Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A subspace-based few-shot intrusion detection system for the Internet of Things*#◇

Zhihui LI[1,2], Congyuan XU†‡[2], Kun DENG[2], Chunyuan LIU[2]

[1]*School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310027, China*

[2]*College of Information Science and Engineering, Jiaxing University, Jiaxing 314000, China*

†E-mail: cyxu@zjxu.edu.cn

Received July 2, 2024; Revision accepted Dec. 9, 2024; Crosschecked May 9, 2025

Abstract: Deep learning-based intrusion detection systems rely on numerous training samples to achieve satisfactory detection rates. However, in the real-world Internet of Things (IoT) environments, the diversity of IoT devices and the subsequent fragmentation of attack types result in a limited number of training samples, which urgently requires researchers to develop few-shot intrusion detection systems. In this study, we propose a subspace-based approach for few-shot IoT intrusion detection systems to cope with the dilemma of insufficient learnable samples. The method is based on the principle of classifying metrics to identify network traffic. After feature extraction of samples, a subspace is constructed for each category. Next, the distance between the query samples and the subspace is calculated by the metric module, thus detecting malicious samples. Subsequently, based on the CICIoT2023 dataset we construct a few-shot IoT intrusion detection dataset and evaluate the proposed method. For the detection of unknown categories, the detection accuracy is 93.52% in the 5-way 1-shot setting, 92.99% in the 5-way 5-shot setting, and 93.65% in the 5-way 10-shot setting.

Key words: Intrusion detection system; Few-shot learning; Internet of Things; Subspace
https://doi.org/10.1631/FITEE.2400556 **CLC number:** TP393

1 Introduction

The Internet of Things (IoT) is rapidly becoming a central aspect of our daily lives and industrial operations. As an increasing number of devices connect to networks, the amount of data they generate and the range of applications they can use expand significantly. However, this connectivity poses new security challenges because the diverse and distributed nature of IoT devices makes them new targets for cyber attacks. Thus, the IoT intrusion detection system has become the key to securing the IoT. In recent years, excellent algorithms for IoT intrusion detection systems have been developed, and they are effective in detecting abnormal traffic (Lu HM et al., 2022; He et al., 2024b). These studies have one thing in common: they train models from large datasets. Traditional intrusion detection methods based on a large number of samples face the challenge of data collection and labeling owing to the diversity of IoT devices and the dynamics of the network environment. In a diverse IoT environment, a discontinuity occurs in security measures due to device diversity, differences in operating system versions, inconsistent firmware updates, and other factors. This fragmentation makes it extremely difficult to maintain a

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62302197) and the Zhejiang Provincial Natural Science Foundation (No. LQ23F020006)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2400556) contains supplementary materials, which are available to authorized users

◇ Special Topic: The 27th Annual Meeting of the China Association for Science and Technology

ORCID: Congyuan XU, https://orcid.org/0009-0003-7760-5980

unified security policy, thereby providing opportunities for attackers to exploit. Furthermore, when faced with zero-day attacks, researchers usually fail to collect sufficient samples and therefore may be unable to build timely datasets. At this point, machine learning-based IoT intrusion detection systems can fail, which in turn can cause irreparable damage to industrial production (Yan et al., 2024).

Researchers have shifted their focus to few-shot learning in the face of very few attack samples, such as zero-day attacks. As the name implies, IoT intrusion detection systems based on few-shot learning aim to develop efficient and accurate intrusion detection models for sparse and unbalanced data. However, the development of efficient few-shot models faces challenges, such as poor generalization and overfitting (Duan RX et al., 2021; Wang ZM et al., 2021).

Research on IoT intrusion detection systems based on few-shot learning is still in its infancy, and more research needs to be made available for reference. However, the feasibility of using few-shot learning to address IoT intrusion detection has already been demonstrated.

In this study, a subspace-based few-shot IoT intrusion detection system is developed to solve these problems. We apply a subspace classifier to an IoT intrusion detection system. Unlike existing parameter optimization-based methods, we use metric learning to detect network traffic, which uses prior knowledge to detect unknown categories (Simon et al., 2020).

The main contributions of this study are as follows:

1. To enhance the discriminability of traffic information from different classes, we propose a few-shot classifier that uses subspaces as a metric to fully learn the common representation of each traffic flow, while designing a four-layer feature extraction network based on channel attention mechanisms to enhance the feature representation of each traffic flow.

2. We demonstrate that the proposed method performs well and has the ability to generalize unknown categories on the CICIoT2023, CICIDS2017, and CICIDS2018 datasets, which could address the capability of detecting unknown category samples under few-shot conditions.

3. We consider the diversity of IoT devices and the specificity of attack patterns, along with the designed data preprocessing steps, and construct a dataset suitable for few-shot intrusion detection.

2 Related works

2.1 IoT intrusion detection

The large number, diverse variety, and wide distribution of IoT devices pose massive challenges to security protection. The traditional method of network security protection may be unable to effectively respond to intrusions in the IoT environments. Therefore, the detection of IoT intrusions is an important research topic (Alani and Awad, 2023; Mehedi et al., 2023).

Traditional machine learning-based intrusion detection systems have difficulties in heterogeneous data preprocessing and fusion training, and they can no longer satisfy the requirements for heterogeneous IoT intrusion detection. Makkar et al. (2021) proposed a machine learning-based framework for detecting spam messages, aiming at addressing the increasingly severe problem of spam in IoT devices. By evaluating five different machine learning models, the framework is capable of calculating the spam score of devices and assessing their reliability based on that. Chen et al. (2023) proposed a deep learning method based on word embedding to solve this problem using the multisource data fusion method of natural language processing, which can effectively preserve data features. Lu HM et al. (2022) established a new model called CMAE that used memory modules to record potential spatial feature representations of normal data and to obtain efficient memory modules with feature reconstruction loss and feature sparsity loss. When the proposed components were used, the model achieved effective improvements in the three metrics of intrusion detection. The experimental outcomes indicated that the employment of feature sparsity loss, as opposed to feature reconstruction loss, achieved enhancements of approximately 0.154%, 0.150%, and 0.160% in precision, recall, and F1-score, respectively. To address the limited computing power of IoT devices, He et al. (2024b) developed a lightweight and efficient intrusion detection method based on feature grouping, which achieved a classification accuracy of over 99.5% on the BotIoT, MedBIoT, and MQTTIoTIDS2020 datasets. Thakkar and Lohiya (2023)

used ensemble learning (bagging classifier) to address the class imbalance problem in a dataset, and the proposed method achieved a precision of 99.77% and a recall rate of 99.96% on the CICIDS2017 dataset. The above studies all refer to malicious detection methods tailored for the IoT environment. There are also many excellent algorithms for research in other network environments. For example, He et al. (2024a) proposed a method based on packet clustering and online data stream processing for real-time detection of distributed denial-of-service (DDoS) attacks. This method achieved an accuracy of 99.86% with only a latency of 1.05 s, surpassing the response speeds of other methods. Niu et al. (2023) combined semi-supervised learning and active learning techniques to achieve an efficient detection model with a limited labeling budget. Fouladi et al. (2022) proposed a method based on discrete wavelet transform (DWT) and autoencoder neural networks. Their research not only illustrates how deep learning can be effectively employed for attack identification with small sample sizes but also validates the effectiveness of their approach across various attack types through experiments. Duan GH et al. (2023) introduced an intrusion detection method based on dynamic line graph neural networks (DLGNNs), which efficiently uses the spatiotemporal characteristics of network traffic within a semi-supervised learning framework. By transforming network traffic into a series of spatiotemporal graphs, the model not only captures the evolving context of communication between each Internet protocol (IP) pair in the network but also strengthens the message aggregation capability of graph convolutions.

IoT intrusion detection has been investigated extensively, but further research is required for few-shot IoT intrusion detection systems. Therefore, we propose an approach that accurately identifies samples of known categories and samples of unknown categories using prior knowledge.

