A Biological Immunity-Based Neuro Prototype for Few-Shot Anomaly Detection with Character Embedding

Cyborg and Bionic Systems
A SCIENCE PARTNER JOURNAL

RESEARCH ARTICLE

A Biological Immunity-Based Neuro Prototype for Few-Shot Anomaly Detection with Character Embedding

Zhongjing Ma[1], Zhan Chen[1], Xiaochen Zheng[2], Tianyu Wang[1], Yuyang You[1], Suli Zou[1], and Yu Wang[3*]

[1]School of Automation, Beijing Institute of Technology, Beijing 100081, China. [2]ETH AI Center, Andreasstrasse 5, 8092 Zürich, Switzerland. [3]State Key Lab of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100095, China.

*Address correspondence to: yu.wang@ia.ac.cn

Citation: Ma Z, Chen Z, Zheng X, Wang T, You Y, Zou S, Wang Y. A Biological Immunity-Based Neuro Prototype for Few-Shot Anomaly Detection with Character Embedding. *Cyborg Bionic Syst.* 2024;5:Article 0086. https://doi.org/10.34133/cbsystems.0086

Submitted 17 September 2023
Accepted 11 December 2023
Published 17 January 2024

Anomaly detection has wide applications to help people recognize false, intrusion, flaw, equipment failure, etc. In most practical scenarios, the amount of the annotated data and the trusted labels is low, resulting in poor performance of the detection. In this paper, we focus on the anomaly detection for the text type data and propose a detection network based on biological immunity for few-shot detection, by imitating the working mechanism of the immune system of biological organisms. This network enabling the protected system to distinguish the aggressive behavior of "nonself" from the legitimate behavior of "self" by embedding characters. First, it constructs episodic task sets and extracts data representations at the character level. Then, in the pretraining phase, Word2Vec is used to embed the representations. In the meta-learning phase, a dynamic prototype containing encoder, routing, and relation is designed to identify the data traffic. Compare to the mean-based prototype, the proposed prototype applies a dynamic routing algorithm that assigns different weights to samples in the support set through multiple iterations to obtain a prototype that combines the distribution of samples. The proposed method is validated on 2 real traffic datasets. The experimental results indicate that (a) the proposed anomaly detection prototype outperforms state-of-the-art few-shot techniques with 1.3% to 4.48% accuracy and 0.18% to 4.55% recall; (b) under the premise of ensuring the accuracy and recall, the number of training samples is reduced to 5 or 10; (c) ablation experiments are designed for each module, and the results show that more accurate prototypes can be obtained by using the dynamic routing algorithm.

Introduction

Text data analysis can effectively help us understand the data corpus, quickly identify potential problems in the data, and guide subsequent model training and selection. This kind of data is widely presented in networks, Internet, logs, devices, and operating systems. In order to find the anomalies in the data and prevent the damage to the system, an anomaly detection (AD) has been used as one of the most critical systems [1]. For example, log AD refers to finding abnormal logs to determine the cause and nature of system faults. Usually, log data is modeled as a natural language sequence for AD.

Machine learning and deep learning are widely leveraged in the field of AD, hoping to improve the performance of AD systems, such as misuse based detection [2], deception based detection, and bio-based detection [3]. Machine learning uses algorithms to parse network data, learn characteristics of traffic data, and then classify and predict a certain class of things. Classic machine learning models, such as random forest [4], support vector machine (SVM) [5], and Adaboost [6], have been introduced to detect anomalies. Horng et al. [7] proposed an AD system based on SVM, in which a hierarchical clustering algorithm was used to deal with typical data. It reduced the complexity and redundancy of the dataset and further reduced the training time. Al-Yaseen et al. [8] proposed a multilevel hybrid detection model based on SVM, with the results showing an accuracy of 95.75%. In [9], a network AD system was presented based on Light GBM. They use an oversampling technique to increase minority samples of imbalanced training data to increase the detection accuracy. Since the detection performance of machine-learning-based algorithms is related to the manual selection of features, it requires plenty of professional knowledge to deeply mine the deep features in the data.

The deep-learning-based algorithms are to learn the inherent distribution and representation level of sample data and can automatically extract the characteristics of data such as text, images and sounds. At the same time, the nonlinear hidden layer structure in the neural network is helpful for the learning and prediction of high-dimensional data. In this context, deep learning algorithms are gradually considered for detection tasks, such as autoencoders [10,11], convolutional neural networks (CNNs) [12,13] and long short-term memory networks (LSTMs) [14]. For example, Min et al. [2] proposed a system that combined

Text-CNN and random forest to construct an anomaly-based network detection system. Kim et al. [15] used deep learning to generate virtual samples on the dataset, and proposed a malware detection method based on deep convolutional generative adversarial networks. The role of generative adversarial network is to generate similar data to detect the deformation of malware more accurately. In [16], it applied a deep belief neural network to extract data features, and then use the backward propagation neural network as a classifier to identify traffic data anomalies. In [17], it proposed an AD method based on BiLSTM deep learning. Experiments showed that in binary and multiclass AD, BiLSTM not only improved the performance of traditional LSTM but also had higher detection accuracy.

An AD based on deep learning is essentially a classifier trained on a large amount of data, highly dependent on feature engineering and dataset capacity. Data imbalance and less training data often appear in reality, which will cause overfitting, falling into local optimal solutions, or other problems. Concerning about these problems, some researchers have tried to introduce few-shot learning prototype into network AD in recent years, such as [18–27]. Yu et al. [28] used the deep neural network (DNN) and CNN as the traffic embedding network to map each sample into a high-dimensional sample space. After that, the prototype vector of each anomaly category is obtained by averaging. Finally, the distance between the new anomaly and each prototype vector is measured to obtain the classification result. Rong et al. [25] proposed a few-shot learning based prototype UMVD-FSL to detect unseen malware variants with a small set of data. Start with network flow data generated by malware variants and benign applications, then convert them to grayscale images. A prototype-based few-shot learning model takes grayscale images as input and leverages meta-training to generalize the meta-learner to adapt to new tasks. Xu et al. [18] designed a deep neural network, which is mainly composed of 2 parts: feature extraction network and comparison network, to classify network traffic samples. Guo et al. [26] integrated a global attention mechanism and aggregated the global information of inputs by capturing the byte relationship between payload sequence. A metric-based AD prototype was proposed in [27], which makes feature extractor fuse original bytes content with network flow features to improve detection precision and recall. As noted by Sung et al., RelationNet [21] builds a learnable nonlinear comparator through neural network to calculate the distance between 2 samples and then analyzes the sample similarity, instead of a fixed linear comparator such as Euclidean distance or cosine distance. In general, the above methods apply mean-based prototypes to measure similarity by Euclidean distance to obtain classification results. However, the nearest-neighbor classifier based on mean prototype and the fixed linear comparator easily cause the estimation bias due to the data scarcity in few-shot scenarios and finally affect detection precision and recall. What is more, because of the hindrance of data parsing, building a universal network system to detect network traffic attacks is still a tough task for deep learning methods.

In this paper, we leverage the the working mechanism of the immune system to design a neural prototype for few-shot AD with character embedding, namely CharNet, which combines text embedding techniques and improved metric-based few-shot learning, improving the accuracy and recall of the existing deep models for few-shot network traffic classification. Specifically, CharNet consists of dataset construction, pretraining, meta-learning 3 phases. Dataset construction phase is to transform the original traffic dataset into a episode-based dataset, converting the multiclassification problem into a 2-way K-shot problem. To skip the data parsing session, Word2Vec, a self-supervised learning method, is used in the pretraining phase to convert traffic into embedding vectors. Therefore, the multidimensional feature data classification task is transformed into a text classification task. In the meta-learning phase, we design a dynamic routing prototype network, which consists of 3 modules: encoder, routing, and relation, to identify whether the network traffic is normal or not. Compare to the mean-based prototype, CharNet uses a dynamic routing algorithm that assigns different weights to samples in the support set through multiple iterations to obtain a prototype that combines the distribution of samples. At last, the relation module gives the classification results by properly performing comparison between those traffic embedding vectors. The main contributions of this paper are as follows:

- We propose the CharNet for few-shot AD. This method uses the dynamic routing algorithm to assign weights to the samples in the support set and builds a routing-based prototype, which effectively reduces the estimation bias and sampling bias caused by a small number of samples.
- We treat network traffic as a string and use character-level coding to omit data parsing sessions. This processing method can use the Word2Vec method to pretrain the embedding layer, which can learn prior knowledge for network traffic classification, effectively improving the training speed and accuracy.
- Compare our method with other methods, the accuracy of our method has improved. Using the proposed method, new types of samples on the basis of only a limited number of labels in an untrained dataset can be detected relying on learned prior knowledge.

