categories)	1

NFHP-RN network model parameters need to be calculated according to hyperparameters:

(1) ResNet18 feature encoder: The input layer is the batch size multiplied by the NFHP image size [bs $*$ (N + K) channel, h, w], where N and K are N-Way and K-Shot values, respectively, channle is the number of NFHP image channels, and h and w are the height and width of NFHP images, respectively. The rest of the parameters are the same as the native ResNet18 network, except that maximum pooling in the last layer is replaced with average pooling.

(2) Meta-learning classifier: The structure of the meta-learning classifier is consistent with that described in Fig. 5. The convolution step of the three denseblocks in each TimeConv block is 1, the padding and dilation are 2, 4 and 8, respectively, and the convolution network parameters in each DenseBlock are consistent. Therefore, the parameters of the classifier model are determined by the output characteristics of the encoder without manual adjustment.

The entire NFHP-RN network model is connected and run according to steps 4 and 5 in Fig. 1. The structure and parameters will be used in all experiments. The computing power and model parameters required by different NFHP feature hierarchy models are as follows Table 5.

Table 5: Computing power and model parameters

Model	FLOPs (GB)	Params (MB)
ResNet18	1.82	11.69
NFHP-RN-Level_1	60.23	15.62
NFHP-RN-Level_2	60.23	15.62
NFHP-RN-Level_4	60.23	15.62
NFHP-RN-Level_8	60.23	15.62
NFHP-RN-Level_16	60.23	15.62
NFHP-RN-Level_32	187.35	15.62
NFHP-RN-Level_1-2	61.24	15.63

(Continued)

Table 5 (continued)

Model	FLOPs (GB)	Params (MB)
NFHP-RN-Level_1-2-4	62.25	15.63
NFHP-RN-Level_1-2-4-8	63.27	15.63
NFHP-RN-Level_1-2-4-8-16	64.28	15.64
NFHP-RN-Level_1-2-4-8-16-32	206.61	15.64

4.2 Impact of Background Flow Feature Characteristics on Model Classification Performance

The NFHP samples contain five overall flow characteristic features. We conducted multi-class experiments on four datasets using four machine learning and deep learning classification models. Firstly, classification training was performed based on the original features. Secondly, for each sample, after computing the five characteristic features, these features were directly concatenated to the target sample, constructing a linear feature sample set NFAF-LN (Network Flow Action Feature-Level N) for training with different SPP pooling levels. The algorithm process is outlined in Table 6.

Table 6: NFAF-LN feature construction

Algorithm 1: NFAF-LN feature Construction

Input: f as the original features of CICIDS2017 samples.
Output: the NFAF-LN feature $\mathbf{x}_f$ as multi-level network flow action feature vector.

1: Define F = Select all network flow samples with the same SIP and DIP of $\boldsymbol{f}$
2: Define $\varnothing_i$ = The max, min, avg, std and mode feature of F calculated by SPP of level i
3: Set $L = \{a, j, z\}$ as the NFAF levels
4: Calculate the $\varnothing_a$, $\varnothing_j$, $\varnothing_z$ on levels a, j and z
5: Then $\mathbf{x}_f$ = Concatenate($\boldsymbol{f}$, $\varnothing_a$, $\varnothing_j$, $\varnothing_z$)

Based on different levels of NFAF samples, the multi-class classification performance is shown in Table 7. Experimental comparisons were conducted on the KNN (K Nearest Neighbors), RF (Random Forest), NB (Naive Bayes), and MLP (Multi-layer Perceptron) models. The maximum number of samples for each class in the dataset was limited to 10,000, leveraging the performance of machine learning models on small datasets. SIP and DIP were removed from the sample features, retaining the original 82-dimensional features. Additionally, the NFAF-L1 features were computed according to Algorithm 1. The training used the same model parameters and underwent 10-fold cross-validation.

As shown in Fig. 7, the experiments indicate that incorporating NFAF background traffic behavioral features has a positive impact on the classification performance of the model across the CICIDS-2017, CICIDS-2018, CIC-ToN-IoT, and CIC-BoT-IoT datasets.

Table 7: NFAF-LN multi-classify on four datasets

Dataset	Model	Original features (82 dims)				NFAF-L1 (492 dims)			
		Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2017	KNN	99.27	91.97	86.35	87.37	99.6	91.35	91.03	91.12
	RF	84.85	90.62	75.66	78.02	98.06	95.47	90.61	92.05
	NB	13.07	14.11	9.03	5.08	41.88	37.5	30.42	26.57
	MLP	99.45	93.89	91.22	91.38	99.27	94.42	90.89	91.15
CICIDS-2018	KNN	98.22	96.54	93.76	94.7	98.39	97.33	93.68	94.62
	RF	88.67	93.33	81.42	82.87	90.00	94.5	83.88	86.36
	NB	26.3	14.42	19.84	14.74	26.87	19.49	20.19	16.32
	MLP	99.66	95.77	93.14	93.5	99.96	96.27	93.17	93.89
CIC-ToN-IoT	KNN	77.99	64.43	64.31	64.12	79.21	66.77	67.18	66.69
	RF	70.05	58.56	60.53	56.56	72.56	58.4	61.86	57.15
	NB	27.47	15.48	32.5	14.5	23.64	13.49	29.95	15.28
	MLP	78.38	67.96	68.13	65.78	82.67	68.5	69.88	66.72
CIC-BoT-IoT	KNN	68.04	72.56	73.16	72.83	68.15	72.76	73.3	73.0
	RF	64.5	66.26	67.79	64.78	65.2	67.2	68.24	64.48
	NB	19.35	22.47	22.91	09.10	12.78	14.86	26.46	10.8
	MLP	69.86	73.17	74.63	72.62	70.27	73.55	75.16	72.87