2.2 Few-shot learning

Research on few-shot learning has made significant progress in recent years, particularly with deep learning models. Few-shot learning can be broadly divided into three categories: parameter-based optimization, metric-based learning, and data-based enhancement.

One of the most impressive studies on parameter optimization-based few-shot algorithms is the model-agnostic meta-learning (MAML) approach, proposed by Finn et al. (2017). This approach enables improved generalization of the model to new tasks by sharing the learned parameters between tasks. This task is accomplished using two nested gradient descent processes. Jamal and Qi (2019) argued that the initial model of a meta-learner may be biased toward certain tasks sampled during the meta-training phase, and they proposed task-agnostic meta-learning (TAML) to avoid learning-biased initialized models. Nichol et al. (2018) proposed the reptile algorithm, which is computationally simpler than the MAML algorithm. The former was used to update the network model parameters with the difference between the most recent parameters and the meta-parameters after K training sessions were executed for each task. The latter performed all tasks and updated the network model parameters with an average loss in the test set of each task. The core principle of parameter optimization-based few-shot algorithms is to accelerate network learning by optimizing the parameters of the model or learning algorithm.

Data augmentation-based few-shot learning algorithms use the original data to generate new data to expand the dataset. Wang YX et al. (2018) used image synthesizers to augment the original dataset; specifically, they inputted the image synthesizer into a network with a feature extraction network and a classifier for end-to-end training. The classification loss was then used to guide the training of the image synthesizer so that its outputs were suitable for classification purposes. Schwartz et al. (2018) used an autoencoder to identify deformations between different samples of the same category, used the deformations to generate new samples for other categories, and finally trained the classifier using an expanded dataset. Wang YK et al. (2020) further investigated the data augmentation-based few-shot algorithm using the instance confidence inference module, which was used to select samples and pseudo-labels with higher confidence and expand the support set. For comparison, samples with lower confidence were used to update the unlabeled dataset.

Most of the metric-based few-shot learning algorithms begin by extracting the sample features through a feature extraction network and metricize them through a metric module (Ouyang et al.,

2021). Many excellent algorithms for small-shot learning have emerged in this direction, such as the Siamese network (Koch et al., 2015), matching network (Vinyals et al., 2016), prototypical network (Snell et al., 2017), and relation network (Sung et al., 2018). In this study, we design an IoT intrusion detection system using a few-shot algorithm based on metric learning.

2.3 Few-shot network intrusion detection

In the field of intrusion detection systems, most of the previous studies were based on parameter optimization and metric learning. We focus on investigating these two approaches in intrusion detection systems.

Xu CY et al. (2020) proposed FC-Net, which applies the metric concept of a relation network. This method is divided into two parts: a feature extraction network and a comparison network. FC-Net learns a pair of feature maps for classification from a pair of network traffic samples and compares the feature maps to determine whether the pair belongs to the same class. The authors evaluated the method on the ISCX2012FS and CICIDS2017FS datasets, which was trained and tested on the same datasets. An average accuracy of 98.88% and an average detection rate of 99.62% were obtained for the untrained dataset. Sun et al. (2023) proposed an attention-based intrusion detection system for prototype capsule networks, which was divided into two parts: a temporal spatial feature fusion method using capsules for feature extraction and a prototype network classification method with attention and voting mechanisms. The performance of the proposed method was evaluated using the CSECICIDS2018 dataset. Feng et al. (2021) proposed a class-adaptive anomaly-detection framework based on MAML, called FCAD. During the data preprocessing phase, FCAD extracts statistical features through a feature extractor and selector, and temporal sequence features through a long short-term memory-based autoencoder. It then fuses the obtained statistical and sequence features and constructs a meta-learning task. The experimental results showed that the accuracy of the method was 95.1% for the CICIDS2017 dataset. Lu CM et al. (2023) were the first to apply MAML to few-shot IoT intrusion detection. They constructed a few-shot IoT dataset called FSIDS-IoT based on a benchmark dataset and evaluated the proposed method. For test against unknown attacks, the best accuracy achieved in the 5-way 10-shot setting was 92.19%. Shi et al. (2023) applied the MAML algorithm to network intrusion detection. Unlike previous studies, Shi et al. (2023) introduced a learn-to-forget (L2F) decay mechanism to solve the problem of conflicts between tasks caused by the forced sharing of initialization of MAML, which prevented the model from quickly converging to the optimal position. For the CICIDS2017 dataset, the accuracy reached 94.66% at K=10.

Most of these studies did not extend to the IoT domain; they only applied a few-shot learning algorithm to the IoT intrusion detection domain (Lu CM et al., 2023). Similarly, our work considers the complexity of the IoT; we evaluate our model on the latest IoT dataset, CICIoT2023. The proposed model can maintain excellent performance in accurately identifying known and unknown attack categories under few-shot conditions.

3 Methodology

Few-shot learning mimics the human ability to rapidly acquire knowledge from limited examples. For example, a child may need only to see a few examples to recognize what a dog is without needing to see hundreds or thousands of dogs of different breeds. This ability partly stems from our brain's efficient processing of pattern recognition and our foundational prior knowledge of the world. In artificial intelligence, we attempt to simulate this ability through algorithms. Few-shot learning algorithms are designed to handle situations where data are scarce; they need to be able to extract critical information from a small number of samples and generalize new, unseen situations. Precisely for this reason, we draw on the capabilities of few-shot learning to detect unknown category attacks.

We consider the few-shot learning problem in two stages: feature extractor and classifier. Let $f_{\boldsymbol{\theta}}: \boldsymbol{X} \to \mathbb{R}^D$ be the mapping from the input space $\boldsymbol{X}$ to the D-dimensional representation implemented by a neural network, and let $\mathcal{X}_c = \{x_{c,1}, x_{c,2}, ..., x_{c,K}\}$ be the class-specific set. We formulate the few-shot learning problem as creating a classifier. To this end, the last layer of the neural network is implemented

along with the softmax layer:

$$p(c|\boldsymbol{q}_l) = \frac{\exp(\boldsymbol{W}_c^{\mathrm{T}} f_{\boldsymbol{\theta}}(\boldsymbol{q}_l))}{\sum_{c'} \exp(\boldsymbol{W}_c^{\mathrm{T}}, f_{\boldsymbol{\theta}}(\boldsymbol{q}_l))} = \frac{\exp(d_c(\boldsymbol{q}_l))}{\sum_{c'} \exp(d_{c'}(\boldsymbol{q}_l))}, \tag{1}$$

where $\boldsymbol{W}_c$ is the weight of class c, $f_{\boldsymbol{\theta}}$ is the feature extractor, c' represents the set of categories in a task, and $d_c(\boldsymbol{q}_l)$ is the distance between query sample $\boldsymbol{q}_l$ and class c. Thus, the problem of few-shot learning can be understood as providing a new task on how to generate $\boldsymbol{W}$.

This section introduces a few-shot intrusion detection method based on a subspace (Simon et al., 2020). The proposed method is divided into three main phases: task construction, feature extraction, and network traffic classification. Fig. 1 shows the general block diagram of the proposed method. During the task construction phase, the raw data are first normalized to conform to the input format of the neural network. Next, n tasks are drawn for model training, in which each task contains a support set and a query set. The samples in the support set are used to compute the subspaces of each category, and the samples in the query set are used to measure the distances to the subspaces of each category. During the feature extraction phase, a feature extraction network with an efficient channel attention (ECA) mechanism is designed to extract the features of each sample, focusing on unique sample features. Finally, during the network traffic classification phase, network traffic is categorized using the metric module.

It must be clarified that our traffic samples consist of feature sequences composed of 46 features, such as flow duration, header length, protocol type, and the number of Finish (FIN) flag bits. After data preprocessing, these feature sequences are fed into the neural network for feature extraction, followed by the construction of subspaces for each category, ultimately achieving the detection of different types of traffic.