The rest of the paper is organized as follows. Materials and Methods reviews the related work to our method. Experiments describes the prototype and details on our proposed prototype and presents our experiments and make comparisons with big-data methods and other few-shot learning methods. Conclusion summarizes the paper and make the conclusion.

Materials and Methods

Problem formulation

We consider network AD as the task of few-shot classifier learning, whose purpose is to train a classifier $f_\theta(\cdot)$ with few samples x_i and predict the corresponding labels $\widehat{y}_i$ of new samples $\widehat{x}_i$. As shown in Fig. 1, we have 3 datasets: a base set $\mathcal{D}_{\text{base}}$, a meta-training task set $\mathcal{T}_{\text{train}} = \left\{\mathcal{S}_{\text{train}}, \mathcal{Q}_{\text{train}}\right\}$, and a meta-test task set $\mathcal{T}_{\text{test}} = \left\{\mathcal{S}_{\text{test}}, \mathcal{Q}_{\text{test}}\right\}$. Base set has a large number of samples with a set of classes $\mathcal{C}_{\text{base}}$. If the support set of meta task sets contain K-labeled samples for each of N unique classes, the target few-shot problem is called N-way K-shot. As the project's goal is usually to distinguish between normal samples and a particular type of malicious samples, we consider the network AD as a 2-way K-shot problem.

According episode-based training proposed in [29], in each episode iteration, the support set $\mathcal{S}$ is formed by K-labeled samples from each of the C classes, that is, $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{K\times C}$. Similarly, the query set $\mathcal{Q}$ is formed from the remainder of those C classes' samples, that is, $\mathcal{Q} = \{(x_j, y_j)\}_{j=1}^{B}$. Meanwhile, the base set with abundant labeled samples $\mathcal{D}_{\text{base}} = \{(x_i, y_i)\}_{i=1}^{m}$. Both meta task sets contain the support set and the query set, and the support set and query set share the same label space, but the label space of meta-training task set is disjoint with the

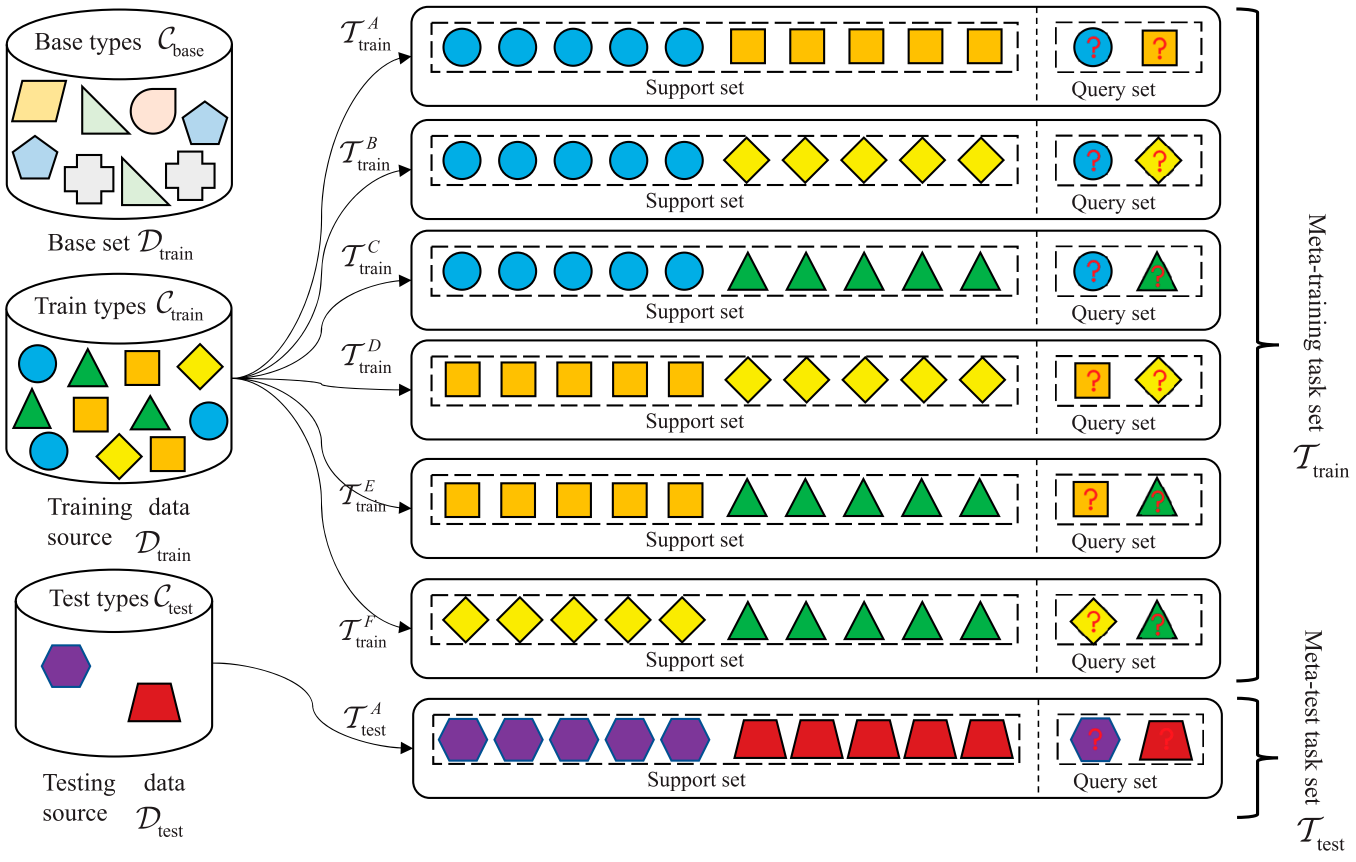
Base types $\mathcal{C}_{\text{base}}$
Base set $\mathcal{D}_{\text{train}}$
Train types $\mathcal{C}_{\text{train}}$
Training data source $\mathcal{D}_{\text{train}}$
Test types $\mathcal{C}_{\text{test}}$
Testing data source $\mathcal{D}_{\text{test}}$
$\mathcal{T}^{A}_{\text{train}}$
$\mathcal{T}^{B}_{\text{train}}$
$\mathcal{T}^{C}_{\text{train}}$
$\mathcal{T}^{D}_{\text{train}}$
$\mathcal{T}^{E}_{\text{train}}$
$\mathcal{T}^{F}_{\text{train}}$
$\mathcal{T}^{A}_{\text{test}}$
Support set
Query set
Meta-training task set $\mathcal{T}_{\text{train}}$
Meta-test task set $\mathcal{T}_{\text{test}}$

Fig. 1. The illustration of the division of meta-learning tasks.

the label space of meta-test task set. That is, $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{test}} = \emptyset$, $\mathcal{S}_{\text{train}} \cap \mathcal{Q}_{\text{train}} = \emptyset$, and $\mathcal{S}_{\text{test}} \cap \mathcal{Q}_{\text{test}} = \emptyset$.

Our goal is to learn a good meta-learner $f_\theta(\cdot)$ on the support set $\mathcal{S}$ based on prior knowledge obtained on the base set $\mathcal{D}_{\text{base}}$ so that it can perform well on the query set $\mathcal{Q}$. In our few-shot experiments (see Results and Discussion), we consider 5-shot ($K = 5$) and ten-shot ($K = 10$) settings.