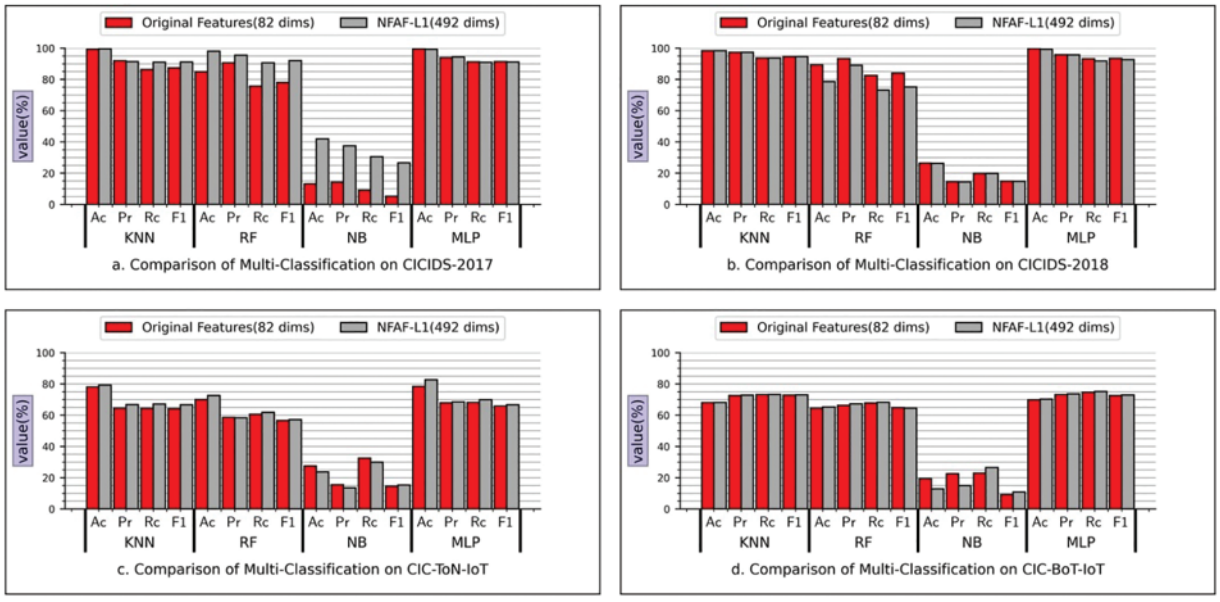

Figure 7: NFAF-LN multi-classify on four datasets

4.3 Impact of NFHP Images on Model Classification Performance

We conducted experiments to investigate the influence of constructing image representations of sample features on the model's classification performance after obtaining the 5 background traffic behavioral features. We compared the training results using the NFAF-LN feature dataset and the NFHP feature dataset. Additionally, we specifically constructed a one-dimensional convolutional model (NFAF-RN1d) consistent with the NFAF-RN model structure for comparison. Table 8 and Fig. 8 indicate that NFHP images can enhance the performance of the feature encoder and classifier. However, in scenarios with a higher number of feature dimensions, the NFAF-RN1d model may struggle to extract abstract features effectively, resulting in underfitting.

Table 8: Multi-classify of NFAF-RN1d and NFHP-RN on different datasets

Dataset	NFAF-RN1d				NFHP-RN			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2017	91.06	91.96	91.03	90.43	96.28	96.36	96.28	95.94
CICIDS-2018	90.26	91.08	90.23	89.5	94.76	94.85	94.73	94.25
CIC-ToN-IoT	92.24	92.56	92.19	91.56	95.12	95.15	95.1	94.65
CIC-BoT-IoT	91.53	92.87	91.53	90.96	95.11	95.49	95.07	94.61

