Few-Shot Abnormal Network Traffic Detection Based on Multi-scale Deep-CapsNet and Adversarial Reconstruction

Wengang Ma¹ · Yadong Zhang¹ · Jin Guo¹ · Qian Yu¹

Received: 11 May 2021 / Accepted: 2 November 2021
© The Author(s) 2021

Abstract

Detecting various attacks and abnormal traffic in the network is extremely important to network security. Existing detection models used massive amounts of data to complete abnormal traffic detection. However, few-shot attack samples can only be intercepted in certain special scenarios. In addition, the discrimination of traffic attributes will also be affected by the change of feature attitude. But the traditional neural network model cannot detect this kind of attitude change. Therefore, the accuracy and efficiency of few-shot sample abnormal traffic detection are very low. In this paper, we proposed a few-shot abnormal network traffic detection method. It was composed of the multi-scale Deep-CapsNet and adversarial reconstruction. First, we designed an improved EM vector clustering of the Deep-CapsNet. The attitude transformation matrix was used to complete the prediction from low-level to high-level features. Second, a multi-scale convolutional capsule was designed to optimize the Deep-CapsNet. Third, an adversarial reconstruction classification network (ARCN) was proposed. The supervised source data classification and the unsupervised target data reconstruction were achieved. Moreover, we proposed an adversarial training strategy, which alleviated the noise interference during reconstruction. Fourth, the few-shot sample classification were obtained by combining multi-scale Deep-CapsNet and adversarial reconstruction. The ICSX2012 and CICIDS2017 datasets were used to verify the performance. The experimental results show that our method has better training performance. Moreover, it has the highest accuracy in two-classification and multi-classification. Especially it has good anti-noise performance and short running time, which can be used for real-time few-shot abnormal network traffic detection.

Keywords Abnormal network traffic · Few-shot sample · Multi-scale Deep-CapsNet · ARCN · Adversarial training

1 Introduction

The problem of network security has become increasingly prominent as the network brings us a richer and faster life. Network security risks pose new challenges to information security in the politics, economy, culture, and society. It is composed of the confidentiality, integrity, and availability of carried information. Any attempt to destroy these features

can be considered network intrusion. Meanwhile, network security is relative rather than absolute, and network intrusions exist objectively. Therefore, enterprises and individuals must detect network intrusions. Detecting various attacks and abnormal traffic in the network is extremely important to network security. We must study the security threats and various malicious attacks that the network faces. In the end, accurately detecting various types of attacks and providing effective processing solutions have become important research directions to ensure network security. However, traditional security techniques are difficult to apply to the current complex networks. Therefore, network security issues are more prominent. As an important method in network intrusion detection, the network abnormal traffic detection has become the focus of current network security research. Network managers can effectively identify various network attacks through network abnormal traffic detection. Moreover, it allows network managers to use effective methods for network intrusion prevention.

✉ Yadong Zhang
ydzhang@home.swjtu.edu.cn

Wengang Ma
1248564936@qq.com

Jin Guo
jguo_scce@swjtu.edu.cn

Qian Yu
qianyu@my.swjtu.edu.cn

¹ School of Information Science and Technology, Southwest Jiao Tong University, Chengdu 611756, China

However, the existing detection methods [1–3] have many problems in the abnormal network traffic detection. For instance, the attack traffic has many features, strong uncertainty, strong concealment, and a wide range of attacks. Existing detection methods [3, 4] often only extract a single traffic feature. It is difficult to accurately extract high-level key features of attacks from large-scale traffic in real-time. As a commonly used technique in the feature extraction process, a convolutional neural network (CNN) [5–7] has achieved better results in the feature extraction process. However, the limitations of its own structure cannot be ignored. Its main limitations are as follows: first, CNN has a strong dependence on the number of samples. Some experimental results [5] show that the more training samples and the fewer target categories, the higher the classification accuracy that CNN can achieve. When dealing with few-shot sample datasets and multi-classification tasks, CNN is prone to overfitting due to the high model complexity and few learnable features. Second, there is no structure for encoding spatial hierarchical information in CNN [6]. Therefore, it is not good at learning the spatial relationship between part and part, part and whole. If we only rely on the pooling operation and the fully connected layer, we can only maintain invariance for small-amplitude translation transformations. When the position of traffic features is changed, network traffic attributes will also be changed. Moreover, this change is called a feature attitude transition in the feature position. Therefore, changes in network attributes brought by the feature attitude of network traffic must be taken into account. Third, CNN is vulnerable to adversarial samples [7], and there are major security flaws. When there is a slight disturbance on the sample, it may interfere with the training network. Aiming at the limitations of CNN, Sabour et al. [8] proposed the concept of Caps Net. It replaced the scalar neuron of CNN with vector neuron-capsule. It used dynamic routing instead of CNN pooling to represent the part-whole relationship of the data. Therefore, it overcame the shortcomings of CNN's low robustness to changing directions and positions. A capsule contains a group of neurons. Therefore, when constructing more advanced features, it can use vectors to represent the various attributes of the entities in the feature attitude. The direction of the vector represents certain attributes of the entity. When dealing with the problem of attack detection, the capsule structure increases the capacity of feature representation. It can more accurately deal with the complicated spatial position relationship. And then, it improves the model's ability to recognize position information. Therefore, we used the capsule network as a framework for feature extraction in this paper.

Furthermore, the current detection methods [8–11] based on deep learning require a large number of attack samples for training. In many special scenarios, security agencies can only intercept few-shot attack samples. This kind of

attack is called a few-shot attack. Therefore, the accuracy is low when the few-shot attack is detected. For instance, the few-shot attack samples can only be obtained by security agencies in a zero-day attack. Unfortunately, there are little researches on the few-shot abnormal traffic detection. However, the few-shot sample attack detection can theoretically be attributed to the few-shot sample classification problem. After referencing a large amount of literature, we used the sample augmentation classification method to solve this problem in this paper. The auxiliary data or information is used to expand the original sample, which is composed of data expansion and feature augmentation. The former is to add new data, and the data can be generated by unlabeled or synthesized labeled data. The latter is to add some features that are convenient for classification in the space.

In summary, we used the capsule network to recognize the attitude change in the traffic feature. Moreover, a new classification network is designed based on the DRCN [12] network. It is called multi-scale Deep-CapsNet and adversarial reconstruction, which is used for few-shot sample attack detection. Compared with the current detection model, our method has the following highlights and contributions:

- The iterative routing is optimized by improved EM vector clustering of the Deep-CapsNet. The attitude transformation matrix is used to complete the prediction from low-level features to high-level features. Moreover, a three-dimensional multi-scale convolutional capsule is designed to optimize the Deep-CapsNet. In the end, the multi-level features of the few-shot abnormal traffic are effectively extracted.
- After DRCN is analyzed, an ARCNet is proposed. The loss function of RCN is updated by ARCNet. It is used to reconstruct the high-order inconsistency between the generated data and the original data. Moreover, adversarial training is added to alleviate noise interference during data reconstruction. In the end, the few-shot abnormal traffic detection is stable.
- We verify the effectiveness of our method in few-shot sample attack scenarios on three datasets. The results show that our method is feasible and effective in few-shot abnormal traffic detection. It has higher detection accuracy and better efficiency.

This paper is organized as follows: the purpose of the research and the ideas are introduced in Sect. 1. The abnormal traffic detection, CapsNet network, and the few-shot sample classification method are introduced in Sect. 2. In Sect. 3, the concrete theory and implementation method of multi-scale Deep-CapsNet are introduced. In Sect. 4, the specific process of ARCNet is implemented. In Sect. 5, a series of comparative experiments are set up to verify the

performance of our method. Moreover, we concluded this paper and put forward our next research plan in Sect. 6.

2 Related Works

In Sect. 1, we have explained that we proposed our method based on the capsule network and sample augmentation classification method. And it is classified in a few-shot sample scenario. Therefore, we divided the related work into three sections to describe. Section 2.1: abnormal traffic detection, Sect. 2.2: feature extraction based on capsule network, Sect. 2.3: few-shot sample classification based on sample augmentation.

2.1 Abnormal Traffic Detection

Network intrusion detection can be defined as a system that the classification of abnormal traffic is achieved. But the premise is that a large number of samples have been obtained. The two-classification model can be divided into normal and attack traffic to achieve intrusion detection. The detection process generally consists of two steps: feature extraction and classification. For instance, Hamed et al. [2] proposed a coded payload method to achieve intrusion detection based on recursive feature addition (RFA). The higher test accuracy is achieved on the ISCX2012 dataset. Wang et al. [3] proposed a HAST-IDS system in which traffic features are divided into time and spatial features. The former is extracted by CNN, and the latter is extracted by a recurrent neural network (RNN). An accuracy rate of 90.23% is achieved on the ISCX2012 dataset. Min et al. [4] proposed a feature TR-IDS system that combines statistical features and CNN extraction features. An accuracy rate of 92.31% is obtained on the ISCX2012 dataset. In 2017, Vinayakumar et al. [13] also used the CNN for intrusion detection, and the traffic is modeled as a time series. The supervised learning method is used to model the TCP/IP protocol packet within a predefined time range, and the effectiveness is proved on the KDD99 dataset. In addition, Wang et al. [5] first converted the original network traffic into images, and botnet traffic is classified through a CNN. But there is a huge difference between network traffic and images. Therefore, simply visualizing traffic to simulate image classification tasks cannot meet the requests of practical applications. Furthermore, there are many types of such research, such as autoencoders [9], long short-term memory networks, support vector machines [10], restricted Boltzmann machine [11], Earth Mover distance, Hybrid neural network, and other models [8]. Moreover, the newer CICIDS2017 dataset is also used for detection in addition to ISCX2012. The conclusions can be drawn: the attack traffic can be accurately identified as long as there are a large number of samples. However, there

are very few traffic samples that can be obtained in certain scenarios. The traffic has many high-level semantic features in a complex environment, which is affected by its feature attitude in the discrimination. The current abnormal traffic detection method based on deep learning is no longer suitable for a few-shot sample. Therefore, a few-shot sample learning method and a feature extraction method that considers the feature attitude needs to be constructed. The following content will describe the existing related work with the capsule network and the few-shot sample classification based on sample augmentation.

2.2 Feature Optimal Extraction Based on Capsule Network

Caps Net was proposed by Geoffrey Hinton to solve the inherent shortcomings of CNN. In related experiments, it is proved that this structure can achieve a better feature extraction. Moreover, it has also been verified to consider the change of feature attitude during feature extraction. But most capsules cannot share weights in dynamic routing, and the optimization ability of dynamic routing is weak.

