Computers, Materials & Continua CMC, vol.64, no.2, pp.941-959, 2020

A Network Traffic Classification Model Based on Metric Learning

Mo Chen[1], Xiaojuan Wang[1,*], Mingshu He[1], Lei Jin[1], Khalid Javeed[2] and Xiaojun Wang[3]

Abstract: Attacks on websites and network servers are among the most critical threats in network security. Network behavior identification is one of the most effective ways to identify malicious network intrusions. Analyzing abnormal network traffic patterns and traffic classification based on labeled network traffic data are among the most effective approaches for network behavior identification. Traditional methods for network traffic classification utilize algorithms such as Naive Bayes, Decision Tree and XGBoost. However, network traffic classification, which is required for network behavior identification, generally suffers from the problem of low accuracy even with the recently proposed deep learning models. To improve network traffic classification accuracy thus improving network intrusion detection rate, this paper proposes a new network traffic classification model, called ArcMargin, which incorporates metric learning into a convolutional neural network (CNN) to make the CNN model more discriminative. ArcMargin maps network traffic samples from the same category more closely while samples from different categories are mapped as far apart as possible. The metric learning regularization feature is called additive angular margin loss, and it is embedded in the object function of traditional CNN models. The proposed ArcMargin model is validated with three datasets and is compared with several other related algorithms. According to a set of classification indicators, the ArcMargin model is proofed to have better performances in both network traffic classification tasks and open-set tasks. Moreover, in open-set tasks, the ArcMargin model can cluster unknown data classes that do not exist in the previous training dataset.

Keywords: Metric learning, ArcMargin, network traffic classification, CNNs.

1 Introduction

Increased use of the Internet has led to the generation of much more log data related to network traffic [Lins, Damasceno, Silva et al. (2012)]. However, the security of computer systems is compromised when attackers are able to gather essential information from

[1] Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[2] Department of Computer Engineering, Bahria University, Islamabad, Pakistan.

[3] School of Electronic Engineering, Dublin City University, Dublin, Ireland.

[*] Corresponding Author: Xiaojuan Wang. Email: wj2718@163.com.

Received: 19 January 2020; Accepted: 11 April 2020.

website databases. The ability to browse network traffic data can provide a record of human behavior. Therefore, analyzing network traffic is a way to detect malicious intrusion behaviors [Lin, Zhang, Lou et al. (2016)]. In addition to network traffic, wireless device traffic data and Internet of Things (IoT) traffic data often need to be further analyzed and detected in order to increase network security situational awareness and anomaly detection [Zhang, Yi, Wang et al. (2018)]. Besides, some application system was also proposed, such as intrusion detection and anticipation system for IEEE 802.15.4 devices [Tariq (2019)]. We studied a network traffic classification model based on convolutional neural network (CNN) with a metric learning method [Yang and Jin (2006)] for the purposes of improving network traffic classification accuracy and the detection of abnormal network traffic patterns.

Some traditional machine learning methods have been used to solve the classification assignment of network traffic [Salakhutdinov (2015)]. Based on the Decision Tree (DT) learning method, Alhawi et al. [Alhawi, Baldwin and Dehghantanha (2018)] proposed a machine learning evaluation method for consistent detection of ransomware in network traffic on Windows computers. Focusing on intrusion detection, Di [Di (2018)] proposed a novel framework based on a Support Vector Machine (SVM) model called LA-SVM which is designed to remove redundant features automatically. Because of the inability to analyze encrypted network traffic, most previous traffic classification methods are unable to adapt to modern traffic environments, such as the method based on Deep Packet Inspection (DPI) [Kumar, Dharmapurikar, Fang et al. (2006)] and port identification-based methods. Gul proposed a traffic classification method that can analyze normal traffic and encrypted traffic based on DT and K-Nearest Neighbor (KNN) algorithms by analyzing network flow [Gul, Yoan, Aapo et al. (2018)].

Feature extraction is an important part in the process of training a model, but it requires sufficient business knowledge and experience. Methods based on deep learning (DL) can collect features automatically in the training process, thereby facilitating the process of acquiring typical features and important information. [Wang, Ye, Chen et al. (2018)].

Methods based on metric learning have been widely applied in computer vision classification domain [Cheng, Yang, Yao et al. (2018)]. Such methods map image datasets from the same category as close as possible and map datasets from different categories as far as possible. However, metric learning has not been applied to optimizing the network traffic classification model yet. In this paper, we seek to integrate ArcMargin into a CNN and build a more effective model for network traffic classification.