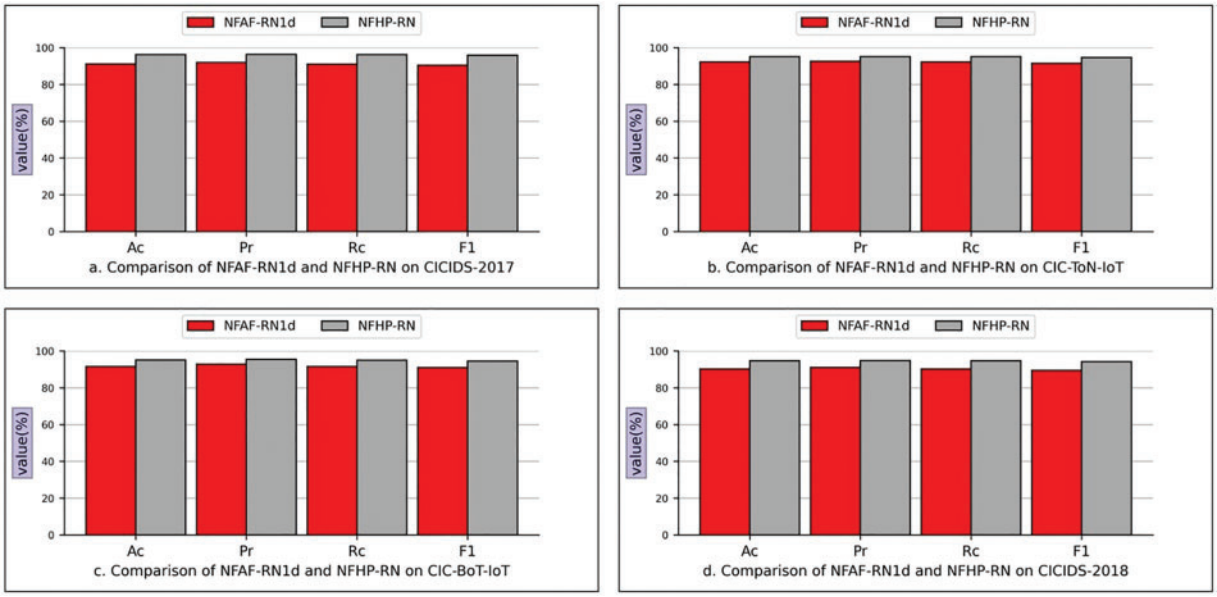

Figure 8: Multi-classify of NFAF-RN1d and NFHP-RN on different datasets

4.4 Impact of Different Hierarchical NFHP Features on Model Performance

In order to validate the influence of the granularity of background traffic representation on the model's classification performance, we constructed various NFHP feature datasets with different combinations of pooling layers and validated them on the datasets. Table 9 and Fig. 9 illustrate the impact of different hierarchical combinations of data on the model's classification performance.

Table 9: Effect of different SPP layers on datasets

Level	CICIDS-2017				CIC-ToN-IoT			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1	92.31	92.7	92.28	91.65	94.57	94.62	94.53	94.06
2	97.14	97.07	97.15	96.81	94.47	94.55	94.31	93.87
4	97.2	97.21	97.15	96.91	95.38	95.7	95.42	95.06
8	97.96	97.96	97.99	97.78	95.6	95.65	95.59	95.21
16	96.73	96.71	96.75	96.4	95.55	95.56	95.55	95.1
32	98.03	98.03	98.05	97.81	95.08	95.08	95.11	94.62
$1+2$	93.69	93.76	93.67	93.11	94.48	94.56	94.37	93.93
$1+2+4$	97.16	97.23	97.08	96.84	95.3	95.32	95.33	94.87
$1+2+4+8$	97.75	97.75	97.75	97.52	95.1	95.15	95.14	94.65
$1+2+4+8+16$	93.69	93.66	93.66	93.0	95.08	95.18	95.1	94.66
$1+2+4+8+16+32$	97.96	97.95	97.96	97.75	95.93	95.86	95.88	95.45
Level	CICIDS-2018				CIC-BoT-IoT			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1	94.51	94.66	94.36	93.94	94.95	95.66	94.87	94.41
2	93.04	93.33	93.01	92.39	95.29	95.29	95.25	94.81
4	94.92	94.99	94.91	94.45	94.93	94.98	94.87	94.41
8	95.41	95.56	95.48	95.05	95.36	95.79	95.35	94.91
16	96.2	96.26	96.2	95.85	95.18	95.63	95.09	94.64
32	93.47	93.6	93.48	92.84	94.93	95.19	94.94	94.49
$1+2$	95.2	95.21	95.11	94.67	95.12	95.69	95.07	94.61
$1+2+4$	94.52	94.51	94.51	93.99	95.18	95.49	95.18	94.62
$1+2+4+8$	94.28	94.29	94.24	93.71	95.22	95.39	95.28	94.79
$1+2+4+8+16$	95.03	95.1	95.08	94.57	94.84	94.98	94.85	94.35
$1+2+4+8+16+32$	95.69	95.77	95.7	95.28	95.06	95.21	94.9	94.47
Level	NF-UNSW-NB15-v2				NF-ToN-IoT-v2			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1	92.74	92.75	92.69	92.1	96.04	96.11	96.01	95.67
2	91.65	91.66	91.55	90.78	94.52	94.51	94.46	93.95
4	91.95	92.44	91.91	91.24	96.64	96.7	96.64	96.32
8	92.9	92.95	92.81	92.23	96.44	96.47	96.45	96.09
16	92.61	92.51	92.5	91.84	97.39	97.33	97.29	97.05
32	91.39	91.88	91.33	90.51	95.69	95.77	95.62	95.24
$1+2$	90.08	90.31	90.06	89.24	95.02	95.02	95.09	94.61
$1+2+4$	90.63	91.3	90.58	89.94	95.61	95.71	95.72	95.28
$1+2+4+8$	89.67	89.93	89.68	88.79	96.62	96.64	96.57	96.25
$1+2+4+8+16$	91.65	91.72	91.61	90.87	96.98	97.11	96.92	96.7
$1+2+4+8+16+32$	92.28	92.56	92.24	91.65	96.3	96.25	96.28	95.89