Overall prototype

As shown in Fig. 2, the proposed prototype is divided into 3 phases, including dataset construction, pretraining, and meta-learning.

Dataset construction

In the phase, we will construct base set $\mathcal{D}_{\text{base}}$, train task set $\mathcal{T}_{\text{train}}$, and test task set $\mathcal{T}_{\text{test}}$. We firstly perform preprocessing operations, such as duplicate value deletion, default value supplementation, and zero padding at the end of the most extended traffic log length. Then, each traffic log is tokenized by character level to extract fine-grained feature expressions. After that, in order to define each task as a 2-way K-shot task, we mix K normal samples with K malicious samples of each type.

The classes $\mathcal{C}_{\text{train}}$ with a large number of samples are selected as the train task set $\mathcal{T}_{\text{train}}$, and the remaining classes $\mathcal{C}_{\text{test}}$ is used as the test task set $\mathcal{T}_{\text{test}}$. To follow the episode-based strategy, the few-shot task generating details are shown in Algorithm 1. In order to improve the training speed and accuracy, we take the support set of train task set $\mathcal{T}_{\text{train}}\langle\mathcal{S}\rangle$ as the base set $\mathcal{D}_{\text{base}}$ to participate in pretraining.

Pretraining

A Word2Vec [30] self-supervised classifier is trained with the samples in base set $\mathcal{D}_{\text{base}}$. Skip-gram model is applied in our word-embedding task, which predicts the surrounding words from the central word. We define 2 parameter matrices, $W \in \mathbb{R}^{D\times|V|}$ and $W' \in \mathbb{R}^{|V|\times D}$, where D represents the embedding dimension and can be set to any size. Since we treat network traffic as a string, we tokenize the string by characters [31] and one-hot encode it according to the dictionary V to get the sequence $u = \{u_1, u_2, \ldots, u_L\}$, L is the length of the sequence. Here, $|V|$ is the size of the dictionary set V. Skip-gram works as the following steps:

Algorithm 1 Few-shot Task Generating

Input: Query set size B. Class $\mathcal{C} = \{0, 1, \ldots, N\}$. Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathcal{C}$. $D\langle t\rangle$ denotes the subset of $\mathcal{D}$ containing all elements (x_i, y_i) that $y_i = t$. The number of samples of each class K.
Require: RandSample(D, K), denotes the random selection of K elements from dataset $\mathcal{D}$.
Output: Few-shot task $\mathcal{T}_i = \{\mathcal{S}, \mathcal{Q}\}$.

```
Initialize support set $\mathcal{S} = \{\}$, query set $\mathcal{Q} = \{\}$.
$Label \leftarrow$ RandSample$(\{1, \ldots, N\}, 1)$
(1) Generate support set
$\mathcal{S}\langle 0\rangle \leftarrow$ RandSample$(D\langle 0\rangle, K)$
$\mathcal{S}\langle 1\rangle \leftarrow$ RandSample$(D\langle Label\rangle, K)$
$\mathcal{S} \leftarrow \mathcal{S}\langle 0\rangle \cup \mathcal{S}\langle 1\rangle$
(2) Generate query set
$\mathcal{Q}\langle 0\rangle \leftarrow$ RandSample$(D\langle 0\rangle, B/2)$
$\mathcal{Q}\langle 1\rangle \leftarrow$ RandSample$(D\langle Label\rangle, B/2)$
$\mathcal{Q} \leftarrow \mathcal{Q}\langle 0\rangle \cup \mathcal{Q}\langle 1\rangle$
(3) Replace label
for $(x_i, y_i)$ in $\{\mathcal{S}, \mathcal{Q}\}$ do
    if $y_i \neq 0$ then
        $y_i \leftarrow 1$
    end if
end for
(4) Generate task
$\mathcal{T}_i \leftarrow \{\mathcal{S}, \mathcal{Q}\}$
return $\mathcal{T}_i$
```

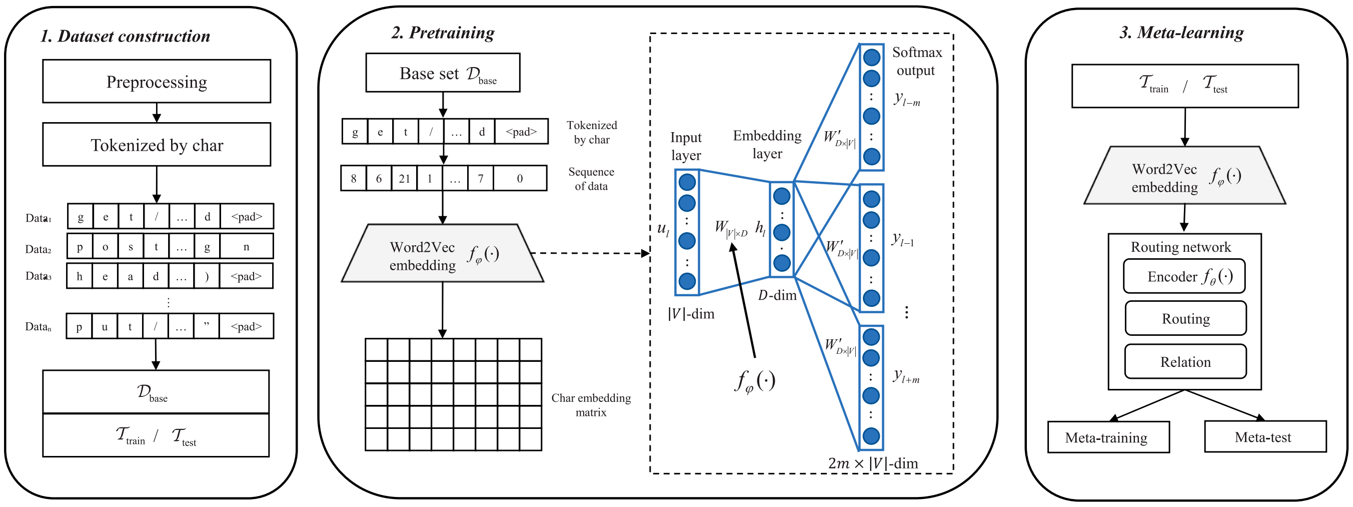

Fig. 2. The prototype of few-shot AD.

Meta-learning

In this phase, a meta-learner $f_\theta(\cdot)$ learn meta knowledge in $\mathcal{C}_{\text{train}}$, and they can learn quickly and accurately with few data in $\mathcal{C}_{\text{test}}$. This phase is divided in 2 phases: meta-training and meta-test. In Dataset construction, we have constructed a train task set $\mathcal{T}_{\text{train}}$ and a test task set $\mathcal{T}_{\text{test}}$, both containing the support set S and the query set Q. In Pretraining, a feature extractor $f_\varphi(\cdot)$ is trained on $\mathcal{D}_{\text{base}}$.

In meta-training phase, based on the feature extractor $f_\varphi(\cdot)$, a meta-learner $f_\theta(\cdot)$ is trained on the support set of train task set $\mathcal{T}_{\text{train}}\langle\mathcal{S}\rangle$ by maximizing the likelihood estimation on the query set of train task set $\mathcal{T}_{\text{train}}\langle\mathcal{Q}\rangle$. That is,

$$\max_\theta \mathbb{E}_{(\mathcal{S},\mathcal{Q})\in\mathcal{T}_{\text{train}}} \sum_{(x,y)\in\mathcal{Q}} \log\left(P\left(y|x,\omega,\mathcal{S},\varphi,\theta\right)\right) \tag{1}$$

where ω represents meta knowledge, and φ and θ represent the parameters of $f_\varphi(\cdot)$ and $f_\theta(\cdot)$, respectively. We learn meta knowledge by sampling a large number of train tasks, so the optimal meta knowledge ω can be expressed as this:

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \log p\left(\omega \mid \mathcal{T}_{\text{train}}\right) \tag{2}$$

In meta-test phase, based on the optimal meta knowledge ω^* that has been learned, the optimal meta-learner parameters θ^* are found as following:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log p\left(\theta \mid \omega^*, \mathcal{T}_{\text{test}}\right) \tag{3}$$

As for evaluation, we directly predict labels in $\mathcal{T}_{\text{test}}\langle\mathcal{Q}\rangle$ by the optimal meta-learner $f_\theta^*(\cdot)$ and then compare with the ground truth to evaluate the performance.