3.1 Task construction

The CICIoT2023 dataset is available in two file formats: PCAP and CSV. The PCAP files store rich raw network traffic which contains all sent packets, and can be used to extract and design other features. The CSV file offers a simpler method for loading and using data, which can be processed and obtained through certain Python libraries or software. In previous studies, researchers focused on investigating the original network traffic samples; this is because the original network traffic contains richer features. The model can thoroughly learn the differences between the traffic samples and quickly identify them more accurately. However, in this study, we choose to investigate the CSV dataset, which is a simple tabular data storage format, in which rows represent samples and columns represent features of the samples. The sample features in the CICIoT2023 dataset include timestamps, protocol type, header length, and packet transmission rate in the data stream.

To make the data suitable for training in a convolutional neural network, we perform preprocessing on the raw network traffic. The preprocessing in this paper is divided into feature extraction, data normalization, and type conversion.

CICFlowMeter (Draper-Gil et al., 2016), a feature extraction tool, is often used to extract features from raw network traffic. Therefore, in this study, we use CICFlowMeter to extract features from network traffic. First, the traffic stored in PCAP packets is inputted into CICFlowMeter, which extracts some statistical features at the transport layer. For intrusion detection, not all features benefit the model's training, and some key features can be extracted. We refer to the official CSV file provided by the CICIoT2023 dataset and ultimately extract 46 features for each network traffic sample. After feature extraction, we find that some samples in the CSV file would have feature values that are infinite or missing, which would result in inconsistent feature sequence lengths for each sample. Therefore, we choose to replace infinite values and missing spots with zeros to ensure that each sample has a feature sequence of the same length. The above operations are illustrated in the following formula:

$$\mathrm{CSV}=\mathrm{Sam}(\mathrm{CFM}(\mathrm{PCAP})). \tag{2}$$

In this context, CFM stands for CICFlowMeter, and Sam represents the process of filtering the sample features from the CSV file.

Another crucial preprocessing step is data normalization. In our dataset, the numerical range of network traffic features varies greatly. For example, packet sizes can range from a few to several thousand bytes, while other features might only be

1 or 0. Without normalization, the model is likely to be biased toward learning features with larger values, neglecting those minor but important features. Therefore, we should normalize the data before converting them to a tensor type. The normalization process is shown in Eq. (3):

$$X_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{3}$$

where $x_{\max}$ and $x_{\min}$ represent the maximum and minimum values in the dataset, respectively. After normalization, all feature values are scaled to the range between 0 and 1. This process helps improve the model's convergence speed and performance.

After data preprocessing, we begin by defining the terminology used in few-shot learning. Currently, most few-shot learning trains models in an episode learning paradigm, in which an episode, $\mathcal{T}_i$, consists of two sets: the support set and the query set. This learning paradigm describes how the model can improve its capabilities with the fragmented data in each iteration. Specifically, deep embedding involves learning using limited labels and data. This learning paradigm is called N-way K-shot categorization, indicating that only N learnable categories exist for each task with K samples in each category. We return to the notation of N-way K-shot few-shot learning. Each episode or task $\mathcal{T}_i$ consists of the support set $S=\{(x_{1,1}, c_{1,1}), (x_{1,2}, c_{1,2}), ..., (x_{N,K}, c_{N,K})\}$ and the query set $Q = \{q_1, q_2, ..., q_{N\times M}\}$, where $x_{i,j}$ is the j^{th} sample of the i^{th} category and $c_{i,j} \in\{1, 2, ..., N\}$ represents the label of the j^{th} sample in the i^{th} category. Algorithm 1 describes the steps involved in task construction.

3.2 Feature extraction

Because the dataset is in a CSV file format, where each row represents a sample and each column represents the sample features, a 1D convolutional neural network (CNN) is used to map the features of the network traffic. Since a few features are used for each sample, with only 46 features, we need to focus on more discriminative features. Therefore,

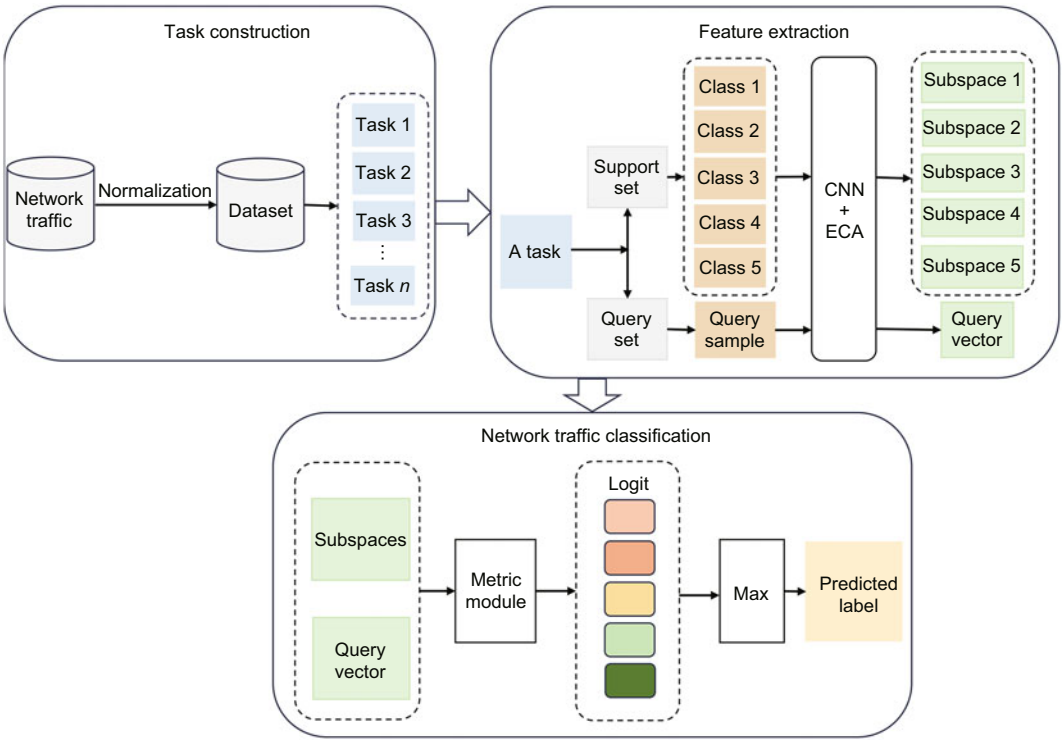

Fig. 1 Diagram of subspace-based few-shot IoT intrusion detection structure. The proposed method is divided into three main modules: task construction, feature extraction, and network traffic classification. During the task construction phase, the main focus is creating N-way K-shot formatted few-shot tasks to facilitate model learning. In the feature extraction module, we design a four-layer structure with an ECA mechanism. Finally, in the network traffic classification module, classification is performed by measuring the distance between the query vector and the subspace of each category

Algorithm 1 Generating a few-shot task

Input: label set $\mathcal{L} = \{1, 2, ..., W\}$, dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_W, y_W)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^D$, $y_i \in \mathcal{L}$
Output: few-shot task $\mathcal{T} = \{S_k, Q_k\}$

```
Initialize T ← ∅
while not done do
    V ← ({1, 2, ..., W}, N)
    for k ∈ {1, 2, ..., N} do
        S_{k,i} ← RandSample(D_k, N_S)
        Q_{k,i} ← RandSample(D_k \ S_k, N_Q)
    end for
    T_i ← {S_{k,i}, Q_{k,i}}
    T ← T ∪ T_i
end while
```

we propose to embed an attention module into a 1D CNN. The specific structure is shown in Fig. 2.

We design a four-layer 1D CNN. After each convolutional block, we add an ECA module to ensure that unique features are attended each time the computation is performed. Each block consists of a 1D convolutional layer with a kernel size of 3, a batch normalization layer, a ReLU layer, and a max-pooling layer. The first three blocks have the same design, and in the last block, a dropout layer is added to minimize the risk of overfitting.