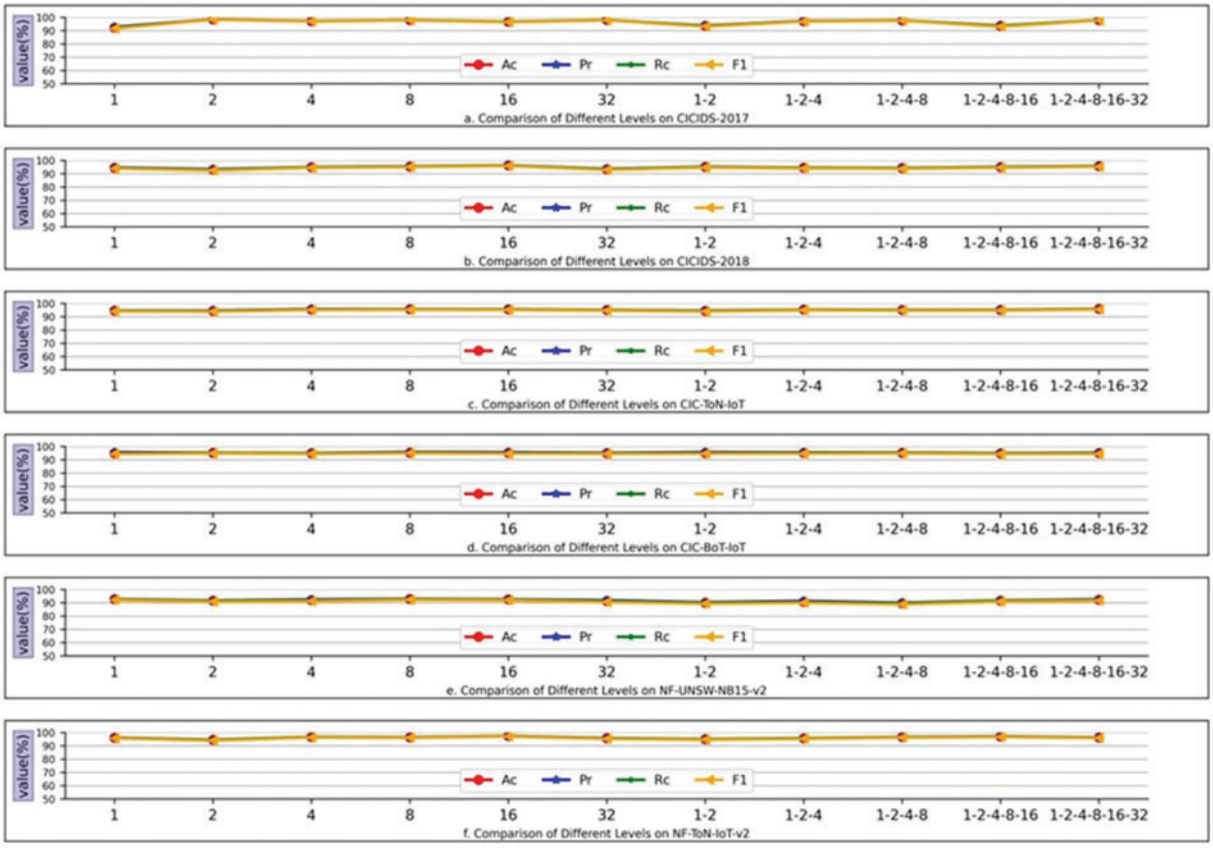

Figure 9: Effect of different SPP layers on datasets

The experiments indicate that the model's classification performance is influenced by the granularity of NFHP features. Coarse-grained NFHP features inadequately represent the details of attack traffic features, while higher levels provide a more refined representation of attack features. This results in lower classification performance on the 1 and 1-2 feature levels compared to using higher-level NFHP features such as 4, 8, 16, and 32. This characteristic is evident across all four datasets, showing that as the NFHP level increases, the model's classification ability improves.

Furthermore, consistent model structure and parameters were used for validation on different datasets (ResNet18 and a meta-learning classifier with 3 layers of self-attention and causal convolution). Higher-level NFHP features have more feature dimensions, causing a slight underfitting when reaching 16 to 32 layers, resulting in a slight decrease in model classification performance. Finally, since the number of original sample categories varies in each dataset, the model's ability to learn "feature extraction experience for known category samples" is inconsistent. This inconsistency leads to better performance on datasets with a higher number of original sample categories, such as CIDIDS-2017 and CIDIDS-2018. Therefore, the model effectively leverages the classification experience of multiple known categories, and the classification performance improves with an increasing number of initial sample categories, as demonstrated in the next experiment.

4.5 Cross-Validation on Different Datasets

To validate the performance of our approach in the context of few-shot malicious traffic detection and the transferability of model knowledge, we trained and cross-validated our model on different-layer NFHP sample datasets using six datasets. This exploration aimed to assess the model's detection performance on unknown-category attack traffic.

We initially trained the model on a single dataset and then conducted abnormal traffic detection on another dataset. Test results of CICFlowMeter-v4 series data set were averaged for all levels, and test results of all levels were displayed for NetFlow series data set. The results are presented in Tables 10–15. Moreover, Fig. 10 clearly illustrates the results of its transformation.

Table 10: Evaluations trained on CICIDS-2017 and test on other datasets (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Test dataset	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2018	73.21	77.49	73.14	70.98	92.98	93.18	92.97	92.38	97.57	97.57	97.63	97.36
CIC-ToN-IoT	65.14	70.12	65.26	63.5	93.42	93.54	93.28	92.73	95.97	96.02	95.88	95.53
CIC-BoT-IoT	69.14	73.59	69.0	65.78	94.99	95.36	94.96	94.55	95.57	95.8	95.46	95.08

Table 11: Evaluations trained on CICIDS-2018 and test on other datasets (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Test dataset	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2017	78.58	84.97	78.52	77.46	96.85	96.86	96.85	96.53	99.09	99.08	99.12	99.0
CIC-ToN-IoT	83.73	88.29	83.71	83.37	94.22	94.3	94.2	93.66	96.02	96.1	96.07	95.68
CIC-BoT-IoT	73.41	83.91	73.35	72.2	94.94	95.19	94.94	94.42	95.46	95.81	95.44	95.02