Meanwhile, the capsule network has not yet been widely applied to network traffic detection due to the limited amount of calculation. But it has excellent performance, which is applied to optimization tasks. For instance, multiple subspaces are used to represent the capsule, which is used as the input feature vector of the next layer. In [14], a twin network is proposed, which is called SCNet. The cosine similarity, Euclidean distance, and Manhattan distance are used to describe the similarity of the output capsule. The discarding operation is added, and the sigmoid function is used to learn the discarding probability of different capsules. Furthermore, a method of capsule pooling is proposed to study how to reduce the number of capsules [15]. Regarding the optimization of dynamic routing, LaLonde and Rajasegaran deepen the capsule network [16, 17]. All capsules are converted along the depth direction. However, the numbers of dynamic routing are increased when it is deepened. And then, the computational complexity will be greatly increased. Therefore, Rajasegaran et al. [18] proposed an improved strategy, which is to be used to reduce the routing iterations. Three-dimensional convolution is used in the routing middle layer, and parameter sharing is used to reduce the parameters. The more detailed information is captured by the newly deepened capsule network. To further optimize the performance of the capsule network, batch standardization and pooling will be added to the capsule network. The verification test is completed with the few-shot sample. When the test samples are far more than the training samples, better performance can be achieved.

In recent years, the combination of capsule network and generative adversarial network (GAN) has also received

attention. Specifically, the capsule network can be used as a discriminator in the GAN. For instance, a capsule network suitable for text classification is proposed [19]. First, the dynamic routing mechanism is optimized by the network. Then, independent categories are added to the network. Moreover, the routing parameter training process is optimized. It ensures the network is more suitable for the application environment. The experiment is tested on six different text classification datasets, and the performance of the four datasets is better than CNN.

2.3 Few-Shot Sample Classification Based on Sample Augmentation

Few-shot sample classification does not require high data volume for task objectives. A good classification effect can be obtained with only a few or even a single labeled sample. It can be divided into three mainstream methods: metric [20–22], meta-learning [23–25], and sample enhancement. In this paper, the third method is used for research. Furthermore, the current related research is described in three ways: unlabeled data augmentation, data synthesis, and feature augmentation.

The first way is that unlabeled data are used to expand the few-shot sample, and semi-supervised learning is the most representative. In 2016, Wang et al. [26] used an additional unsupervised meta-training network. It ensures that a large amount of unlabeled data is touched by multiple top-level units. The diverse sets of low-density separators are learned by these units. Boney et al. [27] used the MAML [28] model for semi-supervised learning in 2018. First, the unlabeled data is used to adjust the parameters. Then, labeled data is used to adjust the classifier parameters. Finally, better few-shot sample classification are achieved. Ren et al. [29] improved the prototype network [21] and added unlabeled data to achieve higher accuracy in a few-shot sample classification. To further optimize the semi-supervised method, direct learning is regarded as a sub-problem of semi-supervised learning for research. Therefore, Liu et al. [30] proposed a transductive propagation network. The network is divided into 4 stages: feature embedding, graph construction, label propagation, and loss calculation. All labeled and unlabeled data are mapped into the vector space. And then, the construction function is used to complete the learning. Hou et al. [31] also proposed a cross attention network based on direct push learning. The attention mechanism is used to generate cross-attention mapping pair features. Moreover, the extracted features are more discriminative.

The specific idea of the data synthesis: the training data is expanded by synthesizing new label data. The commonly used algorithms are mainly various methods for optimizing GAN. Mehrotra et al. [32] proposed a generative adversarial residual pairwise network (GAN-RPN). First, the generator

is used to provide an effective regular representation of invisible data distribution. Then, the RPN is used as a discriminator to measure the similarity of paired samples. Furthermore, a few-shot classifier GAN [33] is also proposed. Two antagonistic CNN are used to generate data and discriminate categories. The overall function is solved to obtain the maximum and minimum solutions. Antoniou et al. [34] proposed a data augmentation GAN (DAGAN) to generate new samples. The accuracy of the few-shot sample is improved. Zhu et al. [35] constructed a GAN model of spectral dimension and space spectral dimension. The generated samples are added to the classification process for training. Moreover, the results show that the few-shot classification can be effectively improved. Zhang et al. [36] constructed an unsupervised feature extraction method by combining FCN and WGAN. First, the features of the hierarchical connection discriminator and the extracted multi-convolutional features are used as the spatial spectrum features. Then, the methods of shallow and deep networks are used for comparison. Finally, the results show that it is trained with few-shot label samples. Zhong et al. [37] extended the discriminator that the network contains label and authenticity information by designing a semi-GAN. The deep features are extracted while adversarial training is added. As a result, the classification accuracy is improved. In addition to the data synthesis method, Hariharan et al. [38] proposed an improved data augmentation method. It is divided into two stages: representation learning and the few-shot sample learning. The first model is learned by a source dataset in the first stage. Moreover, a square gradient magnitude loss is proposed to improve the representation learning. Wang et al. [39] generated virtual data to expand the diversity of samples by combining meta-learning and data augmentation. The end-to-end approach is used to train the generation and classification models. Xian et al. [40] integrated an f-VAEGAN-D2 network by combining the variation encoder (VAE) with GAN. When few-shot sample learning and classification are completed, the generated sample feature space is formally expressed. The final result has good interpretability. Furthermore, Chen et al. [41] formed extended support set to interpolate the support set by using a meta-learning training set. The expanded sample is used to calculate the classification loss. And then, it is used to optimize the weight generation sub-network.

In the above method, the generated sample is utilized to enhance the sample. In addition, the diversity of samples also can be improved by enhancing the sample feature. The key to a few-shot sample classification is to obtain a feature extractor with good generalization. Dixit et al. [42] proposed the attributed guided augmentation (AGA) model. It is used to learn the mapping of synthetic data. Although the migration problem of attitude objects can be solved, the trajectory is discrete and not continuous. Therefore, Liu et al.

[43] proposed a feature transfer network (FATTEN). It is composed of an encoder and decoder. First, the features of the target data are mapped to a pair of attitude parameters. Then, the feature vector is generated by the decoder. In addition, feature extraction and few-shot sample classification are completed. Schwartz et al. [44] proposed the Delta encoder to synthesize new samples for invisible categories. The transferable variables between similar samples can be extracted by the encoder. However, the feature augmentation is too simple to significantly improve the boundary problem. Therefore, Chen et al. [45] proposed a two-way network, which is called TriNet. The data feature is enhanced through the mutual mapping. However, classification networks usually focus on the most discriminative feature areas. Other areas are ignored, and it is not conducive to network generalization. Therefore, Shen et al. [46] proposed a strategy to replace the fixed attention mechanism with an uncertain attention mechanism. The initial feature is convolved multiple times to obtain a one-dimensional feature.

In summary, the few-shot sample classification can be solved to a certain extent by the many methods. However, they have some shortcomings in the classification. For instance, the many complicated data distributions cannot be captured, and the generated features are not interpretable. Especially when there is noise interference, the reconstructed data is unreliable. Therefore, we proposed an adversarial reconstruction classification network (ARCN) to improve the above problems in this paper.

3 Multi-scale Deep-CapsNet Construction

The capsule network is a deep network that consists of stacking multiple non-linear layers. Several hidden layers are coded from the bottom to up, which is used to input data features. It is mainly composed of three layers: convolutional layer, initial capsule layer, and digital capsule layer. The parameters consist of convolutional layer weights, capsule layer weights, and capsule layer coupling coefficients. The coupling coefficient is updated through the iterative dynamic routing mechanism. And then, the attitude transformation matrix between the capsule layers need to be updated by the Margin loss function. Its expression is:

$$L_c = T_c \max(0, m^+ - \|\theta_c\|^2) + \mu(1 - T_c) \max(0, \|\theta_c\| - m^-)^2. \quad (1)$$

In the formula, c is the classification category. L_c is the loss function of the c -th category, and T_c is an indicator function. When the entity belongs to category c , the value of T_c is 1. If it does not belong to category c , the value is 0. The

output capsule of category c is represented by θ_c . Moreover, the length is the probability that the entity belongs to the category. m^+ is the upper boundary with an empirical value of 0.9. m^- is the lower boundary with an empirical value of 0.1. When the network entity is of type c , the θ_c will be greater than 0.9. When it does not belong to the c category, θ_c will be less than 0.1. Its purpose is to ensure the loss value is almost 0 when the prediction is correct. Meanwhile, the loss value is larger when the prediction is wrong. μ is an adjustment coefficient, which is used to adjust the proportion of correct and incorrect results.

Therefore, the capsule neuron carries the connection of the network weights. In addition, features are also enriched. And then, misjudgment of abnormal traffic caused by changes in network traffic attributes is avoided. But when it is used for network traffic feature extraction, the scale of the intermediate vector is too small due to insufficient network depth. Moreover, the inner product routing algorithm is inefficient. These factors reduce the speed of the network. In addition, the clustering effect of the network is affected. As a result, the extraction of traffic features is not sufficient. Therefore, multi-scale Deep-CapsNet is proposed to optimize the capsule network. Moreover, it is divided into two sections. Section 3.1: the Deep-CapsNet network is designed. It is used to replace iterative routing in the capsule network. And then, the operating efficiency is improved. Section 3.2: a multi-scale capsule structure is designed. It is used to extract high-level features in various abnormal traffic. Then, the classification performance of the model is optimized. Therefore, the implementation of Multi-scale Deep-CapsNet is described in Sects. 3.1 and 3.2.

3.1 Deep-CapsNet Network

When the target object has a complex internal representation, the category cannot be correctly predicted by the original capsule. Therefore, a Deep-CapsNet (Fig. 1) is first constructed in this paper. When the network traffic contains N labeled samples of L categories, the set can be represented by $\{x_1, x_1, \dots, x_N\} \in R^{1 \times b}$. The label vector is $Y = \{y_1, y_1, \dots, y_N\} \in R^{1 \times L}$. Moreover, the feature vector is used as the input of the capsule network. And then, the main idea of feature extraction is that convolution is used for the low-level feature. Moreover, capsules are used to characterize more advanced features. Therefore, two convolutional layers are used to convolve the capsule network. The local feature detection of network traffic is completed. The channels of the two convolutional layers are 512 and 256. The size of the convolution kernel is 9×9 and 5×5 . The convolution step length is 1. The ReLU is used as the activation function. Then a primary capsule layer is constructed, which is used to connect the convolutional layer and the capsule module. A set of $32 \times 16 \times 6 \times 6$ scalars are obtained

Fig. 1 Deep-CapsNet structure

by multiple convolutions. A group of 1152 vector neurons from 16-dimensional vectors is composed of the capsule section. Each neuron outputs a 16-dimensional vector. In each 6×6 grid, the weight is shared to each capsule, and then each output vector is output. Meanwhile, the vectors are optimized by adding clustering capsule layer 1. First, the features are preliminarily screened by the EM clustering. A high-level vector with an obvious tendency is formed for the clustering capsule layer 2. Then, the vectors that can be used to represent different information are selected. Moreover, compression is performed in each cluster layer. The clustering process ensures the high-level features are more concentrated. The compression is used to limit the length of the vector. Different from the forward and backward propagation, the transformation of the traffic features is the judgment of the attitude transformation. Furthermore, the connection between different capsule layers is completed through EM vector clustering. The last layer is the category capsule layer. The dimension of each capsule vector neuron is denoted as d_2 . The number of capsules is the corresponding category number L . Therefore, the following content mainly describes the attitude transformation and EM vector clustering.