An ECA mechanism is a local cross-channel interaction strategy without dimensionality reduction that reduces the complexity of the model while maintaining its performance (Wang QL et al., 2020). This strategy allows the adaptive selection of a 1D convolutional kernel size to ensure the coverage of local cross-channel interactions.

The structure of the ECA mechanism is shown in Fig. 3. C, H, and W denote the number of channels, the height, and the width, respectively. GAP denotes global average pooling, and σ is the sigmoid activation function. The aggregated features are first obtained by GAP. Next, the cross-channel interaction is achieved via 1D convolution, and finally the weight assignment of the channel features is achieved using the sigmoid function. The adaptive kernel size for the 1D convolution is expressed in Eq. (4):

$$K = \Psi(C) = \left| \frac{\log_2 C}{\gamma} + \frac{b}{\gamma} \right|_{\text{odd}}, \tag{4}$$

where $|t|_{\text{odd}}$ is the closest odd number to t, and γ and b are customized parameters, set to 2 and 1 in this study, respectively.

3.3 Network traffic classification

The core principle of an intrusion detection system is to analyze features and classify network packets by constructing a deep learning-based discriminative model. Therefore, we design a subspace-based few-shot intrusion detection system. Below, we describe how to create subspaces and network traffic using these subspaces.

Given an input task (N-way K-shot), the

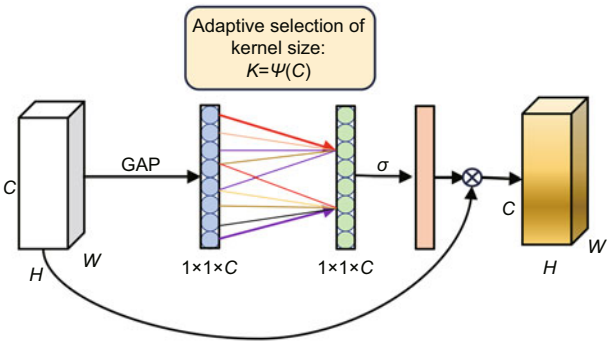

Fig. 3 ECA module

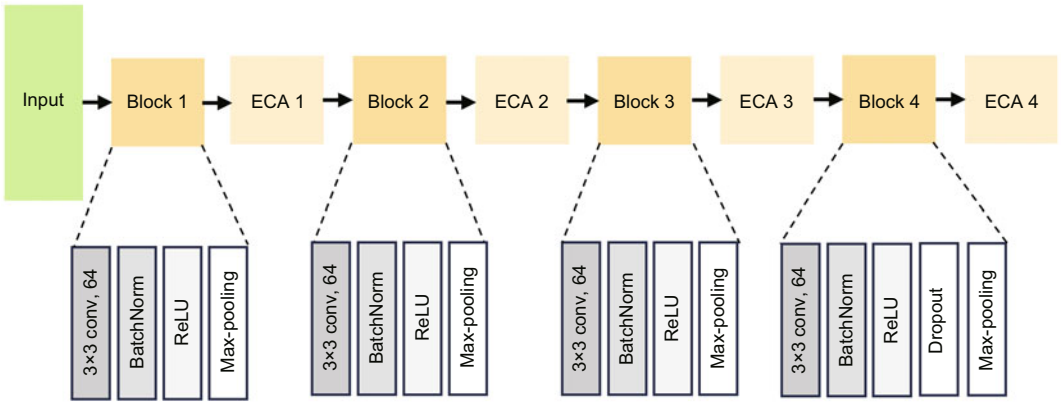

Fig. 2 CNN with ECA mechanism

feature embedding representations for the support set flow and the query set flow can be obtained:

$$\boldsymbol{F}_{c,i}^{\mathrm{sup}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_{c,i}), \tag{5}$$

$$\boldsymbol{F}_{l}^{\mathrm{que}} = f_{\boldsymbol{\theta}}(\boldsymbol{q}_{l}), \tag{6}$$

where $f_{\boldsymbol{\theta}}(\cdot)$ represents the feature extraction network mentioned above, which is primarily composed of attention layers and convolutional layers, with network parameters denoted as $\boldsymbol{\theta}$. $\boldsymbol{F}_{c,i}^{\mathrm{sup}}$ denotes the feature embedding of the i^{th} support set traffic from class c, and $\boldsymbol{F}_{l}^{\mathrm{que}}$ denotes the feature embedding of the l^{th} query set traffic.

Afterward, by learning feature representations from the support set sample traffic of each category in the given task, the corresponding subspace can be calculated, thereby constructing a subspace classifier. For instance, when considering the construction of the subspace for class c, we will provide a detailed description of the construction process.

First, we calculate the prototype of class c according to Eq. (7), with each class in the task corresponding to a prototype:

$$\boldsymbol{\mu}_{c} = \frac{1}{K}\sum_{i=1}^{K}\boldsymbol{F}_{c,i}^{\mathrm{sup}}, \tag{7}$$

where K represents the number of samples in class c of the support set, and $\boldsymbol{\mu}_c$ represents the prototype of class c. Based on the feature embeddings and the prototype representation of class c's samples, we reconstruct the feature space as follows:

$$\boldsymbol{X}_{c} = [\boldsymbol{F}_{c,1}^{\mathrm{sup}} - \boldsymbol{\mu}_{c}, \boldsymbol{F}_{c,2}^{\mathrm{sup}} - \boldsymbol{\mu}_{c}, ..., \boldsymbol{F}_{c,K}^{\mathrm{sup}} - \boldsymbol{\mu}_{c}], \tag{8}$$

where $\boldsymbol{F}_{c,i}^{\mathrm{sup}}$ $(i = 1, 2, \cdots, K)$ represents the feature embedding of the i^{th} traffic flow from class c.

Then, performing singular value decomposition (SVD) on $\boldsymbol{X}_c$ yields a matrix $\boldsymbol{B}_c$ with orthogonal basis properties (i.e., $\boldsymbol{B}_c\boldsymbol{B}_c^{\mathrm{T}}$). By truncating $\boldsymbol{B}_c$, we can obtain the feature subspace $\boldsymbol{P}_c$ for class c. Subsequently, the feature subspaces for other classes in the given task are calculated in the same manner. Ultimately, we can classify the traffic based on the projection distance of the query sample to each class's subspace.

Just as with reconstructing the feature space of the support set samples, we also need to use the obtained prototype of class c ($\boldsymbol{\mu}_c$) to construct the feature space for each query sample $\boldsymbol{q}_l$, which is $\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c}$. Therefore, the projection of $\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c}$ onto the feature subspace of class c is $\boldsymbol{M}_c(\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c})$, where $\boldsymbol{M}_c = \boldsymbol{P}_c\boldsymbol{P}_c^{\mathrm{T}}$. Thus, according to the Pythagorean theorem, the vector perpendicular to the corresponding subspace can be expressed as $(\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c}) - \boldsymbol{M}_c(\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c}) = (\boldsymbol{I} - \boldsymbol{M}_c)(\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c})$, where $\boldsymbol{I}$ denotes the identity matrix. Finally, the projection distance of the query sample $\boldsymbol{q}_l$ to the feature subspace of class c is

$$d_c(\boldsymbol{q}_l) = -||(\boldsymbol{I} - \boldsymbol{M}_c)(\boldsymbol{F}_{l}^{\mathrm{que}} - \boldsymbol{\mu}_{c})||^2. \tag{9}$$

We calculate the probability that a query sample is assigned to class c using the softmax function:

$$p(c, \boldsymbol{q}_l) = p(c|\boldsymbol{q}_l) = \frac{\exp(d_c(\boldsymbol{q}_l))}{\sum_{c'}\exp(d_{c'}(\boldsymbol{q}_l))}. \tag{10}$$

Finally, the query sample can be assigned to the category with the highest prediction probability.