Table 12: Evaluations trained on CIC-ToN-IoT and test on other datasets (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Test dataset	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2017	84.7	87.32	84.56	83.89	96.45	96.36	96.48	96.07	98.94	98.97	98.93	98.84
CICIDS-2018	83.8	87.05	83.65	83.11	93.33	93.51	93.32	92.69	97.57	97.6	97.57	97.33
CIC-BoT-IoT	82.81	85.67	82.72	81.65	95.09	95.3	95.12	94.67	95.31	95.51	95.25	94.81

Table 13: Evaluations trained on CIC-BoT-IoT and test on other datasets (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Test Dataset	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
CICIDS-2017	55.04	62.1	54.84	52.64	93.15	93.37	93.05	92.47	98.22	98.24	98.18	98.04
CICIDS-2018	49.29	53.78	49.6	43.73	90.45	90.73	90.3	89.55	97.01	97.05	97.04	96.75
CIC-ToN-IoT	51.93	56.49	51.73	46.75	91.39	91.75	91.32	90.61	96.24	96.27	96.23	95.88

Table 14: Evaluations trained on NF-UNSW-NB15-v2 and test on NF-ToN-IoT-v2 (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Level	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1	72.56	75.96	72.34	69.71	94.88	95.28	94.79	94.5	97.53	97.63	97.59	97.39
2	73.66	79.76	73.79	71.47	93.84	93.95	93.7	93.2	98.62	98.63	98.63	98.46
4	77.81	82.73	77.51	75.67	96.5	96.59	96.62	96.25	98.59	98.54	98.46	98.32
8	84.19	85.43	84.16	82.84	96.5	96.58	96.56	96.25	98.09	98.22	98.02	97.93
16	78.75	81.5	78.45	76.45	96.22	96.34	96.34	95.96	98.41	98.61	98.4	98.36
32	75.44	80.04	74.78	72.79	95.22	95.62	95.25	94.96	98.12	98.06	98.23	97.95
$1+2$	80.0	83.11	80.49	78.46	93.75	93.92	93.68	93.06	98.19	98.29	98.29	98.12
$1+2+4$	72.88	78.0	72.91	70.03	94.25	94.59	94.32	93.91	97.94	98.12	97.97	97.87
$1+2+4+8$	83.81	84.42	83.48	82.4	93.5	93.58	93.53	92.92	97.47	97.27	97.42	97.06
$1+2+4+8+16$	79.47	84.24	79.45	78.22	95.81	95.85	95.65	95.27	98.03	98.18	98.07	97.95
$1+2+4+8+16+32$	77.97	80.54	78.05	75.97	94.97	95.16	94.85	94.54	97.78	97.9	97.85	97.68
Average	77.87	81.43	77.76	75.82	95.04	95.22	95.03	94.62	98.07	98.13	98.08	97.92

Table 15: Evaluations trained on NF-ToN-IoT-v2 and test on NF-UNSW-NB15-v2 (Trained 0 epochs, trained 10 epochs, trained 100 epochs)

Level	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1	83.66	84.78	84.17	82.86	91.81	91.95	91.68	90.99	95.19	95.48	95.13	94.73
2	83.22	86.95	82.81	82.22	90.66	91.0	90.7	90.03	94.12	94.22	94.15	93.63
4	84.5	87.82	84.68	83.96	92.16	92.18	92.51	91.67	94.97	95.19	94.77	94.49
8	85.38	87.55	85.62	84.63	91.22	91.61	91.05	90.38	94.75	95.01	94.89	94.45
16	84.38	85.91	83.85	83.26	92.53	92.46	92.16	91.63	94.78	94.7	94.65	94.18
32	82.38	84.58	81.71	80.96	92.75	92.95	93.16	92.39	94.03	93.75	93.96	93.23
$1+2$	84.56	86.11	84.59	83.65	91.06	91.22	90.62	90.06	93.16	93.32	93.02	92.5
$1+2+4$	84.56	86.01	84.04	83.42	90.81	91.56	90.88	90.29	93.78	93.81	94.27	93.4
$1+2+4+8$	85.62	88.83	85.95	85.21	91.97	92.45	92.1	91.4	94.16	94.38	94.46	93.83

(Continued)

Table 15 (continued)

Level	*Trained 0 epochs*				*Trained 10 epochs*				*Trained 100 epochs*			
	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1	Ac	Pr	Rc	F1
1 + 2 + 4 + 8 + 16	85.97	87.18	85.98	85.01	92.28	92.33	91.95	91.26	94.78	95.0	94.87	94.41
1 + 2 + 4 + 8 + 16 + 32	85.53	86.81	84.98	84.24	92.28	92.25	92.3	91.51	94.47	94.81	94.73	94.23
Average	84.52	86.6	84.4	83.58	91.78	92.0	91.74	91.06	94.38	94.51	94.44	93.92