Architecture of CharNet

Our CharNet includes 3 modules: encoder module, routing module, and relation module, which is shown in Fig. 3 (the case of 2-way 3-shot model).

Encoder module

In order to consider both the historical information and future information of the sequence, and let the model focus more on finding helpful information in the input data that is salient and relevant to the current output, we adopt the bidirection LSTM network with self-attention [32]. For simplicity, the encoder module receives an input sequence $x = (c_1, c_2, \ldots, c_L)$, where c_l

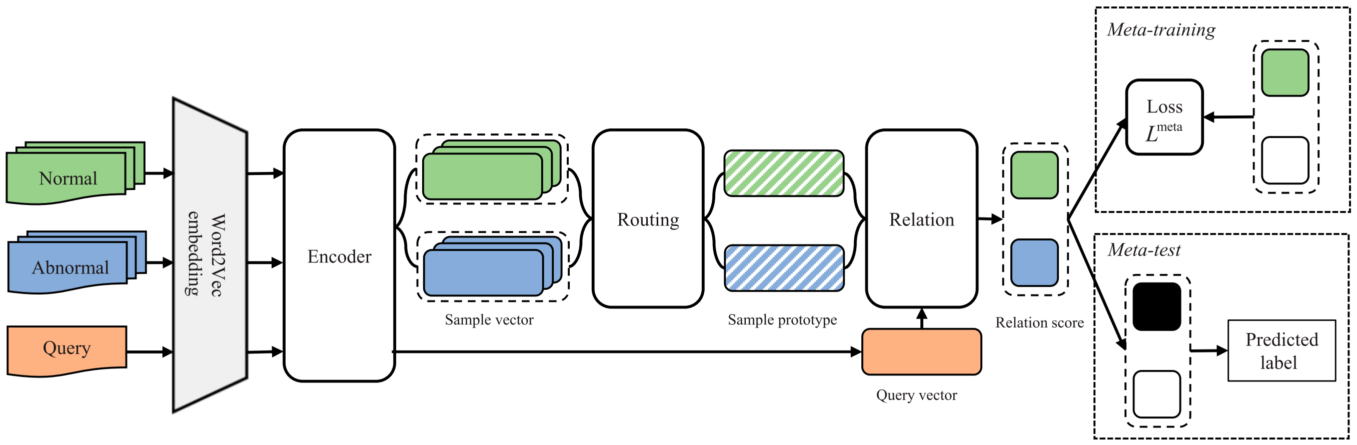

Fig. 3. The architecture of CharNet.

Table 1. The number of samples in the CICIDS2017FS

	Training task		Test task	
Type of attacks	Support set	Query set	Support set	Query set
Botnet	2,622	1,310	-	-
DoS-GoldenEye	13,724	6,862	-	-
DoS-Hulk	308,098	154,048	-	-
DoS-slowloris	7,728	3,864	-	-
Heartbleed	16	6	-	-
Infilteration	48	24	-	-
SSH-Patator	7,864	3,930	-	-
Brute force	2,010	1,004	-	-
SQL injection	28	14	-	-
XSS	870	434	-	-
DDos	-	-	516	2,034
DoS-Slowhttptest	-	-	496	2,062
FTP-Patator	-	-	424	207
PortScan	-	-	490	2,032

Table 2. The number of samples in the CM2021FS

	Training task		Test task	
Type of attacks	Support set	Query set	Support set	Query set
Cloud server access	29,796	14,896	-	-
Blocked IP	25,216	12,608	-	-
Crawler tool	16,106	8,052	-	-
SQL injection	-	-	190	810
Directory traversal	-	-	70	700
XSS cross-site	-	-	24	120

represents the character embedding, L represents the length of sequence. The forward hidden state $\vec{h}_t$ and the reverse hidden state $\overleftarrow{h}_t$ are obtained by biLSTM and then concatenate $\vec{h}_t$ and $\overleftarrow{h}_t$ to obtain the hidden state h_t.

$$\vec{h}_l = \overrightarrow{LSTM}\left(c_l, h_{l-1}\right) \tag{4}$$

$$\overleftarrow{h}_l = \overleftarrow{LSTM}\left(c_l, h_{l+1}\right) \tag{5}$$

$$h_l = concatenate\left(\vec{h}_l, \overleftarrow{h}_l\right) \tag{6}$$

We set the dimension of each LSTM unit hidden state to u, and the set of all hidden units state is $H = (h_1, h_2, \ldots, h_L)$. Through a linear transformation, a variable dimension of input sequence x is transformed into a fixed dimension of hidden state sequence H. Afterwards, the self-attention mechanism is used to assign a corresponding attention score to each hidden state, which takes the set of whole hidden state H as input, and outputs a vector of weights a.

$$a = softmax\left(W_{a2} \tanh\left(W_{a1} H^T\right)\right) \tag{7}$$

Here, $W_{a1} \in R^{d_a \times 2u}$ and $W_{a2} \in R^{d_a}$ are weight matrices and d_a is a hyperparameter. The output representation e of the encoder is the weighted sum of a and H:

$$e = \sum_{l=1}^{L} a_l \cdot h_l \tag{8}$$

Routing module

The dynamic routing algorithm is the core of this section, which is similar to the multihead attention mechanism. It can assign the weight of the samples in the support set through multiple iterations, so as to obtain a prototype vector that combines the distribution of the samples. We regard these vectors e obtained from the support set S by Eq. 8 as sample vectors e^s, and the vectors e from the query set Q as query vectors e^q. Routing module converts sample vectors e^s_{ij} to prototype vectors p_i through a nonlinear mapping, where $i = 1, \ldots, C$ and $j = 1, \ldots, K$.

Since we treat the flow as a string, and the order of the characters plays a crucial role in the model, we multiply all the sample vectors in the support set e^s_{ij} with a transformation matrix $W_s \in R^{2u \times 2u}$ and add a bias b_s. After iteration, a most representative linear mapping can be found. Each sample prediction vector $\hat{e}^s_{ij}$ is computed by:

$$\hat{e}^s_{ij} = squash\left(W_s e^s_{ij} + b_s\right) \tag{9}$$

where *squash* is defined as Eq. 10, which not only ensures that the data is between 0-1, but also preserves the direction of the vector.

$$squash(x) = \frac{\| x \|^2}{1 + \| x \|^2} \frac{x}{\| x \|} \tag{10}$$

In order to ensure that the prototype vector can automatically aggregate the sample feature vectors of this class in the case of very little data capacity, it is necessary to iteratively apply the dynamic routing mechanism. At each iteration, the coupling coefficients d_i for each class i sum to 1 by softmaxing b_i.

$$d_i = softmax\left(b_i\right) \tag{11}$$

Table 3. Comparison of detection result and the number of samples in the proposed method and big-data methods