3.4 Loss function

To make the generated subspaces more discriminative, that is, to increase the distance between subspaces of different categories, we use Grassmann geometry to calculate the distances between subspaces of each category, precisely representing the class differences among different subspaces. Given two subspaces $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$, the distance metric is defined as follows:

$$\begin{aligned}\sigma_p^2(\boldsymbol{P}_i, \boldsymbol{P}_j) &= \|\boldsymbol{P}_i\boldsymbol{P}_i^{\mathrm{T}} - \boldsymbol{P}_j\boldsymbol{P}_j^{\mathrm{T}}\|_{\mathrm{F}}^2 \\ &= 2D_{\mathrm{s}} - 2\|\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{P}_j\|_{\mathrm{F}}^2,\end{aligned} \tag{11}$$

where D_{s} represents the subspace dimension and $\|\cdot\|_{\mathrm{F}}^2$ represents the Frobenius norm.

Eq. (11) shows that maximizing the distance between two subspaces is achieved by minimizing $\|\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{P}_j\|_{\mathrm{F}}^2$. Therefore, we introduce it into the loss function, which can be minimized through a constant iteration of the model:

$$\mathcal{L}_{\mathrm{t}} = -\frac{1}{NK}\sum_{c}\ln(p_{c,q}) + \lambda\sum_{i\neq j}\|\boldsymbol{P}_i^{\mathrm{T}}\boldsymbol{P}_j\|_{\mathrm{F}}^2, \tag{12}$$

where the first term on the right-hand side is the cross-entropy classification loss function, the second term is the subspace loss function, N is the number of classes in the support set, and λ is the weight coefficient. Algorithm 2 describes the training process of the subspace classifier.

Algorithm 2 Training a subspace classifier

Input: task $\mathcal{T}$
$\boldsymbol{\theta} \leftarrow$ random initialization
for $t \in \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_i\}$ **do**
  for $c \in \{1, 2, ..., N\}$ **do**
    Calculate matrix $\boldsymbol{X}_c$
    Calculate the average of the class
    $[\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^{\mathrm{T}}] \leftarrow \mathrm{SVD}(\boldsymbol{X}_c)$
    $\boldsymbol{P}_c \leftarrow$ Truncate $\boldsymbol{U}$
    for $\boldsymbol{q}_l \in Q$ **do**
      Compute $d_c(\boldsymbol{q}_l)$ using Eq. (9)
    end for
  end for
  Compute the final loss $\mathcal{L}_{\mathrm{t}}$ using Eq. (12)
  Update $\boldsymbol{\theta}$ using $\nabla\mathcal{L}_{\mathrm{t}}$
end for

4 Simulations and analysis

4.1 Datasets

Datasets widely used in network intrusion detection are KDD99, NSLKDD, UNSWNB15, CICIDS2017, and CICIDS2018. In this study, we used CICIoT2023 (Neto et al., 2023), CICIDS2017 (Draper-Gil et al., 2016), and CICIDS2018 to evaluate our model. CICIoT2023 was collected by executing 33 attacks on an IoT topology consisting of 105 devices that mimic the real-world deployment of IoT products and services in a smart home environment. The dataset contains 33 attacks organized into seven categories: DDoS, denial-of-service (DoS), Recon, Web-based attacks, Brute force, Spoofing, and Mirai. The CICIoT2023 dataset contains 169 CSV files and numerous network traffic samples. Regarding the research objectives of this study, validating the practicality of our proposed method on this dataset will have a positive impact, and the simulation results obtained will be convincing. At the same time, we conducted multiple simulations on the CICIDS2017 and CICIDS2018 datasets to verify the proposed method. Table 1 describes the dataset created for few-shot IoT intrusion detection based on the CICIoT2023 dataset, which includes the types of attacks and the number of samples in each category. Information about the CICIDS2017 and CICIDS2018 datasets is given in the supplementary materials.

It should be noted that there is a relationship between the number of samples in the training and test sets and the K-shot setting. The model was trained in a task-sequential manner, with each training iteration processing a single task instance. These tasks were sampled from the training or test sets, each consisting of a support set and a query set. The support set followed the N-way K-shot paradigm, which means that it contains N classes with K samples from each class, resulting in a total of $N \times K$ samples in the support set. In the field of few-shot learning, K is typically 1, 5, or 10. Next, we provide a general overview of the attack methods for each type of malicious traffic in the CICIoT2023 dataset in the supplementary materials.

Table 1 Information of the CICIoT2023 dataset

Attack type	Number of samples	
	Training set	Test set
DDoS	360	240
DoS	120	80
Mirai	90	60
Spoofing	60	40
Recon	150	100
Web	180	120
Brute force	30	20

4.2 Simulation settings

The simulations were conducted on Windows 11 with an NVIDIA RTX4060 graphics card of 8 GB and an Intel Core$^{\mathrm{TM}}$ i7-12650H CPU. The simulation environments used were Python 3.11.5, CUDA 12.0, and PyTorch 2.1.1. Table 2 lists the hyperparameter settings of the algorithm used in this study.

Three sets of simulations were conducted to validate the proposed method:

1. The most basic function of an IoT intrusion system is to recognize malicious traffic from a large amount of traffic, a situation called binary classification. For the 2-way K-shot setting, we conducted simulations. K was set to 5 and 10. We relabeled the training and test sets; normal samples were labeled as Benign, and malicious samples were labeled as Attack.

2. In addition to the binary classification simulation, we set up a multi-classification simulation, denoted as 5-way K-shot, where $K = 1$, 5, and 10. In this simulation, we designed a model that recognizes

Table 2 Hyperparameter settings

Hyperparameter	Value
Number of epochs	200
Learning rate	0.002
γ	0.5
λ	0.03
Batch size	100

malicious traffic and accurately predicts the true attack category.

3. Detection simulations were conducted using unknown traffic categories to simulate the emergence of new attacks in a real-world IoT environment. Specifically, seven types of attack traffic were used, six types as the training dataset, and the remaining type as the test dataset. These operations were performed in 5-way 1-shot, 5-way 5-shot, and 5-way 10-shot settings.

4.3 Metrics

Similar to previous studies, for binary classification simulations, we evaluated our model with accuracy, precision, recall, and F1-score, whereas for multi-classification simulations, we evaluated accuracy and precision.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}, \tag{13}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \tag{14}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \tag{15}$$

$$\text{F1-score} = 2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}. \tag{16}$$

Here, TP represents the number of normal samples which are categorized as normal samples, TN represents the number of malicious samples which are categorized as malicious samples, FP represents the number of normal samples which are categorized as malicious samples, and FN represents the number of malicious samples which are categorized as normal samples.

4.4 Simulation results

The results of Simulation 1 are presented in Fig. 4. The performance was better with K=10 than that with K=5. At K=5, the accuracy, precision, recall, and F1-score were 96.12%, 96.88%, 95.37%, and 96.11% respectively. At K=10, the respective metrics were 97.72%, 98.49%, 96.94%, and 97.71%.

In Simulation 2, we evaluated the model in terms of accuracy and precision of detection for each category. Eight categories were selected for this set of simulations, from which five categories were randomly selected for each task, with an average detection precision of 93.83% in the 5-way 1-shot setting. The average detection precision in the 5-shot setting was 94.63%, and that in the 10-shot setting was 95.34%. The detection precision of this group of simulations is shown in Fig. 5. As the sample size increased, the detection precision of each traffic category increased, and regardless of the setting, the detection precision of Mirai was the highest.

The confusion matrix can be found in the supplementary materials.