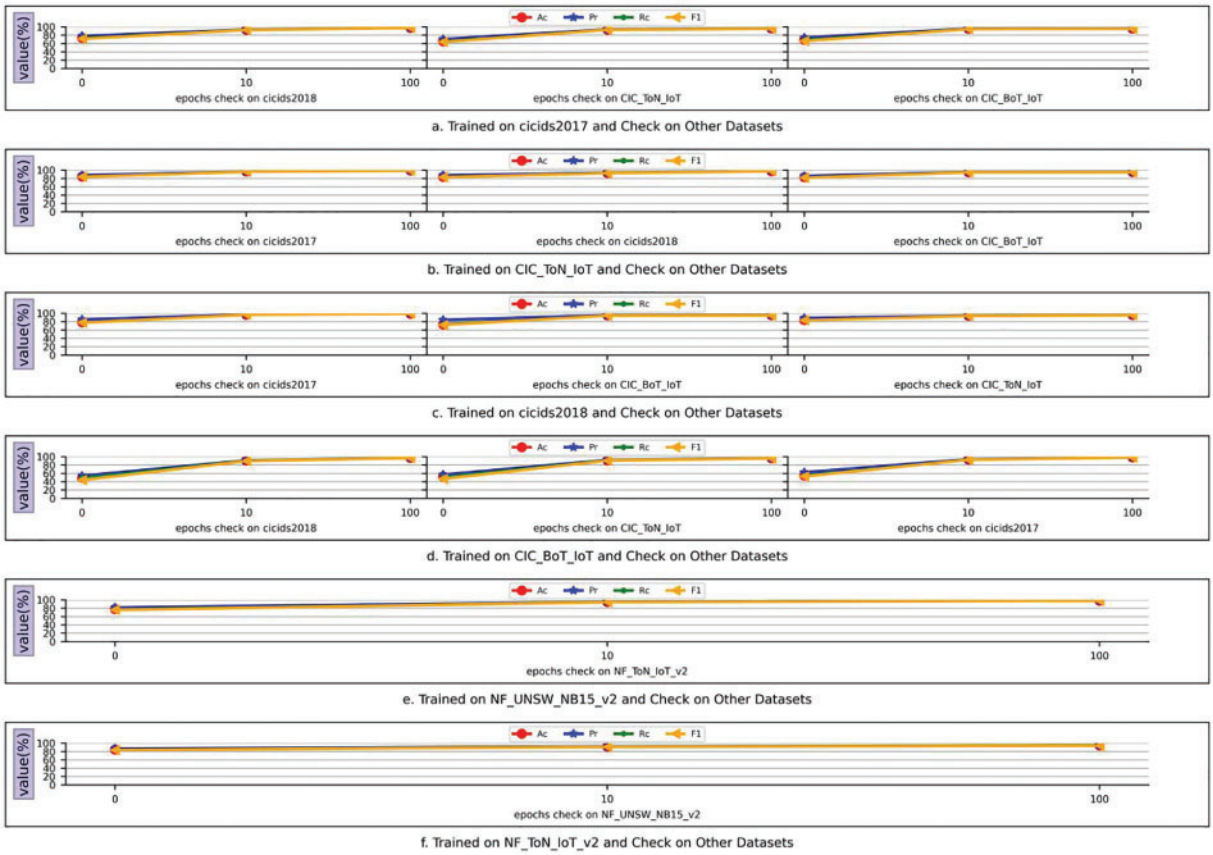

Figure 10: Evaluations trained on one dataset and test on other datasets

As evident from Tables 10 to 15 and Fig. 10, after training on one dataset and directly performing classification on another dataset, the model achieved a classification performance of over 80% in 0 epochs on CICIDS-2017, CICIDS-2018, and CIC-ToN-IoT datasets. This indicates that our approach can effectively transfer the recognition experience of known-category samples and generalize to unknown categories. However, the performance on CIC-BoT-IoT dataset during 0 epochs training and validation on other datasets was around 49%–55%. This is attributed to the fewer sample categories in the CIC-BoT-IoT dataset, preventing the model from fully learning the feature extraction patterns of network attacks. Nevertheless, in subsequent training rounds, as the model gained exposure to various

attack categories in other datasets, its classification accuracy quickly increased, reaching over 90% after 10 epochs.

Moreover, with increasing training epochs, the performance of all datasets rapidly improved at both 10 and 100 epochs. This indicates that as the model encounters more categories, it significantly strengthens its ability to extract a general abstract representation of network attacks. Additionally, the attention meta-learning model better focuses on effective representations in sample features, enabling accurate classification. This trend illustrates that "the more it sees, the more accurate the recognition" tendency. In conclusion, our proposed method, after thoroughly learning various categories of network attack samples, significantly enhances the ability to recognize unknown-category network attacks, It is robust to traffic data generated in different feature dimensions, different attack traffic distribution, different feature formats, and different types of network environments ,demonstrating considerable practical value.

5 Conclusion

In this paper, a series of programmatic methods including feature construction, feature extraction and detection tasks are proposed to solve several challenges in few-shot network attack traffic detection. The constructed NFHP flow image feature provides a multimodal representation of attack traffic and background flow and is suitable for neural networks with fixed-size inputs. The improved ResNet network uses convolutional layer instead of maximum pooling layer for feature calculation, extracts abstract features used by attention meta-learning models, and learns general patterns of different attack classes. The model implements the optimal initialization weights with different granularity on different data sets. The final classification model achieved high accuracy, precision, recall and F1 values, over 80% on previously unseen data, was able to adapt quickly to new samples, and trained with a minimum of unknown class samples. The encoder-decoder structure is adopted in this method. The encoder is conducive to the convenient expansion of the traffic preprocessing equipment, and the decoder can also allocate the encoder and classifier to multiple scalable devices for parallel computation, which is conducive to the operation of the solution in a large-scale network environment. The solution can be extended from four aspects: Feature preprocessing parallelism, encoder parallelism, classifier parallelism and feature detail extension. At the same time, the intermediate features of network traffic generated by this method should be effectively utilized. In the multi-source and multi-point security joint detection solution, the network traffic features in this paper are aligned with the software features collected by EDR by contrast learning, so that network or local threats can be quickly traced, which is one of the next research directions of this paper. However, our proposed approach has limitations. First of all, when constructing NFHP visual image feature models with different detail granularity fusion, the same network structure and model parameters are used, and adjusting the model parameters according to different detail granularity characteristics may better adapt to the detection of malicious stream samples under this granularity. Secondly, the data set used in this study contains a large number of attack class samples. Collecting more samples from different categories and trying to use different data sets for mixed training and detection can obtain experimental data and further optimize our method. Finally, while the construction of NFHP features helps reduce the computational load to some extent, it increases the amount of computation required for data preprocessing. Improving processing performance through methods such as parallel computing is an area that needs to be considered in the future.