Approach	Method	Dataset	Type	Number of samples	Acc (%)	Recall (%)
Machine learning	GA-based Adaptive Method (2018) [40]	CICIDS2017	GA + clustering	760,056	N/A	92.85
	Multilayer ensemble SVM (2018) [5]	CICIDS2017	SVM	N/A	N/A	94.94
	DT and Rule Based IDS (2019) [37]	CICIDS2017	REP Tree + random forest	40,000	96.66	94.47
	GBT-based Big Data Method (2019) [36]	CICIDS2017	Gradient Boosted Tree	1,000,000	99.97	N/A
	Improved AdaBoost-based IDS (2019) [6]	CICIDS2017	AdaBoost	158,021	81.83	100.0
	Multi-Stage Optimized ML-based IDS (2021) [4]	CICIDS2017	KNN + random forest	2,830,540	99.99	99.00
Deep learning	Flow-based deep learning method (2018) [14]	CICIDS2017	CNN + LSTM	1,028,007	98.87	98.83
	SU-IDS(2018) [38]	CICIDS2017	Autoencoder	40,000	71.02	N/A
	Deep Hierarchical IDS (2019) [41]	CICIDS2017	CNN + LSTM	553,850	99.91	99.92
	Random attention capsule (2020) [42]	CICIDS2017	Attention + capsule	863,240	98.60	98.61
	DNN-kNN IDS (2020) [34]	CICIDS2017	DNN + kNN	225,745	99.63	99.69
	CLAIRE (2021) [11]	CICIDS2017	Autoencoder + CNN	100,000	98.01	95.20
Ours	CharNet	CICIDS-2017FS	Siamese + Routing	5	95.94	98.78
	CharNet	CICIDS-2017FS	Siamese + Routing	10	99.87	99.98
	CharNet	CM2021FS	Siamese + Routing	5	94.29	96.56
	CharNet	CM2021FS	Siamese + Routing	10	98.56	97.68

where b_i is the logits of coupling coefficients, and initialized by 0 in the first iteration. Given each sample prediction vector $\widehat{e}_{ij}^{s}$, each candidate prototype vector $\widehat{p}_i$ is a weighted sum of all sample prediction vectors $\widehat{e}_{ij}^{s}$ in class i:

$$\widehat{p}_i = \sum_j d_{ij} \cdot \widehat{e}_{ij}^{s} \tag{12}$$

Then the "squash" function is applied to ensure that the length of the vector output of the routing process will not exceed 1:

$$p_i = squash\left(\widehat{p}_i\right) \tag{13}$$

The last step of each iteration is to adjust the logarithm of the coupling coefficient b_{ij} by means of "protocol routing". If the modulus between the sample prediction vector $\widehat{e}_{ij}^{s}$ and the candidate prototype vector is large, that is, the two are very similar, then increase the coupling coefficient of the prediction vector, and through several iterations, a good coupling relationship can be obtained, and the final prototype vector can be obtained.

$$b_{ij} = b_{ij} + \widehat{e}_{ij}^{s} \cdot p_i \tag{14}$$

Formally, the dynamic routing algorithm is shown in Algorithm 2.

Algorithm 2 Dynamic routing algorithm

```
Input: sample vector $e_{ij}^s$ in support set $\mathcal{S}$.
Output: routing prototype $p_i$.
1: Initialize $b_{ij} = 0$.
2: for all samples $j = 1, \ldots, K$ in class $i$:
3:     $\hat{e}_{ij}^s = \mathrm{squash}(W_s e_{ij}^s + b_s)$
4: for $iter$ iterations do
5:     $d_i = \mathrm{softmax}(b_i)$
6:     $\hat{p}_i = \sum_j d_{ij} \cdot \hat{e}_{ij}^s$
7:     $p_i = \mathrm{squash}(\hat{p}_i)$
8:     for all samples $j = 1, \ldots, K$ in class $i$:
9:         $b_{ij} = b_{ij} + \hat{e}_{ij}^s \cdot p_i$
10: end for
11: return $p_i$
```

Relation module

We obtain the prototype vectors p_i through the routing module mentioned in Routing module and use the encoder module mentioned in Encoder module to convert the samples in the query set into query vectors e^q. Then, we need to measure the connection between query vectors e^q and prototype vectors p_i. We draw on the ideas in [21] and use neural networks to replace common mathematical distance metrics. Thus, the similarity measure between p_i and e^q is represented by the relation score, which is between 0 and 1.

Table 4. Few-shot experimental result on CICIDS2017FS

	Two-way 5-shot		Two-way 10-shot	
Methods	Acc (%)	Recall (%)	Acc (%)	Recall (%)
FC- Net [18]	94.13	98.49	95.39	98.19
DF- Net [23]	93.87	96.29	95.56	96.4
GP- Net [26]	94.58	94.23	96.40	96.96
FS- AD [27]	93.60	98.60	97.51	99.17
CharNet	95.94	98.78	99.87	99.98

Table 5. Ablation study with encoder module on 2-way 10-shot tasks

	CICIDS2017FS		CM2021FS	
Encoder module	Acc (%)	Recall (%)	Acc (%)	Recall (%)
biLSTM + attention	99.87	99.85	98.56	98.92
LSTM + attention	95.60	96.07	94.62	95.16
CNN + attention	97.83	97.98	97.58	97.98
Transformer	98.67	99.87	98.47	98.86
biLSTM	98.60	98.76	98.19	97.29

The neural network consists of a neural tensor layer and a sigmoid layer, where the neural tensor layer outputs a relation vector as follows:

$$v\left(p_i, e^q\right) = f\left(p_i^T M^{[1:h]} e^q\right) \tag{15}$$

where $M^k \in R^{2u\times 2u}$, $k \in [1,\ldots,h]$ is one slice of the tensor parameters and f is RELU activation function. The final relation score r_{iq} between the i-th class and the q-th query is calculated by a fully connected layer activated by a sigmoid function.

$$c_{iq} = sigmoid\left(W_r v\left(p_i, e^q\right) + b_r\right) \tag{16}$$

Objective function

In the process of training the model, we set the mean square error loss as the loss function. The purpose is to transform a 2-way classification problem into a similarity regression problem, that is, the similarity problem between relationship scores r_{iq} and the ground truth y_q. Given the support set $\mathcal{S}$ with 2 classes and query set $\mathcal{Q}$ in an episode, the loss function is defined as:

$$L(\mathcal{S}, \mathcal{Q}) = \sum_{i=1}^{2} \sum_{q=1}^{n} \left(r_{iq} - \mathbf{1}\left(y_q == i\right)\right)^2 \tag{17}$$

All parameters of the 3 modules are trained jointly by backpropagation. The stochastic gradient descent is used on all parameters in each training episode. Our model does not need any finetuning on the classes it has never seen due to its generalization nature. The routing and comparison ability are accumulated in the model along with the training episodes.

Results and Discussion

In this section, 2 datasets are selected to evaluate the performance of our proposed CharNet by comparing with big-data and few-shot network AD methods respectively. After that, the function of each module is analyzed through ablation study.

Datasets

CICIDS2017FS

CICIDS2017 dataset [33] contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). Each network traffic has been labeled by using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack. The dataset includes the most common attacks based on the 2016 McAfee report, such as Web based, Brute force, DoS, DDoS, Infiltration, Heart-bleed, Bot and Scant. We select 14 different types of attacks to form the data sets, 10 types of which were used as base set and 4 types as support set and query set. Then, using the task generating Algorithm 1 to generate 2-way K-shot tasks from each sets. The number of samples in the CICIDS2017FS is shown in Table 1.

CM2021FS

This dataset was collected from a week of real server, which contains nearly 160,000 samples in 6 different types of attacks and normal requests. The 6 classes are respectively divided into 3 (Cloud Server request, blocked IP, crawler tool) and 3 (SQL injection, Directory traversal, XSS cross-site) for pretraining tasks and meta-learning tasks. We create 2-way K-shot learning models on this dataset. The number of samples in the CM2021FS is shown in Table 2.

Implementation details

Experiment platform

The experiments were carried out under the following hardware and software environment: Intel Core i9-10920X @3.50 GHz, 64 GB RAM, NVIDIA GeForce RTX 3090; CUDA 10.2, cuDNN 8.0, and PyTorch 1.10.2.

Architecture

We use Word2Vec [30] to pretrain the language coding layer on dataset $\mathcal{D}_{\text{base}}$, where the dimension is 300 and the window size is 64. In encoder module, we set the dimension of hidden state of LSTM to 128 and the dimension of attention matrix to 64. In routing module, the iteration number *iter* is 4. In relation module, the output dimension is 100, and the activation function is Relu. After that, it is transformed into a score of [0,1] through a sigmoid layer.