In Simulation 3, we used the precision and accuracy as the evaluation indices. The detection of unknown categories under the 5-way 1-shot setting is shown in Fig. 6a. The highest detection precision for each category was 100%, but the overall fluctuation was significant, among which the detection fluctuated the most for Web attacks, and less for the

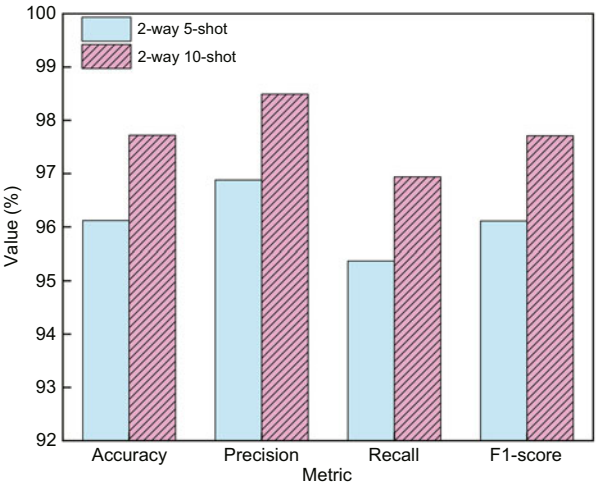

Fig. 4 Comparison of the influence of sample size on the results in a binary classification setting

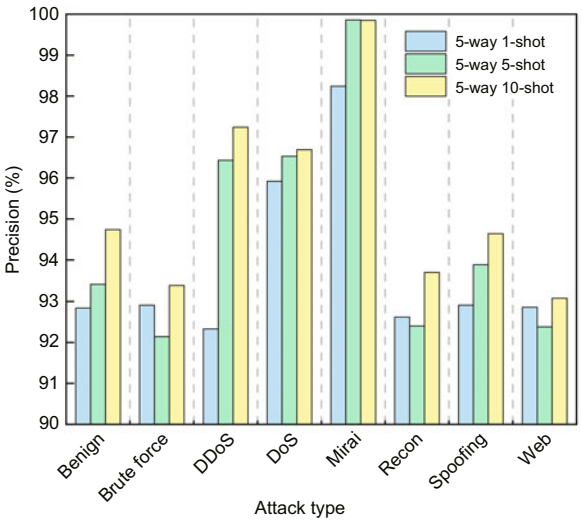

Fig. 5 Comparison of the influence of sample size on the results in a multi-classification setting

other attack types, with an overall detection precision of 93.07%. The detection of unknown categories in a 5-way 5-shot setting is shown in Fig. 6b. The fluctuation was more minor than that of a 5-way 1-shot setting, and the data were more concentrated. The detection precision of the Recon attack was the worst with 86.60%, whereas the detection precision of the Mirai attack was the best with 95.27%. The overall detection precision was 91.99%. The detection of unknown categories in the 5-way 10-shot setting is shown in Fig. 6c. Overall, compared with the previous two settings, the detection effect was the most balanced for each category. The detection precision exceeded 90%, and the detection effect was the worst for the Recon attack, with a precision of 89.17%. The detection effect was the best for the Mirai attack, with a precision of 95.40%. The overall detection precision was 91.59%.

From Simulation 3, we learned that as the number of samples increased, the fluctuation of the model in detection precision decreased; however, the detection precision for a certain category did not increase as the number of samples increased. Fig. 7 shows a comparison between the accuracy distributions for different settings. The fluctuations in accuracy were the largest in the 5-way 1-shot setting and decreased as the number of samples increased. The fluctuations in accuracy were the smallest in the 5-way 10-shot setting. From the perspective of the mean, the magnitude of the accuracy did not change significantly as the number of samples increased.

The simulation results on the CICIDS2017 and CICIDS2018 datasets can be found in the supplementary materials.

5 Discussion

The intrusion detection system for the proposed IoT, which is based on the subspace approach and designed for few-shot scenarios, demonstrates effective performance in detecting known and unknown types of attack within the IoT environment, even under conditions of limited sample availability.

In Table 3, we compared the results of classic machine learning algorithms, deep learning algorithms, and prototype network algorithms in the multi-classification setting. The effect of using CNN on network traffic detection and the results of using the prototype network algorithm under few-shot conditions were not significantly different, and both were <70%. Compared to the K-nearest

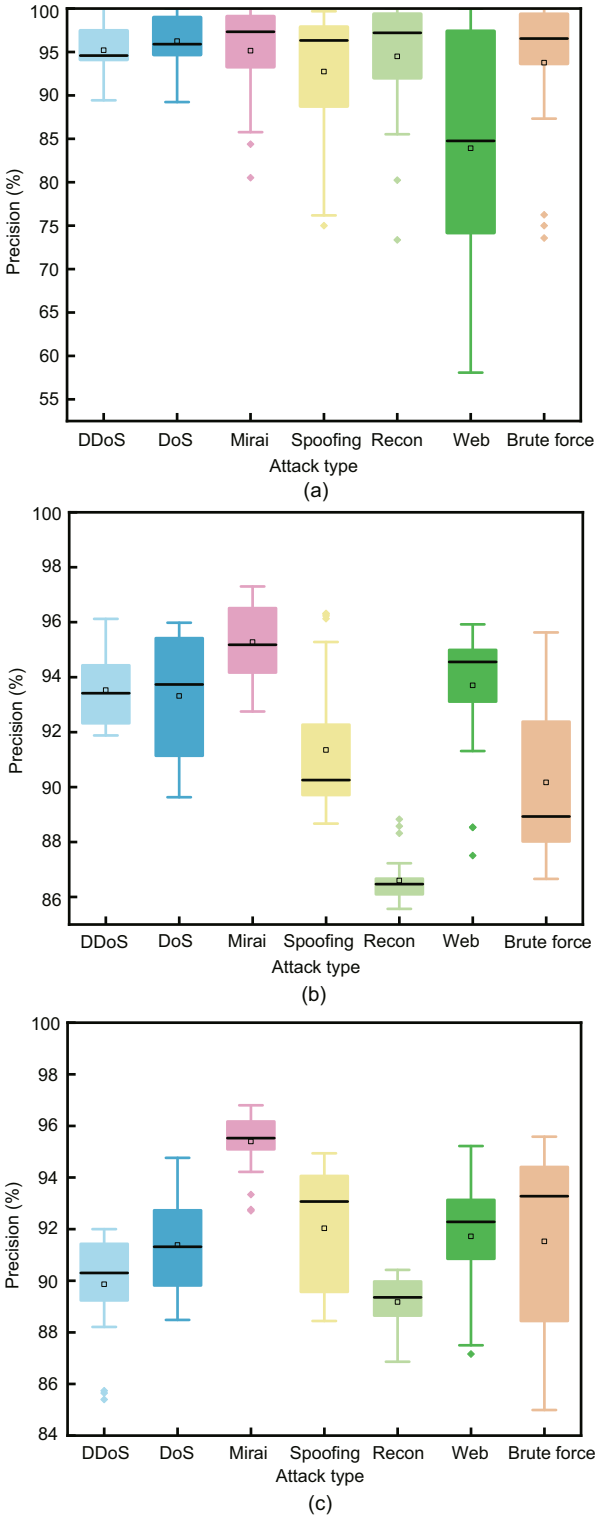

Fig. 6 Distribution of detection precision in different settings: (a) 5-way 1-shot; (b) 5-way 5-shot; (c) 5-way 10-shot

neighbors (KNN) algorithm, the detection accuracy of which was 81.28% and was higher than that of the other comparison algorithms, our proposed algorithm achieved a noticeable improvement in detection accuracy.

In Tables 4 and 5, we compare the results of other researchers from four aspects: dataset, classification setting, number of samples, and accuracy. Among them, the research work of Lu CM et al. (2023) is more similar to ours. CSV table datasets were used in both studies. The difference is that they converted the table data into RGB color images and used 2D convolution to extract features. By contrast, we directly used raw data and extracted features through 1D convolution. We conducted our simulations with the same classification settings and the detection results were better than theirs. Many other studies focused on the study of raw traffic stored in PCAP files; for example, Xu CY et al. (2020) converted raw traffic to RGB color images and extracted the features using 3D convolution; Sun et al. (2023) and Du et al. (2024) converted raw traffic to grayscale images and extracted the features using 2D convolution. Most of the previous studies focused on converting network traffic into images, and the methods used were similar. Yan et al. (2023) used a short-time Fourier transform to convert 1D network traffic into images. We used the CICIoT2023 dataset as the research object, which contains more decadent attack samples than previous datasets and is more reflective of the natural IoT environment.