Acknowledgement: None.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Nos. U19A2081; 62202320), the Fundamental Research Funds for the Central Universities (No. SCU2023D008), the Science and Engineering Connotation Development Project of Sichuan University (No. 2020SCUNG129) and the Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education.

Author Contributions: Study conception and design: Tao Yi, Xingshu Chen; data collection: Tao Yi; analysis and interpretation of results: Mingdong Yang, Qindong Li, Yi Zhu; draft manuscript preparation: Tao Yi. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The CICIDS-2017, CICIDS-2018, CIC-ToN-IoT, CIC-BoT-IoT, NF-UNSW-NB15-v2, NF-ToN-IoT-v2 datasets, which were utilized to back up the study's conclusions, are respectively available at: CICIDS-2017: https://www.unb.ca/cic/datasets/ids-2017.html. CICIDS-2018: https://www.unb.ca/cic/datasets/ids-2018.html. CIC-ToN-IoT: https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA13. CIC-BoT-IoT: https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA14. NF-UNSW-NB15-v2: https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA6. NF-ToN-IoT-v2: https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA7.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Laboratory, A. (2022). Revealing the cyber attack arsenal from the depths of "nopen" remote access trojan. https://www.antiy.com/response/20220315.html (accessed on 15/03/2022).
2. Xu, C., Shen, J., Du, X. (2020). A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Transactions on Information Forensics and Security, 15,* 3540–3552.
3. Snell, J., Swersky, K., Zemel, R. S. (2017). Prototypical networks for few-shot learning. https://doi.org/10.48550/arXiv.1703.05175
4. Xiang, J., Havaei, M., Chartrand, G., Chouaib, H., Vincent, T. (2018). On the importance of attention in meta-learning for few-shot text classification. https://doi.org/10.48550/arXiv.1806.00852
5. Yu, M., Guo, X., Yi, J., Chang, S., Zhou, B. (2018). Diverse few-shot text classification with multiple metrics. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, vol. 1, pp. 1206–1215.
6. Zamir, A. R., Sax, A., Shen, W., Guibas, L., Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722. Piscataway, NJ, USA, IEEE Press.
7. Rong, W., Zhang, B., Lv, X. (2019). Malicious web request detection using character-level CNN. *ML4CS 2019: Machine Learning for Cyber Security*, pp. 6–16. Xi'an, China, Springer International Publishing.
8. Ding, H., Wan, L., Zhou, K., Long, T., Xin, Z. (2019). Ntrusion detection research based on deep convolutional neural networks. *Computer Science, 46(10),* 173–179.
9. Long, C., Xiao, J., Wei, J., Zhao, J., Wan, W. et al. (2022). Autoencoder ensembles for network intrusion detection. *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pp. 323–333. PyeongChang Kwangwoon_Do, Korea. https://doi.org/10.23919/ICACT53585.2022.9728934

10. Christopher, N., Mohamed, S., Mohamed, H. (2020). Autoencoders: A low cost anomaly detection method for computer network data streams. *Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing (ICCBDC '20)*, pp. 58–62. New York, NY, USA, Association for Computing Machinery. https://doi.org/10.1145/3416921.3416937
11. Lu, J., Meng, H., Li, W., Liu, Y., Guo, Y. et al. (2021). Network intrusion detection based on contractive sparse stacked denoising autoencoder. *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6. Chengdu, China. https://doi.org/10.1109/BMSB53066.2021.9547087
12. Deng, H., Yang, T. (2021). Network intrusion detection based on sparse autoencoder and IGA-BP network. *Wireless Communications and Mobile Computing, 2021,* 9510858.
13. Gharib, M., Mohammadi, B., Hejareh Dastgerdi, S., Sabokrou, M. (2019). AutoIDS: Auto-encoder based method for intrusion detection system. https://doi.org/10.48550/arXiv.1911.03306
14. Dutta, V., Choraś, M., Pawlicki, M., Kozik, R. (2020). A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors, 20(16),* 4583.
15. Dai, J., Xu, X., Gao, H., Xiao, F. (2023). CMFTC: Cross modality fusion efficient multitask encrypt traffic classification in IIoT environment. *IEEE Transactions on Network Science and Engineering, 10(6),* 3989–4009.
16. Dalmaz, H., Erdal, E., Ünver, H. M. (2023). A new hybrid approach using GWO and MFO algorithms to detect network attack. *Computer Modeling in Engineering & Sciences, 136(2),* 1277–1314. https://doi.org/10.32604/cmes.2023.025212
17. AlMasri, E., Alkasassbeh, M., Aldweesh, A. (2023). Towards generating a practical SUNBURST attack dataset for network attack detection. *Computer Systems Science and Engineering, 47(2),* 2643–2669. https://doi.org/10.32604/csse.2023.040626
18. Chen, D., Zhao, Z., Qin, X., Luo, Y., Liu, A. (2022). MAGLeak: A learning-based side-channel attack for password recognition with multiple sensors in IIoT environment. *IEEE Transactions on Industrial Informatics, 18(1),* 467–476.
19. Yang, Y., Wang, W., Liu, L., Dev, K., Qureshi, N. M. F. (2023). AoI optimization in the UAV-aided traffic monitoring network under attack: A stackelberg game viewpoint. *IEEE Transactions on Intelligent Transportation Systems, 24(1),* 932–945.
20. Han, Z. (2023). Smart optimization solution for channel access attack defense under UAV-aided heterogeneous network. *IEEE Internet Things Journal, 10(21),* 18890–18897.
21. Tang, D., Wang, X., Li, X., Vijayakumar, P., Kumar, N. (2021). AKN-FGD: Adaptive kohonen network basedFine-grained detection of LDoS attack. *IEEE Transactions on Dependable and Secure Computing, 20(1),* 273–287.
22. Marques, R. S., Al-Khateeb, H., Epiphaniou, G., Maple, C. (2022). APIVADS: A novel privacypreserving pivot attack detection scheme based on statistical pattern recognition. *IEEE Transactions on Information Forensics and Security, 17,* 700–715. https://doi.org/10.1109/TIFS.2022.3146076
23. Demianiuk, V., Gorinsky, S., Nikolenko, S. I., Kogan, K. (2021). Robust distributed monitoring of traffic flows. *IEEE/ACM Transactions on Networking, 29(1),* 275–288. https://doi.org/10.1109/TNET.2020.3034890
24. Ma, H., Xie, Y., Tang, S., Hu, J., Liu, X. (2020). Threat-event detection for distributed networks based on spatiotemporal Markov random field. *IEEE Transactions on Dependable and Secure Computing, 19(3),* 1735–1752.
25. Cai, S., Xu, H., Liu, M., Chen, Z., Zhang, G. (2024). A malicious network traffic detection model based on bidirectional temporal convolutional network with multi-head self-attention mechanism. *Computer Security, 136,* 103580.