Training details

In the pretraining stage, we pretrain the word embedding on base dataset $\mathcal{D}_{\text{base}}$ with Word2Vec [34]. In the meta-training stage, we train the CharNet with 10,000 episodes on the support set of train

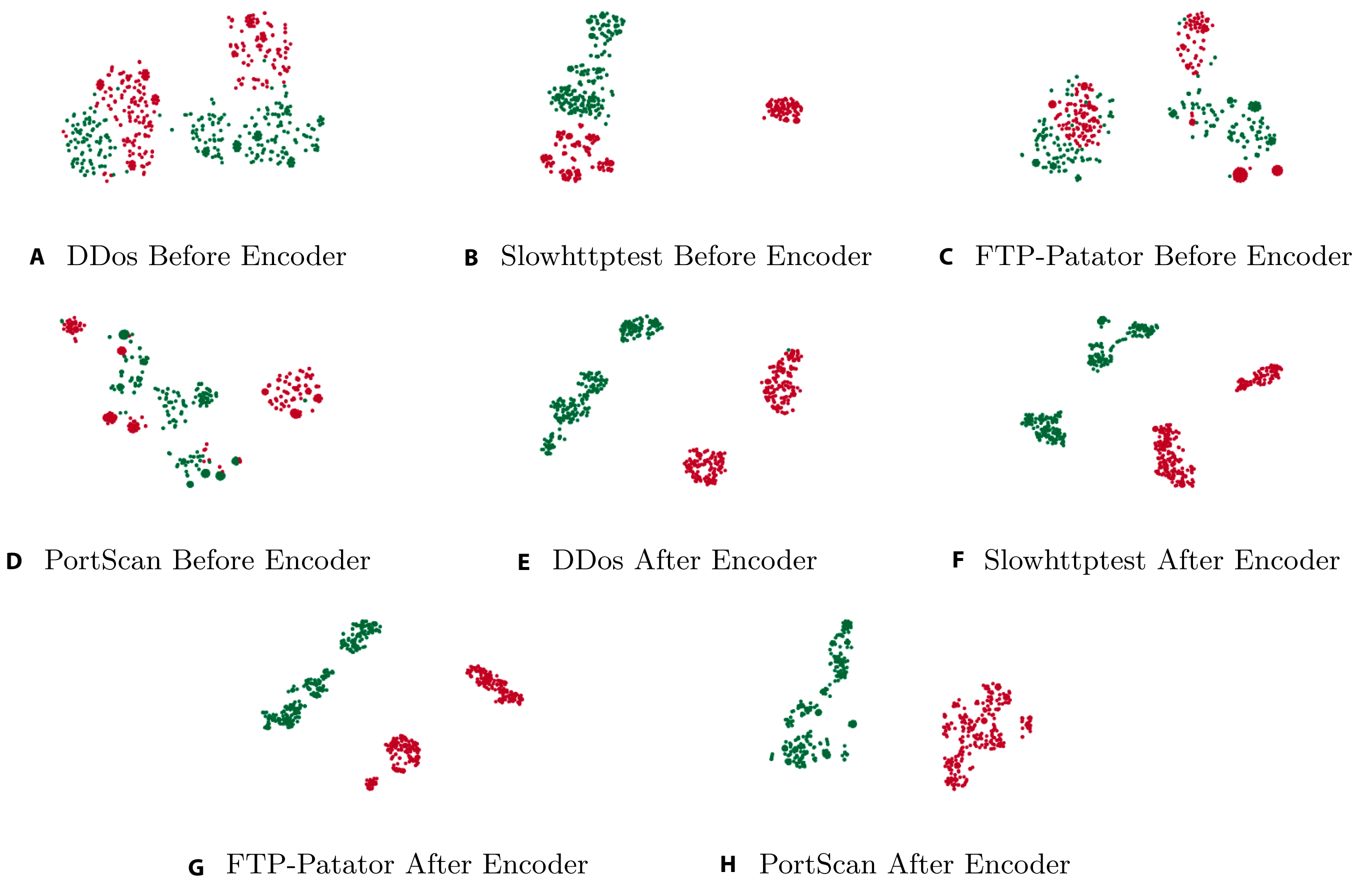

Fig. 4. Effect of Encoder under CICIDS2017FS (the green dots indicate normal traffic, and the red dots indicate abnormal traffic). (A) DDos before Encoder. (B) Slowhttptest before Encoder. (C) FTP-Patator before Encoder. (D) PortScan before Encoder. (E) DDos after Encoder. (F) Slowhttptest after Encoder. (G) FTP-Patator after Encoder. (H) PortScan after Encoder.

Table 6. Ablation study with routing module on 2-way 10-shot tasks

	iter	CICIDS2017FS		CM2021FS	
Routing module		Acc (%)	Recall (%)	Acc (%)	Recall (%)
Routing	1	97.41	98.78	93.94	97.56
Routing	2	97.52	98.89	95.78	98.75
Routing	3	98.52	98.64	97.62	98.24
Routing	4	99.87	99.85	98.34	98.92
Routing	5	98.97	99.37	98.56	99.13
Routing	6	98.02	98.67	97.92	98.36
Mean	-	98.89	99.12	96.98	97.38

Table 7. Ablation study with relation module on 2-way 10-shot tasks

	CICIDS2017FS		CM2021FS	
Relation module	Acc (%)	Recall (%)	Acc (%)	Recall (%)
Relation	99.87	99.85	98.56	98.92
Cosine	96.46	98.69	97.98	98.88
Euclidean	97.04	99.24	97.52	98.58

task set $\mathcal{T}_{\text{train}}\langle\mathcal{S}\rangle$ in an episodic manner via a stochastic gradient descent with momentum of 0.9 and weight decay of 0.0005. Then, we chose the model with the highest accuracy in the query set of train task set $\mathcal{T}_{\text{train}}\langle\mathcal{Q}\rangle$ as the final model. In the meta-test stage, We build 2-way K-shot ($K = [5, 10]$) models on 2 datasets to simulate the scenario of the network AD.

Discussion of results

In this section, we provide an overview of several most recent deep learning and machine learning algorithms in network AD and discuss the detection results and the number of samples. Then, we compare the proposed routing network with other few-shot learning methods. After that, the ablation study is conducted to evaluate the effect of each module.

Comparison with big-data methods

In previous studies, many scholars have achieved excellent performance using machine learning and deep learning algorithms