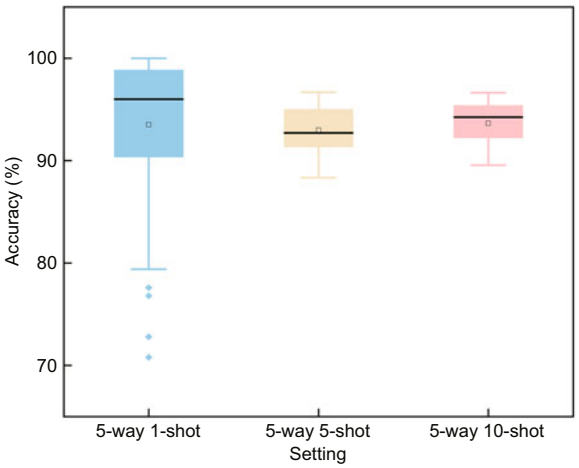

Fig. 7 Accuracy distribution in different settings

Table 3 Comparison of classical methods

Method	Simulation setting	Accuracy (%)
KNN	N/A	81.28
SVM	N/A	46.86
DNN	N/A	61.90
CNN	N/A	66.86
Prototypical network	5-way 1-shot	54.98
Prototypical network	5-way 5-shot	68.74
Prototypical network	5-way 10-shot	69.74
Proposed approach	5-way 1-shot	93.52
Proposed approach	5-way 5-shot	92.99
Proposed approach	5-way 10-shot	93.65

N/A: not applicable

Table 4 Comparison of detection results and sample sizes under binary classification settings for the methods presented in this paper and related works

Method	Dataset	IoT dataset	Classification setting	Number of samples	Accuracy (%)
FC-Net (Xu CY et al., 2020)	CICIDS2017	N	2-way 5-shot	5	94.33
	CICIDS2017	N	2-way 10-shot	10	94.64
FCAD (Feng et al., 2021)	CICAndMal2017	N	N/A	5	94.30
	CICAndMal2017	N	N/A	10	97.10
Proto+Reptile (Xu H and Wang, 2022)	CICIDS2017+NDSec-1	N	2-way 5-shot	5	97.56
	CICIDS2017+NDSec-1	N	2-way 10-shot	10	97.43
L2F+MAML (Shi et al., 2023)	CICIDS2017	N	2-way 10-shot	10	94.66
	CICIDS2018	N	2-way 5-shot	5	96.00
	CICIDS2018	N	2-way 10-shot	10	97.29
Proposed approach	CICIDS2017	N	2-way 5-shot	5	99.16
	CICIDS2017	N	2-way 10-shot	10	99.26
	CICIDS2018	N	2-way 5-shot	5	99.06
	CICIDS2018	N	2-way 10-shot	10	98.57
	CICIoT2023	Y	2-way 5-shot	5	96.12
	CICIoT2023	Y	2-way 10-shot	10	97.72

Table 5 Comparison of detection results and sample sizes under multi-classification settings for the methods presented in this paper and related works

Method	Dataset	IoT dataset	Classification setting	Number of samples	Accuracy (%)
MAML+CNN (Lu CM et al., 2023)	FSIDS-IoT	Y	5-way 1-shot	1	73.81
	FSIDS-IoT	Y	5-way 5-shot	5	89.64
	FSIDS-IoT	Y	5-way 10-shot	10	92.19
IPN-IDS (Wang YH et al., 2024)	CICIDS2017	N	5-way 1-shot	1	75.45
	CICIDS2017	N	5-way 5-shot	5	84.94
	CICIDS2017	N	5-way 10-shot	10	86.35
	CICIoT2023	Y	5-way 1-shot	1	61.03
	CICIoT2023	Y	5-way 5-shot	5	66.93
	CICIoT2023	Y	5-way 10-shot	10	68.77
Proposed approach	CICIDS2017	N	5-way 1-shot	1	83.88
	CICIDS2017	N	5-way 5-shot	5	92.66
	CICIDS2017	N	5-way 10-shot	10	95.58
	CICIDS2018	N	5-way 1-shot	1	91.00
	CICIDS2018	N	5-way 5-shot	5	94.16
	CICIDS2018	N	5-way 10-shot	10	94.70
	CICIoT2023	Y	5-way 1-shot	1	93.52
	CICIoT2023	Y	5-way 5-shot	5	92.99
	CICIoT2023	Y	5-way 10-shot	10	93.65

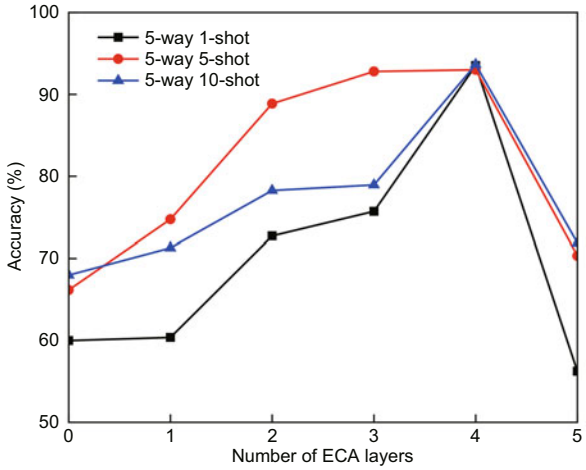

Fig. 8 Accuracy with different numbers of ECA layers on model performance

We also discussed the effect of different numbers of ECA layers on the performance of the model. As shown in Fig. 8, we can see that the accuracy of the model did not increase with more layers superimposed. Regardless of the classification setting, the detection accuracy was the highest when the number of added ECA layers was 4. Thus, we can conclude that the detection accuracy of the model approaches a threshold with an increase in the number of ECA layers and does not always increase.

6 Conclusions

In this study, we developed a subspace-based classification model to solve the few-shot IoT intrusion detection problem. First, we constructed an intrusion detection dataset suitable for few-shot scenarios based on the latest IoT dataset, CICIoT2023. Next, the data samples for 1D CNN learning were formed by normalizing the tabular data and transforming the data types. Subsequently, according to the N-way K-shot learning paradigm, a set of tasks used for model training was constructed. Each task was used to extract the features of each sample through CNN with the ECA mechanism, and the classification effect was realized through a subspace-based classifier. For unknown attack classes, our model eliminated fine-tuning requirements by leveraging prior knowledge to achieve rapid generalization across novel threat categories. Our model showed good detection results for all models at sample number $K = 1$, 5, and 10. However, the performance and robustness of our model should be improved when only one sample is available.

Contributors

Zhihui LI designed the research. Zhihui LI and Congyuan XU completed the simulations. Kun DENG and

Chunyuan LIU processed the data. Zhihui LI drafted the paper. Congyuan XU, Kun DENG, and Chunyuan LIU helped organize the paper. All the authors revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Alani MM, Awad AI, 2023. An intelligent two-layer intrusion detection system for the Internet of Things. *IEEE Trans Ind Inform*, 19(1):683-692. https://doi.org/10.1109/TII.2022.3192035

Chen D, Zhang FB, Zhang XP, 2023. Heterogeneous IoT intrusion detection based on fusion word embedding deep transfer learning. *IEEE Trans Ind Inform*, 19(8):9183-9193. https://doi.org/10.1109/TII.2022.3227640

Draper-Gil G, Lashkari AH, Mamun MSI, et al., 2016. Characterization of encrypted and VPN traffic using time-related features. Proc 2nd Int Conf on Information Systems Security and Privacy, p.407-414. https://doi.org/10.5220/0005740704070414

Du L, Gu ZQ, Wang Y, et al., 2024. A few-shot class-incremental learning method for network intrusion detection. *IEEE Trans Netw Serv Manag*, 21(2):2389-2401. https://doi.org/10.1109/TNSM.2023.3332284

Duan GH, Lv HW, Wang HQ, et al., 2023. Application of a dynamic line graph neural network for intrusion detection with semisupervised learning. *IEEE Trans Inform Foren Secur*, 18:699-714. https://doi.org/10.1109/TIFS.2022.3228493

Duan RX, Li D, Tong Q, et al., 2021. A survey of few-shot learning: an effective method for intrusion detection. *Secur Commun Netw*, 2021(1):4259629. https://doi.org/10.1155/2021/4259629

Feng TT, Qi Q, Wang JY, et al., 2021. Few-shot class-adaptive anomaly detection with model-agnostic meta-learning. IFIP Networking Conf, p.1-9.