3.1.1 Attitude Transformation

As shown in the process shown in Fig. 1, when the feature passes through two clustering capsule layers, the feature attitude will be analyzed (after the affine transformation in the figure, it is the process of feature attitude transformation.). The specific process is as follows: in the propagation between the initial capsule layer and class capsule layer,

vector neurons need to be fully connected. Therefore, an affine transformation coefficient $W_{i,j}$ is added. $W_{i,j}$ is the attitude transformation matrix of the i -capsule output from the l -level to the j -th capsule. The matrix is used to convert the vector neuron of length d_1 in the initial capsule layer (denoted as in_i). It can be multiplied with the length of the vector neuron in the capsule layer d_2 (denoted as cap). Finally, the input vector after affine transformation can be obtained as:

$$U_{i,j} = W_{i,j} in_i, \quad (2)$$

$U_{i,j}$ and cap_j are vectors of equal length. $U_{i,j}$ is regarded as the prediction vector of the j capsule in the i layer. It is the prediction result. Then all the inputs of cap_j are weighted sum. The total input s_i of each cap_j can be expressed as:

$$s_i = \sum_j c_{i,j} U_{i,j}, \quad (3)$$

$c_{i,j}$ is the coupling coefficient, which is determined by dynamic routing. Moreover, $i \in \{1, 2, \dots, (CH_2 \times N_2)\}$, $j \in \{1, 2, \dots, L\}$.

$$cap_i = \frac{\|s_i\|^2}{1 + \|s_i\|^2} \frac{s_i}{\|s_i\|}, \quad (4)$$

$$cap_i \approx s_i s_i \quad \text{when } s_i \text{ is small}, \quad (5)$$

$$cap_i \approx \frac{s_i}{s_i}, \quad \text{when } s_i \text{ is large}. \quad (6)$$

3.1.2 EM Vector Clustering Optimization

Low-level features are predicted as high-level through vector neurons in the capsule network. The distribution of the output vector conforms to the Gaussian mixture model [47] with different high-level features as expected. Therefore, EM clustering is used to replace the iterative routing, which improves operating efficiency. However, it has the following two disadvantages: first, the EM algorithm is complicated to calculate, which is not suitable for large-scale datasets and high-dimensional data. Second, the function to be optimized is not a convex function. It is prone to a locally optimal solution instead of a globally optimal solution. Therefore, the EM clustering is designed to be EM vector clustering. It is used to replace iterative routing in the capsule network. EM vector clustering corresponds to two clustering capsule layers in Fig. 1. The specific implementation process is as follows: a set of prediction vectors generated by the attitude transformation equation conform to the Gaussian mixture distribution. The distribution function with the highest probability is obtained as the capsule output, which can be expressed as:

$$p(X) = \sum_j \frac{\alpha_j}{(2\pi)^{d/2} (\det \Sigma_j)^{1/2}} \exp\left(-\frac{1}{2}(X - \mu_j)^T\right) \sum_j^{-1} ((X - \mu_j)), \quad (7)$$

represents the category, and X is the input vector. α_j is the probability of class j and $\sum_j \alpha_j = 1$ is true. μ_j is the vector expectation of the j category, and Σ_j is the covariance matrix. Because low-level features are the transformation results of traffic features, the vectors are independent of each other. Therefore, the covariance matrix is a diagonal matrix, which is equivalent to the decoupling calculation of input X . Furthermore, the formula (7) is modified, which is expressed:

$$p(X) = \sum_j \frac{\alpha_j}{\sqrt{2\pi\sigma^d}} \exp\left(-\frac{1}{2\sigma^2} \|X - \mu_j\|^2\right). \quad (8)$$

The input distribution is regarded as a mixed Gaussian distribution for clustering. Therefore, the center vector is the weighted average of the vectors within the cluster. The significance of the network cannot be measured by the model length. Therefore, a scalar a_j is added as a scaling scale to measure the significance. The output is substituted into the squashing function. It is used to control the length of the output vector. The EM clustering result is used to obtain the variance of the output Gaussian distribution. The larger the variance, the closer the prediction vector distribution is to a uniform distribution. It further shows that the prediction result is not close to the same feature, and a_j should be small.

Therefore, the information entropy C_j is selected to assist a_j , and C_j is expressed as:

$$C_j = \left(1 + \sum_{l=1}^d \ln \sigma_j^l\right) \sum_j r_{ij}, \quad (9)$$

σ_j^l is the mean square error of the l -th layer in the j capsule. r_{ij} is the ratio of the bottom layer i capsule transformed into the j capsule. C_j is the distinctiveness, which is expressed by the j capsule. The scaling scale a_j is obtained through C_j , which can be expressed as:

$$a_j = \text{sigmoid}(1 - C_j). \quad (10)$$

When the distribution variance is smaller, the C_j value is smaller. Therefore, iterative optimization is achieved by maximizing $-C_j$. When $U_{i,j}$, S_j , and a_j are given, $U_{i,j}$ represents a vector prediction. Specifically, it is expressed as the i capsule is output to the j high-layer capsule. S_j represents the output direction of the $l+1$ layer capsule. σ_j^2 is the variance of the output direction in the $l+1$ layer capsule. a_j represents the prominence of the features. The EM vector clustering algorithm is as follows:

Algorithm 1 EM vector clustering algorithm

The variables $U_{i,j}$, a_j , S_j , and σ_j^2 are initialized;

For Low-level capsules i to high-grade capsules j ;

Step 1 $p_{ij} \leftarrow N(U_{i,j}; S_j; \sigma_j^2)$; $R_{ij} \leftarrow \frac{a_j p_{ij}}{\sum_{j=1}^k a_j p_{ij}}$;

$r_{ij} \leftarrow \frac{\|U_{i,j}\| R_{ij}}{\sum_{i=1}^n \|U_{i,j}\| R_{ij}}$;

Step 2 $S_j \leftarrow \sum_{i=1}^n r_{ij} U_{i,j}$; $\sigma_j^2 \leftarrow \sum_{i=1}^n r_{ij} (U_{i,j} - S_j)^2$;

Step 3 $C_j \leftarrow \sum_i (1 + \sum_{l=1}^d \ln \sigma_j^l) r_{ij}$; $a_j \leftarrow \text{sigmoid}(1 - C_j)$;

Therefore, the process of the EM vector clustering algorithm is as follows: first, each prediction vector is weighted by using the length of the input vector. The weighted probability r_{ij} of an advanced capsule is obtained. Second, the parameters S_j and σ_j^2 of the Gaussian mixture model are obtained by the calculated r_{ij} and the predicted vector distribution $U_{i,j}$. After multiple iterations, the probability a_j is used as the output scale of the j capsule. The output direction vector is scaled to obtain the final output vector:

$$V_j = \text{squashing}(a_j, S_j), \quad (11)$$

$\text{squashing}(\cdot)$ represents compression function.

3.2 Multi-scale Capsule Optimization

To solve the problem of insufficient network parameter training, a multi-scale capsule (Fig. 2) is designed to optimize DeepCapsNet. The commonality and features of various

Fig. 2 Multi-scale capsule structure

abnormal traffic are fully extracted. Moreover, the classification performance of the model is optimized. It is composed of two sections. First, structural and semantic information is obtained through a multi-scale convolution kernel. The two convolution kernels of the top branch are advanced feature extraction processes in Fig. 2. They are used to extract semantic information. The first convolution kernel of the intermediate branch is used to extract intermediate features. Meanwhile, the first convolution kernel is used to extract low-level features in the bottom branch. Then, the feature level is encoded by the multi-dimensional initial capsule. The initial capsule layers of the three branches are used to encode the high, medium, and low-level features. The

initial capsule layer can be seen as an extension of the two-dimensional convolutional layer. Therefore, the size of the two-dimensional convolution kernel is 4×4 , and the number of channels is 4. The convolution kernel has been subjected to 12, 8, and 4 convolution operations. Finally, the splicing operation is used to merge the three initial capsules. Moreover, four fusion capsules with a dimension of 24 are obtained.

To further improve the reliability of the capsule network, multi-scale feature learning is added to the training process. The multi-scale convolution kernel is used to extract the multi-scale features. Therefore, the multi-scale Deep-CapsNet is proposed, and the structure is shown in Fig. 3. It is mainly composed of a multi-layer convolution structure, multi-scale capsule extraction of deep features, and clustering capsule layers. The specific steps are as follows:

Step 1: the input traffic features are extracted by the multi-scale Deep-CapsNet. In addition, the channels of the convolution kernel are 16. This process is a preliminary feature mapping of traffic data. Therefore, the primary feature extraction is completed in a larger range of common features.

Step 2: the multi-scale capsule structure is set to include three convolution kernels with dimensions $1 \times 1 \times 3$, $1 \times 1 \times 5$, and $1 \times 1 \times 1$. The convolution kernel channels are

Fig. 3 Multi-scale Deep-CapsNet network structure

also designed to 16. It enables the different local structures is used to extract features. Moreover, the output features are output through three primary capsule networks. Finally, the three output features are spliced to fuse traffic features of different scales.

Step 3: it is called clustering capsule layer optimization. After the shallow information is extracted by a three-dimensional convolutional capsule, the optimized capsule is used to replace the iterative routing. This section is composed of two clustering capsule layers. They are output after reaching the classification capsule layer.

4 Few-Shot Sample Classification Based on ARCN

After the network traffic features are extracted by multi-scale Deep-CapsNet, a method suitable for a few-shot sample classification is proposed. Since there are few-shot network traffic samples, the feature training will fall into under-fitting. Moreover, many enhanced network structures are more

complicated. In the few-shot samples, a better classification result cannot be obtained. As a result, Ghifary et al. [12] proposed a deep reconstruction classification network (DRCN) model based on deep learning. Two tasks can be learned by DRCN with a set of shared features: supervised source data classification and unsupervised target domain data reconstruction. The extracted features can obtain information while maintaining the discrimination. However, the classification depth in DRCN is not enough. As a result, deep features cannot be effectively identified. Furthermore, the abnormal traffic of few-shot samples is susceptible to noise interference. And then, DRCN is not stable in the classification process. Therefore, an adversarial reconstruction classification network (ARCN) is proposed based on DRCN. It is composed of the reconstruction classification network (RCN) and adversarial training. In the end, the above problems are solved through these two improvement strategies. Furthermore, the implementation of ARCN is described in Sects. 4.1 and 4.2.

Fig. 4 RCN structure

4.1 RCN Network Construction

First, the reconstruction classification network (RCN) is designed. The reconstruction network comes from the DRCN network in the Fig. 4. Moreover, the classification network is designed in the paper. The specific structure of the RCN is shown in Fig. 4.