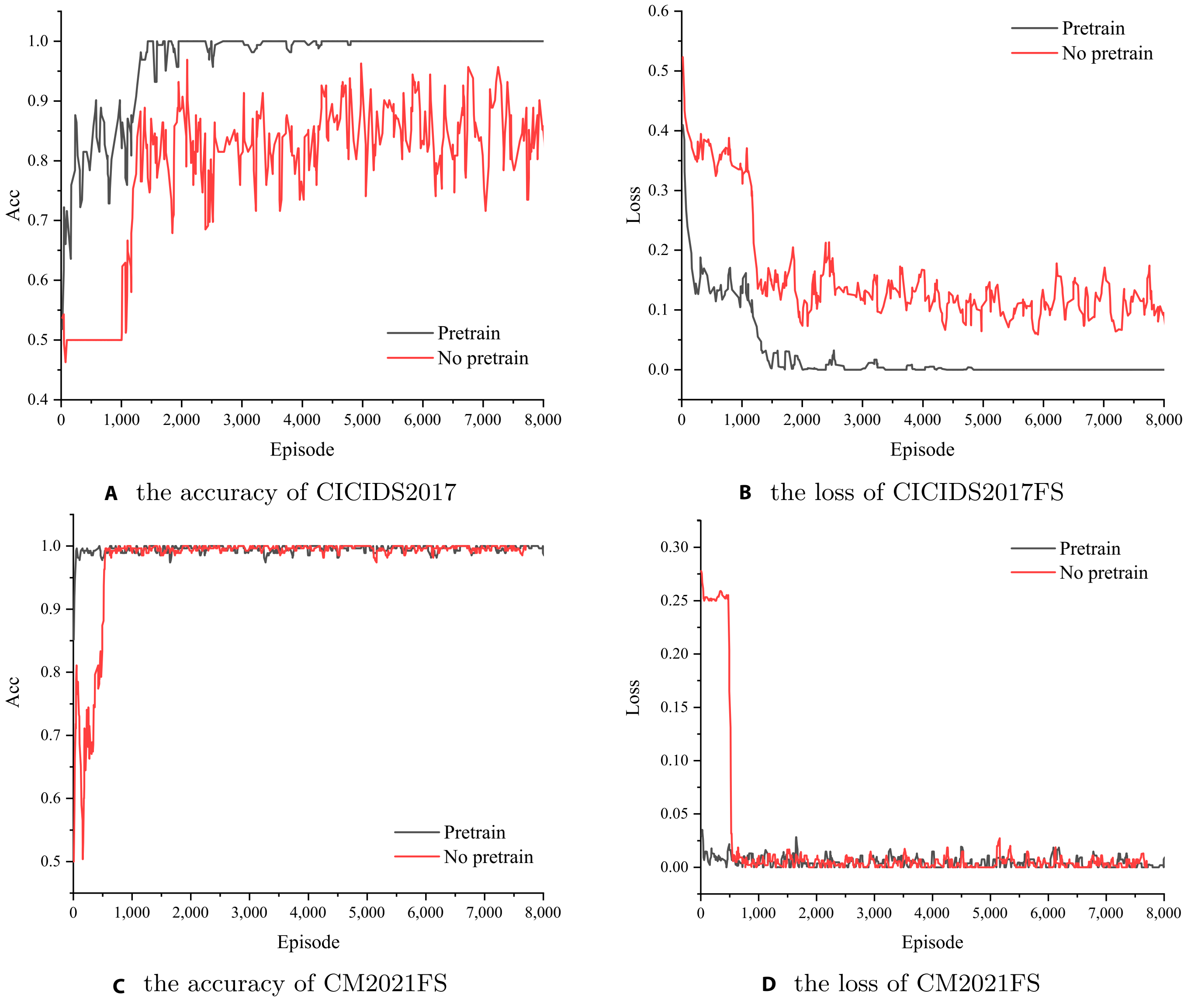

Fig. 5. Effect of pretraining under CICIDS2017FS and CM2021FS. (A) The accuracy of CICIDS2017. (B) The loss of CICIDS2017F. (C) The accuracy of CM2021FS. (D) The loss of CM2021FS.

based on large amounts of data. To ensure the fairness of the experiments, we made a comparison between CharNet with other existing researches that used the same public benchmark dataset CICIDS2017. At the same time, we also conduct few-shot experiments on the private dataset CM2020FS, the precision and recall were 94.29% and 96.56% for $K = 5$, and 98.56% and 97.68% for $K = 10$. The results are shown in Table 3.

The noteworthy observation is that the overwhelming majority of researches, whether machine learning or deep learning, are based on "big data". As can be seen from Table 3, the accuracy and recall of these researches are almost above 95%, for example, Multi-Stage Optimized ML-based AD (2021) [4] even achieves 99.99% accuracy and 99.00% recall. However, their sample size has reached hundreds of thousands, or even millions, which requires tremendous human efforts to collect, process and label these data manually. In addition, it is difficult for us to quickly identify new and ever-changing attacks.

As shown in Table 3, CharNet ($K = 5$) only outperforms GA-based Adaptive Method [35], Multilayer ensemble SVM [5], and SU-AD [36] in both accuracy and recall, while CharNet ($K = 10$) outperforms all methods except GBT-based Big Data Method [37], Improved AdaBoost-based AD [6] and Deep Hierarchical AD [38]. With a slight decline about 4% and 0.8% in accuracy and recall ($K = 5$), and about 0.1% and 0.02% in accuracy and recall ($K = 10$), required training samples of CharNet is much fewer than these method. It should be noted that the result obtained by CharNet is on the basis of only 5 and 10 labeled samples used in training process, which drastically reduces the cost of data collection and manual labeling.

Comparison with few-shot learning methods

We compare CharNet with some state-of-the-art metric-based approaches of network AD, such as FC-Net, DF-Net, GP-Net, and FS-AD. FC-Net [18] is based on RelationNet to implement few-shot traffic classification and achieves good performance in its experimental setting. DF-Net [23] takes use of siamese capsule network for AD with imbalanced traning data. A relative position mechanism and a global-enhanced feature extractor are designed in GP-Net [26] to capture the relationship between arbitrary 2-byte payload sequences. FS-AD [27] adopts autoencoder, CNN, and euclidean distance metric module in series in the few-shot frame.

To ensure fairness, we compare the performance of our method and the baseline methods on the same benchmark dataset CICIDS2017FS, the result is shown as Table 4. In order to be consistent with the setting of FC-Net, we did 2 experiments with $K = 5$ and $K = 10$. It can be found that our proposed network achieves the highest precision and recall in both $k = 5$ and $k = 10$ experiments. In the 2-way 5-shot experiment, the accuracy of CharNet exceeds the state-of-the-art methods by 1.3%~2.34%, and the recall rate exceeds 0.18%~4.55%. In the 2-way 10-shot experiment, the accuracy exceeds the advanced method by 2.36%~4.48%, and the recall rate exceeds 0.81%~3.58%. It is worth noting that our method beats FC-Net, which demonstrates that the routing-based prototype is more effective than the mean-based prototype.

Ablation study

The experiments are conducted on CICIDS2017FS and CM2021FS to evaluate the effect of each module, i.e., encoder module, routing module, relation module, and pretraining. Specifically, (a) we change the backbone network of the encoder to explore the best feature extractor; (b) for the routing module, we change the number of iterations *iter* of the routing algorithm to find the best routing-based prototype in different datasets and also compare with the mean-based prototype. (c) We replace the relation module with cosine distance metric and euclidean distance metric.

The effect of encoder module

In the experiment of the encoder module, the different backbone network are applied, such as biLSTM, LSTM, CNN, and Tranformer. The results are shown in Table 5. It can be found that (a) the accuracy of biLSTM with attention on both datasets outperform the other 3 backbone networks, while the transformer is not too far apart, whose recall achieves the best results on CICIDS2017FS; (b) the attention mechanism plays a role and helps biLSTM improve the accuracy and recall by around 1%; (c) LSTM with attention is around 4% in precision and 3% in recall lower than biLSTM with attention, which means that the informal text such as network traffic requires bidirectional semantics to better express the information in it.

In Fig. 4, it shows the t-stochastic neighbor embedding [39] visualization before and after encoder module. We carry out the test task of CICIDS2017FS, which includes 4 categories: DDos, Slowhttptest, FTP-Patator, and PortScan. We can see that the vectors after encoder are more separable, demonstrating the effectiveness of encoder to separate the solution space.

The effect of routing module

To explore the effect of iterations *iter* on the routing module, we set *iter* from 1 to 6 on CICIDS2017FS and CM2021FS. In addition, the routing module is removed and replaced the mean-based prototype. According to the result in Table 6, we observe that (a) the best performance is achieved when we used 4 and 5 iterations, and more rounds of iterations did not further improve the performance; (b) the best performance of routing module exceeds the mean-based prototype by around 1%, which indicates that the routing module shows effectiveness.

The effect of relation module

For the relation module, we draw on the idea of relation network [21] and use neural network training to obtain a learnable nonlinear similarity measure function, thereby constructing an end-to-end network structure. The experimental results of the relaton module are shown in Table 7, from which we find that the relation module outperform the cosine and euclidean distance metric on both datasets.

The effect of pretraining

To pursue faster training speed, we consider adding pretraining before the encoder module. Figure 5 shows the accuracy and loss curves with and without pretraining in CICIDS2017FS and CM2021FS, where *iter* is set to 4 and 5 respectively, and *episode* is set to 8,000. It can be seen intuitively that the convergence can be faster with pretraining, indicating that the pretraining can effectively extract the text information in the traffic log.

Conclusion

In this paper, we propose the CharNet, a novel neural model for few-shot network AD prototype. For this purpose, a basic binary classification task was defined and a pair of network traffic samples including a normal unaffected sample and a malicious one were constructed for learning. The routing module combines the dynamic routing algorithm with a meta-learning prototype, and the routing mechanism finds a more accurate prototype through multiple iterations, making our model more general to recognize unseen classes. The experiment results show that the proposed model outperforms the existing state-of-the-art few-shot network AD models, and only need 5 or 10 samples to get a high-accuracy model. We found that both the pretraining and encoder contribute tremendously to the few-shot learning tasks.