Finn C, Abbeel P, Levine S, 2017. Model-agnostic meta-learning for fast adaptation of deep networks. Proc 34th Int Conf on Machine Learning, p.1126-1135.

Fouladi RF, Ermiş O, Anarim E, 2022. A DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN. *Comput Netw*, 214:109140. https://doi.org/10.1016/j.comnet.2022.109140

He MS, Zhao XW, Wang XJ, 2024a. An efficient DDoS detection method based on packet grouping via online data flow processing. *IEEE Trans Sustain Comput*, 10(2):202-216. https://doi.org/10.1109/TSUSC.2024.3409712

He MS, Huang YM, Wang XL, et al., 2024b. A lightweight and efficient IoT intrusion detection method based on feature grouping. *IEEE Int Things J*, 11(2):2935-2949. https://doi.org/10.1109/JIOT.2023.3294259

Jamal MA, Qi GJ, 2019. Task agnostic meta-learning for few-shot learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.11711-11719. https://doi.org/10.1109/CVPR.2019.01199

Koch G, Zemel R, Salakhutdinov R, 2015. Siamese neural networks for one-shot image recognition. Proc 32nd Int Conf on Machine Learning, p.1-30.

Lu CM, Wang XF, Yang AM, et al., 2023. A few-shot-based model-agnostic meta-learning for intrusion detection in security of Internet of Things. *IEEE Int Things J*, 10(24):21309-21321. https://doi.org/10.1109/JIOT.2023.3283408

Lu HM, Wang T, Xu X, et al., 2022. Cognitive memory-guided autoencoder for effective intrusion detection in Internet of Things. *IEEE Trans Ind Inform*, 18(5):3358-3366. https://doi.org/10.1109/TII.2021.3102637

Makkar A, Garg S, Kumar N, et al., 2021. An efficient spam detection technique for IoT devices using machine learning. *IEEE Trans Ind Inform*, 17(2):903-912. https://doi.org/10.1109/TII.2020.2968927

Mehedi ST, Anwar A, Rahman Z, et al., 2023. Dependable intrusion detection system for IoT: a deep transfer learning based approach. *IEEE Trans Ind Inform*, 19(1):1006-1017. https://doi.org/10.1109/TII.2022.3164770

Neto ECP, Dadkhah S, Ferreira R, et al., 2023. CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13):5941. https://doi.org/10.3390/s23135941

Nichol A, Achiam J, Schulman J, 2018. On first-order meta-learning algorithms. https://arxiv.org/abs/1803.02999

Niu ZQ, Guo WJ, Xue JF, et al., 2023. A novel anomaly detection approach based on ensemble semi-supervised active learning (ADESSA). *Comput Secur*, 129:103190. https://doi.org/10.1016/j.cose.2023.103190

Ouyang YK, Li BB, Kong QL, et al., 2021. FS-IDS: a novel few-shot learning based intrusion detection system for SCADA networks. IEEE Int Conf on Communications, p.1-6. https://doi.org/10.1109/ICC42927.2021.9500667

Schwartz E, Karlinsky L, Shtok J, et al., 2018. Δ-encoder: an effective sample synthesis method for few-shot object recognition. 32nd Conf on Neural Information Processing Systems, p.2850-2860.

Shi ZX, Xing MY, Zhang J, et al., 2023. Few-shot network intrusion detection based on model-agnostic meta-learning with L2F method. IEEE Wireless Communications and Networking Conf, p.1-6. https://doi.org/10.1109/WCNC55385.2023.10118898

Simon C, Koniusz P, Nock R, et al., 2020. Adaptive subspaces for few-shot learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.4135-4144. https://doi.org/10.1109/CVPR42600.2020.00419

Snell J, Swersky K, Zemel R, 2017. Prototypical networks for few-shot learning. Proc 31st Int Conf on Neural Information Processing Systems, p.4080-4090.

Sun HD, Wan L, Liu MY, et al., 2023. Few-shot network intrusion detection based on prototypical capsule network with attention mechanism. *PLoS ONE*, 18(4):e0284632. https://doi.org/10.1371/journal.pone.0284632

Sung F, Yang YX, Zhang L, et al., 2018. Learning to compare: relation network for few-shot learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.1199-1208. https://doi.org/10.1109/CVPR.2018.00131

Thakkar A, Lohiya R, 2023. Attack classification of imbalanced intrusion data for IoT network using ensemble-learning-based deep neural network. *IEEE Int Things J*, 10(13):11888-11895. https://doi.org/10.1109/JIOT.2023.3244810

Vinyals O, Blundell C, Lillicrap T, et al., 2016. Matching networks for one shot learning. Proc 30th Int Conf on Neural Information Processing Systems, p.3637-3645.

Wang QL, Wu BG, Zhu PF, et al., 2020. ECA-Net: efficient channel attention for deep convolutional neural networks. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.11531-11539. https://doi.org/10.1109/CVPR42600.2020.01155

Wang YH, Zhang ZY, Zhao KJ, et al., 2024. A few-shot learning based method for industrial Internet intrusion detection. *Int J Inform Secur*, 23(5):3241-3252. https://doi.org/10.1007/s10207-024-00889-x

Wang YK, Xu CM, Liu C, et al., 2020. Instance credibility inference for few-shot learning. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.12833-12842. https://doi.org/10.1109/CVPR42600.2020.01285

Wang YX, Girshick R, Hebert M, et al., 2018. Low-shot learning from imaginary data. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.7278-7286. https://doi.org/10.1109/CVPR.2018.00760

Wang ZM, Tian JY, Qin J, et al., 2021. A few-shot learning-based Siamese capsule network for intrusion detection with imbalanced training data. *Comput Intell Neurosci*, 2021:7126913. https://doi.org/10.1155/2021/7126913

Xu CY, Shen JZ, Du X, 2020. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans Inform Forens Secur*, 15:3540-3552. https://doi.org/10.1109/TIFS.2020.2991876

Xu H, Wang YJ, 2022. A continual few-shot learning method via meta-learning for intrusion detection. IEEE 4th Int Conf on Civil Aviation Safety and Information Technology, p.1188-1194. https://doi.org/10.1109/ICCASIT55263.2022.9986665

Yan Y, Yang Y, Gu YH, et al., 2023. A few-shot intrusion detection model for the Internet of Things. 3rd Int Conf on Electronic Information Engineering and Computer Science, p.531-537. https://doi.org/10.1109/EIECS59936.2023.10435498

Yan Y, Yang Y, Shen F, et al., 2024. Meta learning-based few-shot intrusion detection for 5G-enabled industrial Internet. *Compl Intell Syst*, 10(3):4589-4608. https://doi.org/10.1007/s40747-024-01388-1

List of supplementary materials

1 Simulations and analysis

2 Discussion

Table S1 Information on the CICIDS2017 dataset

Table S2 Information on the CICIDS2018 dataset

Fig. S1 Confusion matrices in different settings

Fig. S2 Detection precision on the CICIDS2017 dataset

Fig. S3 Detection precision on the CICIDS2018 dataset

Fig. S4 Visualization of dataset before model training

Fig. S5 Classification visualization after model training