It is composed of two sections: a supervised classification network and an unsupervised reconstruction network. The two sections can be divided into the following three functions: the coding function can be represented by the overlapping section of the blue box and the red box in Fig. 4. The classification function can be represented by the remaining section of the red box in Fig. 4. The reconstruction function can be represented by the remaining section of the blue box in Fig. 4. The two sections of the RCN have a shared coding feature representation. It enables the coding function to learn the commonalities between the two tasks. First, $f_c : \alpha \rightarrow \beta_c$ is set as the supervised sample classification section. Then, $f_r : \alpha \rightarrow \alpha$ is set as the sample reconstruction section. α and β_c represent the sample space and label space. They can be defined by the mathematical model: the encoding function is described as $h_{enc} : \alpha \rightarrow \delta$. The reconstruction function is described $h_{rec} : \delta \rightarrow \alpha$. The classification function is described as $h_{cla} : \delta \rightarrow \beta_c$. δ represents the coding feature space. When an input sample $x \in \alpha$ is given, f_c and f_r can be expressed as:

$$\begin{cases} f_c(x) = (h_{cla}^\circ h_{enc})(x) \\ f_r(x) = (h_{rec}^\circ h_{enc})(x) \end{cases} \quad (12)$$

$\theta_c = \{\theta_{enc}, \theta_{cla}\}$ and $\theta_r = \{\theta_{enc}, \theta_{rec}\}$ represent the parameters of the supervised classification and the unsupervised reconstruction. θ_{enc} , θ_{rec} , and θ_{cla} are the parameters of coding, reconstruction, and similar functions. The goal of parameters is to find a set of shared coding functions h_{enc} to support f_c and f_r .

4.2 Adversarial Training

In the unsupervised reconstruction model, adversarial training is designed to optimize the influence of noise and jitter interference. Therefore, we proposed ARC for adversarial training based on RCN. The RCN model is represented by the green box in Fig. 5. The features are used to generate classification results and reconstruct samples. Moreover, the adversarial network is represented by the blue box on the right in Fig. 5.

The traffic is used to generate category labels (1 = real sample, 0 = synthetic sample). θ_a represents the adversarial network parameter. $f_a : \alpha \rightarrow \beta_a$ represents an adversarial network, where $\beta_a \in [0, 1]$ is established. The ARC process

is mainly as follows: when an input sample $x \in \alpha$ is given, f_a can be expressed as follows:

$$f_a(X) = \begin{cases} 1, X = x \\ 0, X = f_r(X) \end{cases} \quad (13)$$

When N_1 labeled training samples (x_i, y_i) are given, $y_i \in \{0, 1\}^K$ is one-hot code. Moreover, N_2 unlabeled training samples are also given, and the loss function can be defined as:

$$\begin{aligned} L_{ARC}(\theta_c, \theta_r, \theta_a) \\ = \sum_{i=1}^{N_1} l_c(f_c(x_i), y_i) - \tau l_a(f_a(f_r(x_i)), 0) \\ - \tau \left[\sum_{i=1}^{N_1} l_a(f_a((x_i), 1)) + \sum_{i=1}^{N_2} l_a(f_a((x_j), 1)) + l_a(f_a(f_r(x_i)), 0) \right]. \end{aligned} \quad (14)$$

$l_c(\hat{y}, y) = -\sum_{k=1}^K y_k \ln \hat{y}_k$ represents multiple types of cross-entropy loss, which is used to predict $\hat{y}$. All of l_a can be described as $l_a(\hat{z}, z) = -[\ln \hat{z} + (1 - z) \ln(1 - \hat{z})]$, which is a binary cross-entropy loss. The loss of parameters θ_c and θ_r related to RCN can be minimized. Moreover, the loss related to parameter θ_a of the adversarial network is maximized. Since the term containing l_a in (14) is related to the adversarial network, the loss function can be expressed as:

$$\begin{aligned} L_A(\theta_a) = \sum_{i=1}^{N_1} l_a(f_a(f_r(x_i)), 0) + l_a(f_a((x_i), 1)) \\ + \sum_{j=1}^{N_2} l_a(f_a(f_r(x_j)), 0) + l_a(f_a((x_j), 1)). \end{aligned} \quad (15)$$

The training process of the adversarial network is to minimize the loss function in (15). Under the given conditions of the adversarial network, the minimization process of the RCN model is equivalent to minimize the multi-class loss function. Therefore, the loss function of the RCN model can be expressed as:

$$\begin{aligned} L_{RCN}(\theta_c, \theta_r) = \sum_{j=1}^{N_2} -\tau l_a(f_a(f_r(x_j)), 0) \\ + \sum_{i=1}^{N_1} l_c(f_c(x_i), y_i) - \tau l_a(f_a(f_r(x_i)), 0). \end{aligned} \quad (16)$$

$-\tau l_a(f_a(f_r(x_i)), 0)$ is updated $-\tau l_a(f_a(f_r(x_i)), 1)$. The maximization adversarial model is adopted to predict $f_r(x)$ as the true sample probability. Moreover, a strong gradient signal will be generated. The experiments have proved that the update is of great significance for accelerating the training process. Therefore, the loss function of the RCN model can be updated as:

Fig. 5 ARCNet structure

$$L_{RCN}(\theta_c, \theta_r) = \sum_{j=1}^{N_2} \tau l_a(f_r(x_j), 1) + \sum_{i=1}^{N_1} l_c(f_c(x_i), y_i) + \tau l_a(f_a(f_r(x_i), 1)). \quad (17)$$

To further optimize the network, the Adam algorithm is used to alternately minimize L_A and L_{RCN} . The optimization of L_{ARCNet} is achieved, and the algorithm will stop in

two cases. First, the preset accuracy is exceeded by multiple consecutive training accuracy rates. Second, the iterations reach the maximum number. The specific implementation of ARCNet is given by algorithm 2. In the optimization of L_{RCN} , dropout regularization is used to prevent overfitting.

Algorithm 2 ARCN

Input: Labeled sample: $\{(x_i, y_i)\}_{i=1}^{N_1}$; Unlabeled sample: $\{(x_j)\}_{j=1}^2$; Learning rate: $\alpha_c, \alpha_r, \alpha_a$

- 1: Initialization parameters: $\theta_c, \theta_r, \theta_a$
- 2: **while** continue to iterate **do**
- 3: **for** each batch of labeled samples **do**
- 4: The first sub-formula of formula $\begin{cases} f_c(x) = (h_{cla} \circ h_{enc})(x) \\ f_r(x) = (h_{rec} \circ h_{enc})(x) \end{cases}$ and formula $f_a(X) = \begin{cases} 1, X = x \\ 0, X = f_r(X) \end{cases}$ are used to pass forward.
- 5: Update $\theta_c, \theta_r, \theta_a$:

$$\theta_c \leftarrow \theta_c - \alpha_c \nabla_{\theta_c} L_{RCN}(\theta_r)$$

$$\theta_r \leftarrow \theta_r - \alpha_r \nabla_{\theta_r} L_{RCN}(\theta_r)$$

$$\theta_a \leftarrow \theta_a - \alpha_a \nabla_{\theta_a} L_A(\theta_a)$$
- 6: **End for**
- 7: **for** each batch of unlabeled samples **do**
- 8: The first sub-formula of formula $\begin{cases} f_c(x) = (h_{cla} \circ h_{enc})(x) \\ f_r(x) = (h_{rec} \circ h_{enc})(x) \end{cases}$ and formula $f_a(X) = \begin{cases} 1, X = x \\ 0, X = f_r(X) \end{cases}$ are used to pass forward.
- 9: Update θ_r, θ_a :

$$\theta_r \leftarrow \theta_r - \alpha_r \nabla_{\theta_r} L_{RCN}(\theta_r)$$

$$\theta_a \leftarrow \theta_a - \alpha_a \nabla_{\theta_a} L_A(\theta_a)$$
- 10: **End for**
- 11: **end while**

Output: ARCN learning parameters: $\theta_{ARC N}^{\wedge} = \{\theta_c^{\wedge}, \theta_r^{\wedge}, \theta_a^{\wedge}\}$

Table 1 Number of each attack traffic category

Number	Attack traffic category
iA	Infiltrating the network from inside
iB	HTTP denial of service
iC	Distributed denial of service using an IRC Botnet
iD	Brute force SSH
cA	FTP-Pataor
cB	SSH-Pataor
cC	DoS (Slowloris, Slowhttptest, Hulk, GoldenEye)
cD	PortScans
cE	DDoS with LOIT

5 Experiment and Result Analysis

5.1 Experimental Dataset and Settings

Our experiment is implemented under the following platforms. Hardware: IntelXeonE5-2600V3@2.6 GHz, 256GBRAM; Software: Ubuntu16.04LTS, CUDA 10.0, cuDNN7.5 and PyTorch0.4.1. The dataset that can be used to verify the proposed model should have the following three features: first, the dataset should be the complete original network traffic. Moreover, it is not a feature vector obtained through feature extraction. Second, network traffic must have a corresponding category label. Third, network traffic should be packaged into few-shot sample learning tasks. It is used to evaluate whether the algorithm is still effective. In the field of network intrusion detection, commonly used datasets include KDD99, NSL-KDD, DEFCON, LBNL, Kyoto, AVFA, ISCX2012, and CICIDS2017, etc. However, the datasets with features (1) and (2) are ISCX2012 and CICIDS2017. As for the research in the few-shot sample scenario with feature (3), there is no relevant public dataset. To satisfy feature (3), we used the complete raw network traffic in ISCX2012 and CICIDS2017. Therefore, the complete original network traffic is used as the dataset in ICSX2012 and CICIDS2017. The intrusion detection dataset suitable for a few-shot sample environment is constructed, which is called ICSX2012FS and CICIDS2017FS. The suffix "FS" means a few-shot. Therefore, three types of experimental datasets are established.

Types I and II are single datasets, and type III is cross-dataset. Type I: ICSX2012FS dataset contains 4 types of attacks. Three types of attacks and normal traffic are selected as the training set. Moreover, the remaining type 1 attacks and normal traffic are used as the test set. The few-shot sample training and testing are constructed to perform two-classification experiments. Type II: CICIDS2017FS dataset contains 5 types of attacks. Three types of attacks and normal traffic are selected as the training set. One of the remaining two types of attacks and the normal traffic sample is used as the test set for a two-classification experiment. Type III: the data are a cross-experimental dataset for cross-network dataset detection. Five types of attacks and normal traffic are selected as the training set in the CICIDS2017FS dataset. Any type 1 attack and normal traffic in the ICSX2012FS dataset are selected as the test set for a two-classification experiment. We call this type of experimental dataset "cross dataset". The two datasets come from experimental networks that contain different hardware and software environments. And then, the types of attacks are also different. The adaptability of the proposed method can be fully verified in the few-shot sample classification. The specific explanations of various types of attacks are shown in Table 1. Moreover, a

Fig. 6 Comparison of under the three datasets

unique code is assigned to each type of attack to facilitate the experiment. The code prefix "i" indicates that the data comes from ICSX2012FS. The prefix "c" indicates that the data comes from CICIDS2017FS. Therefore, ICSX2012FS is composed of normal traffic samples and four types of attack samples (iA, iB, iC, and iD). CICIDS2017FS is composed of normal traffic samples and five types of attack samples (cA, cB, cC, cD, and cE). For each task in the training set and test set, we set $K=5, 10$ (we call it 5-shot, 10-shot). It means that the number of samples is 5 or 10 in each category. Therefore, only "a few" sample scenes are accurately simulated in the actual network.

5.2 Training Performance Experimental Verification

5.2.1 Multi-scale Deep-CapsNet Performance Verification

To verify the influence of the EM parameters in the multi-scale Deep-CapsNet, an EM parameter selection experiment is constructed. The parameter is the clustering length, which is used to measure the probability of classification. When the longer length is reached, the clustering effect is better.