Acknowledgments

Funding: This work was supported in part by Beijing Natural Science Foundation under Grant 4222055, in part by the National Natural Science Foundation of China under Grant 62122087, 62073316, and 62373051.
Competing interests: The authors declare that they have no competing interests.

Data Availability

The open dataset CICIDS2017FS can be accessed through the following link: https://www.unb.ca/cic/datasets/ids-2017.html.

References

1. Tian Y, Liao H, Xu J, Wang Y, Yuan S, Liu N. Unsupervised spectrum anomaly detection method for unauthorized bands. *Space Sci Technol*. 2022;2022:9865016.
2. Min E, Long J, Liu Q, Cui J, Chen W. TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest. *Secur Commun Netw*. 2018;2018:4943509.
3. Liu R, Ren C, Fu M, Chu Z, Guo J. Platelet detection based on improved YOLO_v3. *Cyborg Bionic Syst*. 2022;2022:9780569.
4. Injadat M, Moubayed A, Nassif AB, Shami A. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Trans Netw Serv Manag*. 2021;18(2):1803–1816.
5. Marir N, Wang H, Feng G, Li B, Jia M. Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark. *IEEE Access*. 2018;6:59657–59671.

6. Yulianto A, Sukarno P, Suwastika NA. Improving AdaBoost-based intrusion detection system (IDS) performance on CICIDS 2017 dataset. *J Phys Conf Ser*. 2019;1192:Article 012018.
7. Markel Z, Bilzor M. Building a machine learning classifier for malware detection, *Proceedings of the 2014 Second Workshop on Anti-malware Testing Research (WATeR)*; Canterbury, UK; 23 October 2014; pp. 1–4.
8. Al-Yaseen WL, Othman ZA, Nazri MZA. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Syst Appl*. 2017;67:296–303.
9. Liu J, Gao Y, Hu F. A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm. *Comput Secur*. 2021;106:Article 102289.
10. Min B, Yoo J, Kim S, Shin D, Shin D. Network anomaly detection using memory-augmented deep autoencoder. *IEEE Access*. 2021;9:104695–104706.
11. Andresini G, Appice A, Malerba D. Nearest cluster-based intrusion detection through convolutional neural networks. *Knowl-Based Syst*. 2021;216:Article 106798.
12. Zheng F, Yan Q, Leung VC, Yu FR, Ming Z. HDP-CNN: Highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection. *Comput Secur*. 2022;114:Article 102584.
13. Shi Z, Wang T, Huang Z, Xie F, Song G. A method for the automatic detection of myopia in optos fundus images based on deep learning. *Int J Numer Methods Biomed Eng*. 2021;37(6):Article e3460.
14. Pekta A, Acarman T. A deep learning method to detect network intrusion through flow-based features. *Int J Netw Manag*. 2019;29(3):e2050.
15. Kim J, Kim J, Kim H, Shim M, Choi E. Cnn-based network intrusion detection against denial-of-service attacks. *Electronics*. 2020;9(6):916.
16. Peng W, Kong X, Peng G, Li X, Wang Z. Network intrusion detection based on deep learning, *Proceedings of the 2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*; Haikou, China; 5–7 July 2019; pp. 431–435.
17. Imrana Y, Xiang Y, Ali L, Abdul-Rauf Z. A bidirectional lstm deep learning approach for intrusion detection. *Expert Syst Appl*. 2021;185:Article 115524.
18. Xu C, Shen J, Du X. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans Inf Forensics Secur*. 2020;15:3540–3552.
19. Vinyals O, Blundell C, Lillicrap T, Wierstra D. Matching networks for one shot learning. *Proceedings of the 30th International Conference on Neural Information Processing System*; December 2016; pp. 3637–3645.
20. Snell J, Swersky K, Zemel R. Prototypical networks for few-shot learning. *Proceedings of the 31st International Conference on Neural Information Processing Systems*; December 2017; pp. 4080–4090. (ICCCS 2018); 27-30, 2018 Apr 27-30; Nagoya, Japan.
21. Sung F, Yang Y, Zhang L, Xiang T, Torr PH, Hospedales TM. Learning to compare: Relation network for few-shot learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; Salt Lake City, UT, USA; 18–23 June 2018; pp. 1199–1208.
22. Geng R, Li B, Li Y, Zhu X, Jian P, Sun J. Induction networks for few-shot text classification. arXiv. 2019. https://doi.org/10.48550/arXiv.1902.10482
23. Wang Z-M, Tian J-Y, Qin J, Fang H, Chen L-M. A few-shot learning-based siamese capsule network for intrusion detection with imbalanced training data. *Comput Intell Neurosci*. 2021;2021:7126913.
24. Ye T, Li G, Ahmad I, Zhang C, Lin X, Li J. FLAG: Few-shot latent dirichlet generative learning for semantic-aware traffic detection. *IEEE Trans Netw Serv Manag*. 2021;19(1):73–88.
25. Rong C, Gou G, Hou C, Li Z, Xiong G, Guo L, UMVD-FSL: Unseen malware variants detection using few-shot learning. *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*; Shenzhen, China; 18–22 July 2021, pp. 1–8.
26. Guo J, Cui M, Hou C, Gou G, Li Z, Xiong G, Liu C. Global-aware prototypical network for few-shot encrypted traffic classification. *Proceedings of the 2022 IFIP Networking Conference (IFIP Networking)*; Catania, Italy; 13–16 June 2022; pp. 1–9.
27. Yang J, Li H, Shao S, Zou F, Wu Y. FS-IDS: A framework for intrusion detection based on few-shot learning. *Comput Secur*. 2022;122:Article 102899.
28. Yu Y, Bian N. An intrusion detection method using few-shot learning. *IEEE Access*. 2020;8:49730–49740.
29. Zhan G, Wang W, Sun H, Hou Y, Feng L. Auto-CSC: A transfer learning based automatic cell segmentation and count framework. *Cyborg Bionic Syst*. 2022;2022:9842349.
30. Mikolov T, Chen K, Corrado G, Dean J, Efficient estimation of word representations in vector space. arXiv. 2013. https://doi.org/10.48550/arXiv.1301.3781
31. Lin SZ, Shi Y, Xue Z. Character-level intrusion detection based on convolutional neural networks. *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*; Rio de Janeiro, Brazil; 8–13 July 2018; pp. 1–8.
32. Bai D, Liu T, Han X, Yi H. Application research on optimization algorithm of sEMG Gesture recognition based on light CNN+LSTM model. *Cyborg Bionic Syst*. 2021;2021:9794610.
33. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*; 2018; pp. 108–116.
34. de Souza CA, Westphall CB, Machado RB, Sobral JBM, dos Santos Vieira G. Hybrid approach to intrusion detection in fog-based IoT environments. *Comput Netw*. 2020;180:Article 107417.
35. Rong X. word2vec parameter learning explained. arXiv. 2014. https://doi.org/10.48550/arXiv.1411.2738
36. Faker O, Dogdu E. Intrusion detection using big data and deep learning techniques. *Proceedings of the 2019 ACM Southeast Conference*; April 2019; pp. 86–93.
37. Ahmim A, Maglaras L, Ferrag MA, Derdour M, Janicke H. A novel hierarchical intrusion detection system based on decision tree and rules-based models. *Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*; 29–31 May 2019; Santorini, Greece; pp. 228–233.
38. Min E, Long J, Liu Q, Cui J, Cai Z, Ma J. SU-IDS: A semi-supervised and unsupervised framework for network intrusion detection. Paper presented at: International Conference on Cloud Computing and Security; 2018.
39. van der Maaten L, Hinton G. Visualizing data using t-SNE. *J Mach Learn Res*. 2008;9(86):2579–2605.
40. Resende PAA, Drummond AC. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *Secur Priv*. 2018;1(4):Article e36.
41. Zhang Y, Chen X, Jin L, Wang X, Guo D. Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access*. 2019;7:37004–37016.
42. Zhang X, Shenglin Y. Intrusion detection model of random attention capsule network based on variable fusion. *J Commun*. 2020;41(11):160.