26. Dodia, P., AlSabah, M., Alrawi, O., Wang, T. (2022). Exposing the rat in the tunnel: Using traffic analysis for tor-based malware detection. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*, pp. 875–892. Los Angeles, CA, USA. https://doi.org/10.1145/3548606.3560604
27. Lee, Y. R., Park, N. E., Kim, S. Y., Lee, I. G. (2023). Malicious traffic compression and classification technique for secure internet of things. *Computers, Materials & Continua, 76(3),* 3465–3482. https://doi.org/10.32604/cmc.2023.041196
28. Sharma, S., Saraswat, M., Dubey, A. K. (2022). Fake news detection on Twitter. *International Journal of Web Information Systems, 18,* 388–412.
29. Barbosa, L. (2019). Learning representations of web entities for entity resolution. *International Journal of Web Information Systems, 15(3),* 346–358. https://doi.org/10.1108/ijwis-07-2018-0059
30. Tikekar, P. C., Sherekar, S. S., Thakre, V. M. (2021). Features representation of botnet detection using machine learning approaches. *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*, pp. 1–5. Nagpur, India. https://doi.org/10.1109/ICCICA52458.2021.9697320
31. Jiang, J., Yin, Q., Shi, Z., Wang, Q., Zhou, W. (2019). A new hybrid approach for C&C channel detection. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 583–590. Zhangjiajie, China. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00090
32. Wang, W., Shang, Y., He, Y., Li, Y., Liu, J. (2020). BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Information Sciences, 511,* 284–296.
33. Li, Z., Qin, Z., Shen, P., Jiang, L. (2019). Zero-shot learning for intrusion detection via attribute representation. *ICONIP 2019: Neural Information Processing*, pp. 352–364. Sydney, NSW, Australia, Springer International Publishing.
34. Ouyang, Y., Li, B., Kong, Q., Song, H., Li, T. (2021). FS-IDS: A novel few-shot learning based intrusion detection system for SCADA networks. *IEEE International Conference on Communications (ICC)*, pp. 1–6. Electr Network, IEEE.
35. Zerhoudi, S., Granitzer, M., Garchery, M. (2020). Improving intrusion detection systems using zero-shot recognition via graph embeddings. *44th Annual IEEE-Computer-Society International Conference on Computers, Software, and Applications (COMPSAC)*, pp. 790–797. Montreal, QC, Canada, IEEE.
36. Zhong, Y., Gao, Z., Li, R., Que, C., Yang, X. et al. (2021). STRAD: Network intrusion detection algorithm based on zero-positive learning in real complex network environment. *2021 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–8. Athens, Greece, IEEE.
37. Zhu, S., Xu, X., Gao, H., Xiao, F. (2023). CMTSNN: A deep learning model for multi-classification of abnormal and encrypted traffic of internet of things. *IEEE Internet of Things Journal, 10(13),* 11773–11791.
38. Singh, M. N., Khaiyum, S. (2021). Enhanced data stream classification by optimized weight updated meta-learning: Continuous learning-based on concept-drift. *International Journal of Web Information Systems, 17(6),* 645–668.
39. Lan, J., Liu, X., Li, B., Sun, J., Li, B. et al. (2022). MEMBER: A multi-task learning model with hybrid deep features for network intrusion detection. *Computers & Security, 123,* 102919. https://doi.org/10.1016/j.cose.2022.102919
40. Agrafiotis, G., Makri, E., Flionis, I., Lalas, A., Votis, K. et al. (2022). Image-based neural network models for malware traffic classification using PCAP to picture conversion. *2022 17th International Conference on Availability, Reliability and Security (ARES)*, pp. 1–7. Vienna, Austria. https://doi.org/10.1145/3538969.3544473