Fig. 7 Comparison of performance under 5-shot

The experiment is trained on the three datasets. The clustering length is mainly tested under different R (number of clustering iterations). To further illustrate the effectiveness, the clustering length of CapsNet is also used for comparison. The experiment is performed ten times, and the specific results are shown in Fig. 6. The left side of Fig. 6a–c is the comparison of the model length, and the right side is the clustering result.

The conclusions that can be drawn from Fig. 6a–c are as follows: first, different clustering iterations have a great influence on the clustering length. Before the CapsNet is optimized by the improved EM clustering, the longest clustering length is less than 0.1. After performing three clusterings, the average clustering length reached 0.45 in the CapsNet. Therefore, it shows that CapsNet can be optimized by the improved EM clustering. Second, the clustering length will continue to be increased as the iterations are increased. When iterations are increased to 6, the clustering length will be decreased. Because the clustering effect will be affected by too many clustering iterations, the length will be reduced. Third, the clustering length of the EM clustering is significantly higher than that of routing clustering. It shows that the improved EM clustering effect is better than the traditional iterative routing. When the CapsNet is designed as

Fig. 8 Comparison of performance under 10-shot

a multi-scale Deep-CapsNet, its effect is better. The same conclusions are also obtained from the clustering result.

To evaluate the training performance of the multi-scale Deep-CapsNet, the multi-scale Deep-CapsNet (called 3DConv-Deep-CapsNet) is baselined to construct the experiment. Several classic methods are selected for comparative analysis. The comparison methods are CNN, CapsNet, Deep-CapsNet, three-dimensional convolutional neural networks (3DCNN), and three-dimensional empty spectrum capsule networks (Conv-Caps). The learning rate is set to 0.001, and the training iterations are 100. Training accuracy (ACC) and average loss rate (loss) are used to verify the effectiveness. The accuracy and loss rates of the three datasets are given under 5-shot and 10-shot, which are compared in Figs. 7 and 8. From Fig. 7a–c, the following conclusions can be drawn: first, the training accuracy of the capsule network is significantly higher than CNN. Although the accuracy of the 3DCNN is higher than that of the capsule, it is lower than other capsule optimization methods. Second, when the capsule is optimized by the improved EM vector clustering, the accuracy is significantly improved. When the Deep-CapsNet is further designed as 3DConv-Deep-CapsNet, the accuracy is the highest. The average accuracy rate reaches 0.96 in the 5-shot training task. Third, the corresponding training accuracy continues to be increased as the epochs are increased.

Fig. 9 Comparison of performance under 5-shot

However, the accuracy curves of CNN and 3DCNN have poor convergence. When the training epochs are increased, the accuracy will be decreased. Because CNN will appear gradient disappears as the training epochs are increased. Moreover, CNN will fall into a local optimal solution, and the final training performance will be affected. In contrast, the proposed 3DConv-Deep-CapsNet accuracy curve has the best convergence. When epochs are increased to 16, the accuracy rate remains stable. Therefore, the training performance of 3DConv-Deep-CapsNet is verified on the 5-shot task. The same conclusion is also obtained from the loss rate curve in Fig. 7d–f.

Correspondingly, we can draw the following conclusions from Fig. 8a–c: first, the accuracy of the capsule is also significantly higher than that of the CNN. However, the accuracy under the 10-shot is slightly higher than that of the 5-shot. Because the traffic features under 10-shot are more than 5-shot. Many types of features can be obtained, and the accuracy will be improved. Moreover, the accuracy of the 3DCNN is the same as that of the capsule in the 10-shot training task. The advantages of the capsule over CNN have been proven again. Second, when the capsule is

Fig. 10 Comparison of performance under 10-shot

optimized by the improved EM vector clustering, the accuracy is significantly improved. When the Deep-CapsNet is further designed as 3DConv-Deep-CapsNet, the accuracy is the highest. The average accuracy rate reaches 0.975 in the

10-shot training task. Third, the accuracy curves of CNN, 3DCNN, and traditional capsules have poor convergence when epochs are increased. There are some malicious traffic features in the 10-shot training. Therefore, the traffic features are misjudged by the capsule. The proposed 3DConv-Deep-CapsNet accuracy curve has the best convergence. When epochs are increased to 16, the accuracy rate remains stable. The conclusion is consistent with the 5-shot training situation, and the same conclusion is also obtained from the loss rate curve in Fig. 8d–f.

5.2.2 ARCNet Training Performance Verification

Similarly, the same experiment is constructed to verify the performance of ARCNet. In the experiment, GAN is used as the baseline to compare the training performance of GAN, RCN, DRCN, CapsNet-GAN, and ARCNet. The learning rates are also set to 0.001, and the iterations are 100. The accuracy and loss rates are given under 5-shot and 10-shot, which are compared in Figs. 9 and 10. When only the GAN is used to train the dataset, its training performance is relatively poor. The accuracy rate of GAN is only 76.5% under 5-shot. Other networks are optimized based on the GAN network, and the accuracy is better. However, the highest accuracy is achieved by the ARCNet network, which is 97.7%. Second, although it is constructed with DRCN and RCN networks, the two networks have not joined the adversarial training. The data are too noisy in the reconstruction, so the performance is average. Third, the accuracy rate continues to be increased as the epochs are increased. However, the accuracy curve of other networks has poor convergence except for ARCNet and DRCN. Moreover, when epochs are increased, the accuracy will be decreased. But the ARCNet accuracy curve has the best convergence. The same conclusion is also obtained from the loss rate curve in Fig. 9d–f.

Table 2 Comparison of accuracy of each method on ICSX2012FS dataset

Method	Method type	10-shot accuracy (%)		5-shot accuracy (%)	
		Normal	Attack	Normal	Attack
Meta LSTM [29]	Meta	59.64 ± 0.75	69.33 ± 0.18	–	–
MAML [31]	Meta	63.07 ± 0.16	74.22 ± 0.39	–	–
Matching Nets [23]	Metric	66.08 ± 0.57	76.64 ± 0.31	59.16 ± 0.43	73.05 ± 0.66
Proto Nets [22]	Metric	60.24 ± 0.35	73.25 ± 1.26	56.31 ± 0.44	70.88 ± 0.63
CapsNet-GAN [18]	Data augmentation	73.86 ± 0.14	82.08 ± 1.02	66.30 ± 0.09	79.39 ± 0.37
GAN-RPN [38]	Data augmentation	74.02 ± 0.06	83.66 ± 0.49	70.25 ± 0.56	78.03 ± 0.95
Improved GAN [39]	Data augmentation	75.81 ± 0.35	86.27 ± 0.30	71.64 ± 0.69	80.32 ± 0.61
DAGAN [40]	Data augmentation	71.08 ± 0.06	82.64 ± 0.97	68.22 ± 0.71	78.10 ± 0.56
ACGAN [41]	Data augmentation	72.07 ± 1.54	84.96 ± 0.22	69.03 ± 0.66	80.63 ± 0.28
Semi GAN [43]	Data augmentation	73.82 ± 0.08	85.34 ± 0.65	70.20 ± 0.39	82.37 ± 0.23
f-VAEGAN-D2 [46]	Data augmentation	75.23 ± 0.07	88.12 ± 0.33	71.05 ± 0.75	83.06 ± 0.66
Proposed	Data augmentation	80.64 ± 0.89	90.38 ± 0.29	78.35 ± 1.09	87.26 ± 0.91

The bold mark is the detection accuracy obtained by our proposed method

Table 3 Comparison of accuracy of each method on CICIDS2017FS dataset

Method	Method type	10-shot accuracy (%)		5-shot accuracy (%)	
		Normal	Attack	Normal	Attack
Meta LSTM [29]	Meta	63.64 ± 0.75	76.33 ± 0.26	–	–
MAML [31]	Meta	65.49 ± 0.38	74.64 ± 0.12	–	–
Matching Nets [23]	Metric	70.84 ± 0.59	83.40 ± 0.19	67.47 ± 0.39	82.26 ± 0.55
Proto Nets [22]	Metric	71.46 ± 0.35	85.28 ± 0.37	70.04 ± 0.84	84.93 ± 1.38
CapsNet-GAN [18]	Data augmentation	76.57 ± 0.59	87.64 ± 0.62	73.51 ± 0.26	85.89 ± 0.62
GAN-RPN [38]	Data augmentation	75.10 ± 1.62	86.09 ± 0.25	70.64 ± 0.55	84.17 ± 0.29
Improved GAN [39]	Data augmentation	78.17 ± 0.09	88.93 ± 0.41	75.72 ± 1.33	87.21 ± 0.90
DAGAN [40]	Data augmentation	76.96 ± 0.05	89.37 ± 0.12	74.60 ± 0.88	86.15 ± 0.68
ACGAN [41]	Data augmentation	80.68 ± 0.25	90.08 ± 0.63	78.95 ± 0.27	89.22 ± 0.75
Semi GAN [43]	Data augmentation	81.67 ± 1.24	91.04 ± 0.77	80.08 ± 0.36	89.33 ± 0.09
f-VAEGAN-D2 [46]	Data augmentation	78.64 ± 0.32	88.58 ± 0.30	74.98 ± 0.25	86.16 ± 0.72
Proposed	Data augmentation	84.05 ± 0.66	94.49 ± 0.22	81.62 ± 0.37	92.04 ± 0.63

The bold mark is the detection accuracy obtained by our proposed method

Table 4 Comparison of accuracy of each method on cross dataset

Method	Method type	10-shot accuracy (%)		5-shot accuracy (%)	
		Normal	Attack	Normal	Attack
Meta LSTM [29]	Meta	49.68 ± 0.75	63.28 ± 0.34	–	–
MAML [31]	Meta	53.20 ± 0.42	67.07 ± 0.66	–	–
Matching Nets [23]	Metric	57.26 ± 1.04	70.07 ± 0.66	52.58 ± 1.27	65.08 ± 0.36
Proto Nets [22]	Metric	59.64 ± 0.51	72.33 ± 0.89	55.06 ± 0.38	67.24 ± 0.81
CapsNet-GAN [18]	Data augmentation	63.70 ± 0.32	74.05 ± 1.27	60.83 ± 0.26	70.58 ± 0.62
GAN-RPN [38]	Data augmentation	67.21 ± 1.04	75.53 ± 0.56	63.09 ± 0.24	71.29 ± 1.35
Improved GAN [39]	Data augmentation	71.26 ± 0.88	82.36 ± 0.75	67.53 ± 0.86	80.26 ± 1.29
DAGAN [40]	Data augmentation	69.45 ± 0.63	80.04 ± 1.41	64.82 ± 0.37	78.22 ± 0.54
ACGAN [41]	Data augmentation	70.34 ± 0.96	82.31 ± 0.82	68.44 ± 0.66	79.97 ± 0.23
Semi GAN [43]	Data augmentation	72.20 ± 0.05	83.66 ± 0.09	69.31 ± 0.27	80.94 ± 0.06
f-VAEGAN-D2 [46]	Data augmentation	70.85 ± 0.44	81.17 ± 0.28	67.40 ± 0.93	79.03 ± 0.25
Proposed	Data augmentation	75.06 ± 0.39	85.90 ± 0.34	70.26 ± 0.61	82.38 ± 0.44

The bold mark is the detection accuracy obtained by our proposed method

The following conclusions can also be drawn in Fig. 10a–c: the accuracy under 10-shot is slightly higher than that of 5-shot training tasks. Because the traffic features under 10-shot are more than 5-shot. Other methods have poor convergence of accuracy curves under 5-shot except for ARCN. The ARCN accuracy curve has the best convergence, and the average accuracy rate is 98.49%. When epochs are increased, the accuracy rate remains stable.

5.3 Few-Shot Sample Classification Experiments

5.3.1 Two-Classification experiments

The few-shot sample classification comparison experiments are constructed in the section, which are given in Tables 2, 3 and 4 under different classification methods. The experiment

is performed under 5-shot and 10-shot settings. Normal traffic is represented by normal, and abnormal traffic is represented by an attack. The Adam optimization method is used to train the network, and the initial learning rate is set to 1×10^{-3} . Moreover, the learning rate is decayed with a decay factor of 0.5 every 10,000 iterations. The classification of the single dataset is compared in Tables 2 and 3. As can be seen from Table 2, we found that the accuracy of the meta-learning is poor in the ICSX2012FS. The accuracy rates are 64.485 and 68.645% under the 10-shot setting, which is lower than other methods. The average accuracy rate of the proposed method reaches 85.51% under the 10-shot setting. They are 21.025 and 16.865% higher than Meta LSTM and MAML, respectively. Because they are meta-learning methods, a large number of categories are required to train. However, the dataset that is constructed has a few-shot

category in the paper. Therefore, meta-learning often has the problem of underfitting. Moreover, the proposed method is 14.15 and 18.765% higher than Matching Nets and Proto Nets, respectively. Furthermore, the highest accuracy rate is also achieved in the data augmentation method. In comparison, the accuracy rate of each method under the 5-shot setting is lower than the 10-shot setting. But the method with the highest accuracy is still proposed. The average accuracy rate is 86.83%. Although it is 2.705% lower than 5-shot, it is higher than other methods.

Similarly, the same conclusions can be obtained from Table 3. For instance, the accuracy rate of the two meta-learning methods under the 10-shot setting is 69.985% and 70.065%, which are lower than other methods. The average accuracy rate of the proposed method is 89.27% under the

10-shot setting. They are 19.285 and 19.205% higher than Meta LSTM and MAML, respectively. They are 12.15 and 10.9% higher than Matching Nets and Proto Nets, respectively. The reasons for the low accuracy are consistent with the ICSX2012FS experiment. The average accuracy of the proposed method is 86.83% under the 5-shot setting. Although it is 2.44% lower than 10-shot, it is higher than other methods. The traffic features are extracted through a multi-scale Deep-CapsNet. Moreover, the ARCNet is used to reconstruct and classify the data. The noise interference is removed during data generation, so the final classification effect is better. Therefore, it can be observed from the classification experiments of the single dataset: regardless of 5-shot or 10-shot, the proposed method has the highest accuracy in the few-shot sample.

Table 5 Multi-classification accuracy in the ICSX2012FS dataset

Method	Method type	Accuracy (%)			
		iA	iB	iC	iD
Meta LSTM [29]	Meta	45.64 ± 0.47	46.48 ± 0.55	–	–
MAML [31]	Meta	47.07 ± 0.94	49.89 ± 0.74	–	–
Matching Nets [23]	Metric	49.23 ± 0.52	48.05 ± 0.62	50.95 ± 0.49	48.63 ± 0.41
Proto Nets [22]	Metric	53.91 ± 0.66	56.71 ± 1.06	48.67 ± 0.56	52.39 ± 0.58
CapsNet-GAN [18]	Data augmentation	57.86 ± 0.83	54.32 ± 0.35	56.13 ± 0.44	57.18 ± 0.34
GAN-RPN [38]	Data augmentation	55.20 ± 0.09	56.10 ± 0.67	57.68 ± 0.47	60.07 ± 0.62
Improved GAN [39]	Data augmentation	58.82 ± 0.41	59.38 ± 0.51	59.42 ± 0.34	59.45 ± 0.08
DAGAN [40]	Data augmentation	62.76 ± 0.49	63.17 ± 0.30	64.89 ± 1.28	63.52 ± 0.84
ACGAN [41]	Data augmentation	65.74 ± 1.28	62.85 ± 0.12	61.80 ± 0.91	65.87 ± 0.05
Semi GAN [43]	Data augmentation	63.55 ± 0.32	67.09 ± 0.39	65.73 ± 0.86	68.19 ± 0.76
f-VAEGAN-D2 [46]	Data augmentation	70.47 ± 0.25	69.08 ± 0.97	69.97 ± 0.08	71.28 ± 0.34
Proposed	Data augmentation	76.97 ± 0.46	78.44 ± 0.63	75.29 ± 0.15	79.64 ± 0.49

The bold mark is the detection accuracy obtained by our proposed method

Table 6 Multi-classification accuracy in the CICIDS2017FS dataset

Method	Method type	Accuracy (%)				
		cA	cB	cC	cD	cE
Meta LSTM [29]	Meta	50.71 ± 0.49	48.52 ± 0.61	–	–	–
MAML [31]	Meta	49.85 ± 0.66	49.72 ± 0.43	–	–	–
Matching Nets [23]	Metric	51.27 ± 0.44	51.09 ± 0.78	51.39 ± 0.71	47.14 ± 0.97	51.24 ± 0.61
Proto Nets [22]	Metric	48.66 ± 0.32	53.76 ± 0.55	49.67 ± 0.33	49.61 ± 0.36	52.19 ± 0.14
CapsNet-GAN [18]	Data augmentation	55.75 ± 0.96	56.94 ± 0.59	53.19 ± 0.49	52.68 ± 0.65	56.92 ± 0.63
GAN-RPN [38]	Data augmentation	57.49 ± 0.61	60.18 ± 0.76	55.28 ± 0.36	54.10 ± 0.28	55.18 ± 0.42
Improved GAN [39]	Data augmentation	59.73 ± 0.81	61.27 ± 0.42	60.29 ± 0.07	57.19 ± 0.26	57.25 ± 0.34
DAGAN [40]	Data augmentation	62.15 ± 0.94	59.48 ± 0.37	58.16 ± 0.92	61.82 ± 0.67	60.17 ± 0.22
ACGAN [41]	Data augmentation	65.10 ± 0.31	63.19 ± 0.45	65.08 ± 0.06	64.09 ± 0.94	63.08 ± 0.19
Semi GAN [43]	Data augmentation	63.28 ± 0.39	65.93 ± 0.26	68.49 ± 0.38	69.15 ± 0.45	68.29 ± 0.55
f-VAEGAN-D2 [46]	Data augmentation	69.85 ± 0.67	70.29 ± 0.46	71.18 ± 0.07	70.28 ± 0.67	72.17 ± 0.05
Proposed	Data augmentation	75.19 ± 0.26	78.41 ± 0.35	79.42 ± 1.36	78.24 ± 0.03	79.16 ± 0.09

The bold mark is the detection accuracy obtained by our proposed method

Table 7 Comparison of classification accuracy

EM vector clus- tering	Multi- scale capsule	RCN	Adversar- ial training	Accuracy/(%)	
				Normal	Attack
				63.44 ± 0.38	70.56 ± 0.41
✓				70.12 ± 0.32	74.92 ± 0.48
✓	✓			74.66 ± 0.19	79.34 ± 0.20
✓	✓	✓		79.67 ± 0.39	83.44 ± 0.22
✓	✓	✓	✓	85.58 ± 0.17	85.09 ± 0.33
EM clus- ter- ing	Multi- scale capsule	RCN	Adver- sarial training	Normal	Attack
	✓	✓	✓	80.26 ± 0.48	79.65 ± 0.41
✓		✓	✓	79.33 ± 0.72	80.12 ± 0.63
✓	✓		✓	79.27 ± 0.24	78.05 ± 0.38
✓	✓	✓		78.69 ± 0.55	78.12 ± 0.92

The bold mark is the detection accuracy obtained by our proposed method

The classification results on the cross dataset are compared in Table 4, and the conclusions are as follows: first, the accuracy of each method on the cross dataset is lower than the single dataset. The cross dataset is set that crosses the database, which is much lower than that of a single dataset. Moreover, the features of some attack traffic in the CICIDS2017FS are similar to the ICSX2012FS. Therefore, misjudgments are often caused. Compared with the data augmentation methods, the four methods based on metric learning and meta-learning have lower accuracy. The reason for the poor performance is the same as described above. Third, the proposed method has the highest accuracy under 10-shot and 5-shot settings. The average accuracy rates are 80.48 and 76.32%, respectively. Therefore, it is known from the experimental results that it has better performance in the cross datasets.

From the two classification results on three datasets, the effectiveness of the proposed method has been proven again. Further explanation: first, it is effective for improving the capsule and GAN. After the multi-scale Deep-CapsNet is used to extract traffic features, the discrimination of traffic features can be enhanced. Second, the complex high-level features can be further recognized by the attitude transformation. Third, high accuracy can still be obtained in a few-shot sample by reconstruction classification and adversarial training.

5.3.2 Multi-classification Experiments

To further verify the performance in a few-shot sample classification, all the attacks dataset are combined as a training

Fig. 11 Visualization comparison of ICSX2012FS

set on the ICSX2012FS and CICIDS2017FS. The attack categories of the ICSX2012FS dataset and the CICIDS2017FS dataset are used as the test sets. The multi-classification of attack traffic is shown in Tables 5 and 6 under each method. The main conclusions drawn from Tables 5 and 6 are as follows: first, the training and testing of cross-database datasets are feasible. Although the two datasets are obtained by different software and hardware, the traffic has certain commonalities. Therefore, the experiment is feasible in this paper. Second, it can be observed from the accuracy results are obviously inferior to the two-classification. Because the similarity of the two datasets is easy to be misjudged in the

Fig.12 Visualization comparison of CICIDS2017FS

classification. Especially the classification of each attack traffic is more likely to cause misjudgment. Moreover, there is also cross-validation set in the distribution of the training set and the test set. When multi-classification is performed, the results are poor due to an insufficient cross-validation set. Third, the proposed method has the best performance compared to other methods. The average accuracy rates are 77.585% and 78.084% on the ICSX2012FS and CICIDS2017FS. From the multi-classification results on the datasets, the effectiveness of the proposed method is verified again. Moreover, the validity and universality are also verified again in a few-shot sample scene.

Fig.13 Visualization comparison of the cross dataset

5.4 Ablation Experiments

To further verify the performance of the optimization strategy, we have constructed an overall ablation experiment. In the experiment, the accuracy without any optimization strategy is used as the baseline. The accuracy of the cross dataset is given when each optimization strategy is gradually added and removed (Table 7). The upper section of Table 7 is the comparison that the optimization strategy is gradually added. The lower section is the comparison that the optimization strategy is removed each time.

Table 8 GAM comparison results between DRCN and ARCN

Dataset	DRCN	ARCN	r_{test}	r_{sample}
ICSX2012FS	$T=1$	$T=1$	0.964	1.008
	$T=3$	$T=3$	0.992	1.019
	$T=5$	$T=5$	0.983	2.674
CICIDS2017FS	$T=1$	$T=1$	1.00	1.006
	$T=3$	$T=3$	1.00	1.051
	$T=5$	$T=5$	1.07	3.875
Cross dataset	$T=1$	$T=1$	1.086	1.331
	$T=3$	$T=3$	1.00	2.149
	$T=5$	$T=5$	1.039	3.366

The conclusions are obtained in the upper section of Table 7: first, the average accuracy rate is 67% under the baseline experiment. The average accuracy rate reaches 72.52% when the improved EM vector clustering is added. It is 5.52% higher than the baseline experiment. The improved EM vector clustering for capsule optimization is first proved. Second, when the multi-scale convolutional capsule, RCN, and adversarial training are added, the accuracy is further improved based on the previous step. The average accuracy rate is 85.335% when all four strategies are added. It is 18.335% higher than the baseline experiment. Fourth, the performance is poor only RCN is used. Because the RCN is easily disturbed by noise during reconstruction. After the adversarial training is added, the accuracy will be greatly

improved. The importance of adversarial training for RCN has been proven once again.

Similarly, the lower section of Table 7 is the comparison that the optimization strategy is removed. Therefore, all strategies is added as baseline experiments. We found that the accuracy is reduced when one of the optimization strategy is removed. However, the most influential is adversarial training. The average accuracy rate is 78.405 when it is removed. It is 6.93% lower than all four strategies are added. The importance of adversarial training for the noisy few-shot samples is fully explained. In summary, the necessity of each strategy is proved from the overall ablation experiment.

To more intuitively verify the performance of each optimization strategy, the visualization experiment is constructed. The visualization results of the three datasets are given when each optimization strategy is gradually added. The visualization of larger traffic samples is also given as a comparison, which is shown in Figs. 11, 12 and 13.

From the large sample visualization results of the ICSX2012FS and CICIDS2017FS, it can be observed the classification effect is getting better as the optimization strategies are added. When the adversarial training is added, attack and normal sample achieve effective classification. The data sample is more convergent and compact. From the few-shot sample visualization results shown in Figs. 11 and 12 (type 1 samples are attack traffic, and type 2 samples are normal traffic), it can be observed attack and normal can achieve effective classification when each optimization strategy is added. However, the data distribution is relatively

Table 9 Comparison of IS rate between DRCN and ARCN

Dataset	DRCN			ARCN		
	$T=1$	$T=3$	$T=5$	$T=1$	$T=3$	$T=5$
ICSX2012FS	1.257 ± 0.09	1.472 ± 0.05	1.473 ± 0.04	1.533 ± 0.11	1.798 ± 0.04	1.806 ± 0.07
CICIDS2017FS	2.334 ± 0.06	2.463 ± 0.05	2.471 ± 0.03	2.857 ± 0.12	2.964 ± 0.02	2.981 ± 0.03
Cross dataset	7.651 ± 0.06	7.961 ± 0.11	8.012 ± 0.12	7.946 ± 0.02	8.369 ± 0.06	8.413 ± 0.04

Table 10 Comparison of IS rate with other GAN models

Method	ICSX2012FS			CICIDS2017FS		
	$T=1$	$T=3$	$T=5$	$T=1$	$T=3$	$T=5$
GAN-RPN	1.031 ± 0.08	1.107 ± 0.18	1.112 ± 0.04	1.334 ± 0.06	1.483 ± 0.17	1.509 ± 0.03
Improved GAN	1.145 ± 0.03	1.236 ± 0.07	1.252 ± 0.06	1.418 ± 0.02	1.456 ± 0.06	1.498 ± 0.08
DAGAN	1.337 ± 0.09	1.359 ± 0.04	1.374 ± 0.05	1.525 ± 0.07	1.573 ± 0.06	1.580 ± 0.12
ACGAN	1.296 ± 0.14	1.312 ± 0.15	1.329 ± 0.11	1.497 ± 0.12	1.525 ± 0.15	1.538 ± 0.27
Semi GAN	1.413 ± 0.12	1.425 ± 0.23	1.433 ± 0.16	1.754 ± 0.09	1.779 ± 0.24	1.791 ± 0.18
f-VAEGAN-D2	1.429 ± 0.12	1.461 ± 0.26	1.472 ± 0.34	1.816 ± 0.12	1.833 ± 0.15	1.842 ± 0.16
DRCN	1.257 ± 0.09	1.472 ± 0.05	1.473 ± 0.04	2.334 ± 0.06	2.463 ± 0.05	2.471 ± 0.03
ARCN	1.533 ± 0.11	1.798 ± 0.04	1.806 ± 0.07	2.857 ± 0.12	2.964 ± 0.02	2.981 ± 0.03

The bold mark is the detection accuracy obtained by our proposed method

Fig. 14 Visualization comparison when noise is added

scattered and messy. It also shows that the effect of feature extraction and training is poor in the few-shot sample. The sample can not only achieve effective classification when four optimization strategies are added, but the sample is more compact. Therefore, the necessity of an optimization strategy has also been proved.

From the larger sample visualization of the cross dataset in Fig. 13a–d, the following conclusions can be drawn: the classification effect is getting better as the optimization strategies are added. When adversarial training is added, most samples of attack and normal achieve effective classification. However, individual samples are still not effectively classified. Because the CICIDS2017FS dataset contains 5 types of attacks, they are disguised as the normal samples in the individual attack. When an attack of ICSX2012FS

Fig. 15 Noisy traffic signal and processing effect

and normal traffic is used for testing, misjudgments will be caused. But when the four optimization strategies are added, the visualization results are best. The sample is more convergent and compact. Furthermore, the same conclusion is obtained from the few-shot sample visualization. Therefore, the effectiveness of the strategy has been proved in the visualization of cross datasets. From the visualization results of the three datasets in Figs. 11, 12 and 13, it can be observed the classification effect is getting better as the optimization strategies are added. Particularly, adversarial training is most important for traffic classification.

5.5 Generated Data Quality Evaluation

To verify the generated data quality, generated data experiment is constructed. Batch training is used in the experiment, which is composed of 128 sampling data. Two evaluation indexes: generative adversarial metric (GAM) and Inception score (IS). Moreover, GAM includes the training phase and testing phase of ARCN.

$M_1 = \{(G_1, D_1)\}$ and $M_2 = \{(G_2, D_2)\}$ are two different generative adversarial networks. In the training phase, M_1 and M_2 are trained separately. In the test phase, the opponent's discriminator is used to discriminate the generated samples by its generator. The same random noise z is used as the input of the generators in M_1 and M_2 . $G_1(z)$ and $G_2(z)$ are used to represent the generated samples by M_1 and M_2 . The real numbers are input to the discriminator during the training and testing. They are denoted by x_{train} and x_{test} . In GAM comparison, there are two ratios: r_{test} and r_{sample} . $\text{err}(\cdot)$ represents the classification error rate, which is defined as follows:

Table 11 Accuracy under the three types of datasets when noise is added

Method	Accuracy/(%)			t/s	
	CICIDS2017FS	ICSX2012FS	Cross dataset	Training	Test
CapsNet-GAN [18]	52.33 ± 0.68	55.92 ± 1.26	41.25 ± 0.37	43.146	2.359
GAN-RPN [38]	55.64 ± 0.52	56.08 ± 0.07	44.89 ± 0.28	58.218	6.845
Improved GAN [39]	59.42 ± 0.49	61.26 ± 0.35	47.50 ± 1.34	50.334	4.296
DAGAN [40]	53.27 ± 0.73	57.28 ± 0.62	45.68 ± 1.29	56.173	5.334
ACGAN [41]	54.08 ± 0.55	59.52 ± 0.61	46.71 ± 0.06	49.144	3.087
Semi GAN [43]	61.36 ± 0.21	66.75 ± 0.33	52.43 ± 0.31	50.389	4.223
f-VAEGAN-D2 [46]	59.98 ± 0.44	63.96 ± 1.04	50.08 ± 0.82	53.248	4.426
Proposed	67.34 ± 0.06	75.87 ± 0.59	69.24 ± 0.83	43.848	2.108

The bold mark is the detection accuracy obtained by our proposed method

$$r_{\text{test}} = \frac{\text{err}(D_1(x_{\text{test}}))}{\text{err}(D_2(x_{\text{test}}))}, \quad (18)$$

$$r_{\text{sample}} = \frac{\text{err}(D_1(G_2(z)))}{\text{err}(D_2(G_1(z)))}. \quad (19)$$

In the formula: the generalization ability of M_1 and M_2 can be reflected by r_{test} . To reduce the bias of the discriminator, r_{test} is used to ensure the fairness of comparison. r_{sample} is used to determine the winning model, and the specific discriminant rules can be expressed as:

$$\text{win} = \begin{cases} M_1, r_{\text{sample}} < 1 \text{ and } r_{\text{test}} \approx 1 \\ M_2, r_{\text{sample}} > 1 \text{ and } r_{\text{test}} \approx 1 \\ \text{Tie, others} \end{cases} \quad (20)$$

First, the GAM comparison experiment is constructed in the ARCN and DRCN. The time step of the generator is set to $T = \{1, 3, 5\}$. Ten test samples are selected as real samples, and the generated samples are 10 samples from the generator. Moreover, the former model is regarded as $M_1 = \{(G_1, D_1)\}$, and the latter model is regarded as $M_2 = \{(G_2, D_2)\}$. The comparison results are shown in Table 8.

Moreover, the main conclusions are as follows: first, $M_2 = \{(G_2, D_2)\}$ is the winner at the same time step. It shows that the quality of the generated data by the ARCN is better than that of DRCN. Second, the quality of the generated data keeps better as time steps are increased. However, it can be observed that quality is the best when $T=3$. Third, the best GAM of the three datasets is the CICIDS2017FS. Because this dataset has more attack categories, the discriminator in another model is more likely to be "spoofed" by the generated samples. To further verify the performance of ARCN, the Inception score (IS) rate experiment is continued to be constructed. The higher the IS, the higher the diversity

of the generated samples. On the contrary, the diversity is lower.

Table 9 is IS rate that is obtained by DRCN and ARCN under different time steps. The IS rate of the two models will be increased as the time step is increased. It shows that the proposed structure is beneficial to increase diversity. Furthermore, we can observe that the IS rate of the cross dataset is much higher. Because the cross dataset contains more categories than other datasets. The span between categories is larger, and the different samples are easier to obtain. Moreover, it can be observed that the IS rate of ARCN is higher than DRCN.

Furthermore, the proposed method is compared with other GAN. The comparison models are GAN-RPN [27], Improved GAN [28], DAGAN [29], ACGAN [21], Semi GAN [31], f-VAEGAN-D2 [34], DRCN, and ARCN network. To highlight the diversity, only ICSX2012FS and CICIDS2017FS are used for experiments. The IS rate is still compared, which is shown in Table 10.

It can be observed from Table 10 that the IS rate also increases as the time step is increased. However, the ARCN has the highest IS rate. It has obvious advantages, which are conducive to improving diversity. The very few labeled samples can be used by ARCN for training. High-order inconsistencies between the reconstructed and the original data are detected. Therefore, the data quality is improved.

To more intuitively verify the data quality, noise is added to the cross dataset for visualization analysis. The visualizations of the seven data augmentation and the proposed method are given, which are shown in Fig. 14.

Type 1 is normal. Type 2 is attacked, and type 3 is noise. From Fig. 14a–g, the data quality generated by other methods is mediocre when noise is added. Moreover, the samples have not been effectively separated. On the contrary, it is known that the three datasets can be effectively distinguished by the proposed method when noise traffic is added. In addition, the samples are more convergent. Therefore, the

effectiveness of the proposed method is verified from the visualization experiments. Furthermore, the importance of adversarial training has been verified again.

5.6 Anti-noise and Time Analysis

To verify the anti-noise performance of the method more effectively, the signal of the cross dataset is input into the Netflow Analyzer in a certain period of time. It is used to form a traffic signal. Moreover, noise is added to this section of the signal. It is used to form a noisy signal (the upper section of Fig. 15). This signal is processed by the proposed method, and the processed signal is shown in the lower section of Fig. 15. The following conclusions can be drawn from the Fig. 15: the noise in the signal can be removed by adversarial training. Moreover, there is no distortion and smoothness in the processed signal.

Furthermore, the three datasets are added to the malicious noise traffic. The classification accuracy and the time of each method are compared. Moreover, the final results are used to verify the robustness. When there is noise in the three datasets, the average accuracy of the seven data augmentation methods, the running time, and the testing time are given in Table 11.

The conclusions that can be drawn from Table 11 are as follows: first, the accuracy will be decreased when malicious traffic noise is added to the three datasets. The most affected of the three datasets is the cross dataset. When malicious noise traffic is added, its performance is poor as the training categories are increased. Second, the proposed method can obtain the best performance when noise traffic is added. The accuracy rate is the highest under the three datasets. Third, although the training time of the CapsNet-GAN is 0.702 s shorter than that of the proposed method, the test time of the proposed method is the shortest. Because the EM vector clustering is designed to optimize dynamic iterative routing, the final efficiency is improved. It shows that the proposed model can be used for real-time few-shot sample traffic classification, and it has good robustness.

6 Conclusions

Because of the scenario where the few-shot attack samples can only be obtained, the capsule and the reconstruction classification are combined to propose a few-shot abnormal traffic detection method. Two internationally public datasets ICSX2012 and CICIDS2017 are used to complete the training and testing of a few-shot sample. The main conclusions are as follows:

1. The proposed method has better training performance in a few-shot sample of three datasets. Few-shot sample

training is more superior when multi-scale Deep-CapsNet and ARCN are combined. It has a higher training accuracy and a lower loss rate.

2. It has the highest classification accuracy under the 5-shot and 10-shot sample settings compared to other methods. It is known from the ablation experiment that it is effective for improving the capsule network and GAN. After the multi-scale Deep-CapsNet is used to extract traffic features, the discrimination of features can be enhanced. Moreover, the ARCN can still obtain high classification accuracy in the few-shot sample.
3. The visualization effect has the best classification effect in both large and few-shot samples. Normal and attack traffic can be effectively classified. Especially when noise is added to the dataset, the proposed method has good anti-noise performance and short running time. It shows that the proposed method can be used for real-time few-shot sample traffic detection, and it has good robustness. In the next research, the abnormal traffic will be used for detection in the real network environment.

Acknowledgements The authors express great thanks to the financial support from the National Natural Science Foundation of China. This research is funded by the National Natural Science Foundation of China, Grant number 61703349; and Supported by Sichuan Science and Technology Program, Grant number 2021YJ0070. Moreover, it is funded by the Science and Technology Research and Development Project, Grant number N2018G062, K2018G011.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Hamed, T., Dara, R., Kremer, S.C.: Network intrusion detection system based on recursive feature addition and bigram technique. *Comput. Secur.* **73**, 137–155 (2018)
2. Wang, W., Sheng, Y., Wang, J., et al.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **6**, 1792–1806 (2018)

3. Min, E., Long, J., Liu, Q., et al.: TR-IDS: anomaly-based intrusion detection through text-convolutional neural networks and random forest. *Secur. Commun. Netw.* (2018)
4. Vinayakumar, R., Soman, K., Poornachandran, P.: Applying convolutional neural network for network intrusion detection. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1222–1228 (2017)
5. Sun, C., Shrivastava, A., Singh, S.: Revisiting unreasonable effectiveness of data in deep learning era. In: IEEE Computer Society, pp. 843–852 (2017)
6. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations. *J. Mach. Learn. Res.* **20**(184), 1–25 (2019)
7. Islam, M.A., Jia, S., Bruce, N.: How much position information do convolutional neural networks encode. In: Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, pp. 1–10 (2020)
8. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules, in: Advances in neural information processing systems, pp. 3856–3866 (2017)
9. Shone, N., Ngoc, T.N., Phai, V.D., et al.: A deep learning approach to network intrusion detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2**(1), 41–50 (2018)
10. Al-Qatf, M., Lasheng, Y., Al-Habib, M., et al.: Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access* **6**, 52843–52856 (2018)
11. Fiore, U., Palmieri, F., Castiglione, A., et al.: Network anomaly detection with the restricted Boltzmann machine. *Neurocomputing* **122**, 13–23 (2013)
12. Ghifary, M., Kleijn, W.B., Zhang, M., et al.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: European Conference on Computer Vision, pp. 597–613 (2016)
13. Wang, W., Zhu, M., Zeng, X., et al.: Malware traffic classification using convolutional neural network for representation learning. In: Information Networking (ICOIN), 2017 International Conference on, pp. 712–717 (2017)
14. Neill, J.O.: Siamese capsule networks. arXiv preprint <http://arxiv.org/abs/1805.07242> (2018)
15. Duarte, K., Rawat, Y., Shah, M.: Videocapsulenet: a simplified network for action detection. In: Advances in neural information processing systems, pp. 7610–7619 (2018)
16. LaLonde, R., Bagci, U.: Capsules for object segmentation. ArXiv Preprint (2018)
17. Rajasegaran, J., Jayasundara, V.: Deep-Caps: going deeper with capsule networks. <http://arxiv.org/abs/1904.09546> [v1] (2019)
18. Deng, F., Pu, S., Chen, X.: Hyperspectral image classification with capsule network using limited training samples. *Sensors* **18**(9), 3153 (2018)
19. Zhao, W., Ye, J., Yang, M.: Investigating capsule networks with dynamic routing for text classification. arXiv preprint <http://arxiv.org/abs/1804.00538> (2018)
20. Vinyals, O., Blundell, C., Lillicrap, T., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, pp. 3630–3638 (2016)
21. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems, pp. 4077–4087 (2017)
22. Jiang, L.B., Zhou, X.L., Jiang, F.W., Che, L.: One-shot learning based on improved matching network. *Syst. Eng. Electron.* **41**(6), 1210–1217 (2019)
23. Xiang, J., Havaei, M., Chartrand, G., et al.: On the importance of attention in meta-learning for few-shot text classification. arXiv preprint <http://arxiv.org/abs/1806.00852> (2018)
24. Munkhdalai, T., Yu, H.: Meta networks. In: International Conference on Machine Learning, Proc. of the PMLR, pp. 2554–2563 (2017)
25. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: Proc. of the ICLR (2016)
26. Wang, Y.X., Hebert, M.: Learning from small sample sets by combining unsupervised meta-training with CNNs. In: Advances in Neural Information Processing Systems, pp. 244–252 (2016)
27. Boney, R., Ilin, A.: Semi-supervised few-shot learning with MAML. In: Proc. of the ICLR (Workshop) (2018)
28. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proc. of the 34th Int'l Conf. on Machine Learning, vol. 70, pp. 1126–1135 (2017)
29. Ren, M.Y., Triantafyllou, E., Ravi, S., et al.: Meta-learning for semi-supervised few-shot classification. arXiv preprint <http://arxiv.org/abs/1803.00676> (2018)
30. Liu, Y., Lee, J., Park, M., et al.: Learning to propagate labels: transductive propagation network for few-shot learning. arXiv preprint <http://arxiv.org/abs/1805.10002> (2018)
31. Hou, R.B., Chang, H., Ma, B.P., et al.: Cross attention network for few-shot classification. In: Advances in Neural Information Processing Systems, pp. 4003–4014 (2019)
32. Mehrotra, A., Dukkipati, A.: Generative adversarial residual pairwise networks for one shot learning. arXiv preprint <http://arxiv.org/abs/1703.08033> (2017)
33. Ali-Gombe, A., Elyan, E., Savoye, Y., Jayne, C.: Few-shot classifier gan. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8 (2018)
34. Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks. arXiv (2017)
35. Lin, Z., Chen, Y., Ghamisi, P., et al.: Generative adversarial networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **56**(9), 5046–5063 (2018)
36. Zhang, M., Gong, M., Mao, Y., et al.: Unsupervised feature extraction in hyperspectral images based on Wasserstein generative adversarial network. *IEEE Trans. Geosci. Remote Sens.* **57**(5), 2669–2688 (2018)
37. Zhong, Z., Li, J.: Generative adversarial networks and probabilistic graph models for hyperspectral image classification. In: Thirty-second AAAI conference on artificial intelligence. New Orleans, USA, pp. 8191–8192 (2018)
38. Hariharan, B., Girshick, R.: Low-shot visual recognition by shrinking and hallucinating features. In: Proc. of the IEEE Int'l Conf. on Computer Vision, pp. 3018–3027 (2017)
39. Wang, Y.X., Girshick, R., Hebert, M., et al.: Low-shot learning from imaginary data. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 7278–7286 (2018)
40. Xian, Y., Sharma, S., Schiele, B., et al.: f-VAEGAN-D2: a feature generating framework for any-shot learning. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 10275–10284 (2019)
41. Chen, Z., Fu, Y., Kim, Y.X., et al.: Image deformation meta-networks for one-shot learning. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 8680–8689 (2019)
42. Dixit, M., Kwitt, R., Niethammer, M., et al.: AGA: attribute guided augmentation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 7455–7463 (2017)
43. Liu, B., Wang, X., Dixit, M., et al.: Feature space transfer for data augmentation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 9090–9098 (2018)
44. Schwartz, E., Karlinsky, L., Shtok, J., et al.: Delta-encoder: an effective sample synthesis method for few-shot object recognition. In: Advances in Neural Information Processing Systems, pp. 2845–2855 (2018)

45. Chen, Z., Fu, Y., Zhang, Y., et al.: Semantic feature augmentation in few-shot learning. arXiv preprint <http://arxiv.org/abs/1804.05298> (2018)
46. Shen, W., Shi, Z., Sun, J.: Learning from adversarial features for few-shot classification. arXiv preprint <http://arxiv.org/abs/1903.10225> (2019)
47. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Proceeding of the Neural Information Processing Systems, pp. 3856–3866 (2017)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.