In addition, the characteristics of network traffic often lead to the situation that real test set contains an unknown data class that the classification model has difficulty identifying. We call this the open-set problem. However, in the current classification model, it is difficult to detect such an unknown class in the task. To solve this problem and combine it with the proposed model, we put forth a cluster method based on the deep feature output from the ArcMargin model. The results indicate that our open-set model improves the classification result.

The contributions of this paper can be summarized as follows:

1) Proposing a network traffic classification model with additive angular margin loss embedded on CNNs (ArcMargin), which classifies the traffic data in the cosine space.

2) Using ArcMargin to train a feature layer, from which we can extract the embedded vectors to solve the open-set problem. Specifically, we make the CNN model more discriminative, which means decreasing the within-class distance and increasing the between-class distance.
3) Performing a comparative study through experiments that proves the effectiveness of our general classification model and open-set classification model. The experiment results show that our model further improves the classification results in three traffic network datasets in accuracy, F1-score and recall rate.

The remainder of this paper is arranged as follows. Section 2 presents related work. In Section 3, we explain the framework and process of classification experiments. In Section 4, we introduce ArcMargin and then discuss the general classification model and the proposed open-set classification model. In Section 5, we analyze results from the traditional models and CNN model embedded with metric learning. Section 6 summarizes the paper and outlines directions for future research.

2 Related work

Network traffic records a large amount of access information belonging to Internet users, which plays an important role in the field of network security. Extracting features from these records is almost always the first step in classifying network traffic in anomaly detection [Wang, Zhu, Zeng et al. (2017)].

DPI-based, port-based, behavioral-based, and statistical based methods are commonly used traffic classification methods [Finsterbusch, Richter, Rocha et al. (2014)]. These are traditional machine learning methods, which require a large number of features to classify the traffic. Schultz et al. [Schultz, Eskin, Zadok et al. (2002)] were the first to embed the concept of data mining into malware detection, which employs three static features: strings, Portable Executable (PE) and byte sequences. Gul et al. [Gul, Yoan, Aapo et al. (2018)] proposed a method Using KNN [Peterson (2009)] and the DT [Safavian and Landgrebe (1991)].

The limitation of traditional machine learning methods lies in the fact that user performance depends on the feature engineering resulting from manual analysis and private information. In addition, to construct human-engineered features, traditional machine learning approaches require extensive computational resources and substantial storage and are also highly labor-intensive. Representation learning is a relatively recent method that can learn features from raw data automatically, thus reducing the time spent on constructing human-engineered features. There have been several studies on traffic classification based on representation learning. Gao et al. [Gao, Gao, Gao et al. (2014)] proposed a malware traffic detection method based on deep belief networks. Javaid et al. [Javaid, Niyaz, Sun et al. (2016)] proposed a malware traffic identification model based on a sparse autoencoder. Nejad et al. [Nejad and Shiri (2019)] proposed a new enhanced learning approach to automatic image classification based on salp swarm algorithm. Zhang et al. [Zhang, Qin, Yin et al. (2016)] resolved binary executables into opcodes sequences and then transformed them into images, which used the CNN model to determine whether a binary executable is secure or not. Zeng et al. [Zeng, Gu, Wei et al. (2019)] proposed a network traffic classification method using CNN, Long Short-Term

Memory (LSTM) and Stacked Auto-Encoder (SAE).

Many classification algorithms rely on the distance metric of data. In order to handle the features from different datasets, it is necessary to build various measurement functions, which consume a great deal of both time and energy. The major contribution of metric learning is to discover a proper similarity measurement between pairs of data such as images that will not change original data distance and will retain the necessary distance structure. Today, metric learning is always used in facial verification. Metric learning methods such as the Siamese network and Joint Bayesian are used along with deep facial representation [Chopra, Hadsell, LeCun et al. (2005)] applied the Siamese network in collecting features from two inputs with the same two sub-networks and regarded the distance as dissimilar between each other. Huang et al. [Huang, Lee and Learned-Miller (2012)] utilized Information Theoretic Metric Learning (ITML) on the features learned from convolutional deep belief networks.

3 Network traffic classification framework and learning model

The proposed model is a CNN with metric learning under ArcMargin. To verify the experimental results and to improve the classification accuracy achieved by metric learning, we built a data classification and validation framework as shown in Fig. 1. At the same time the open-set classification results were improved by embedding cluster method in CNN with the ArcMargin model. The framework consists of the following four processes.

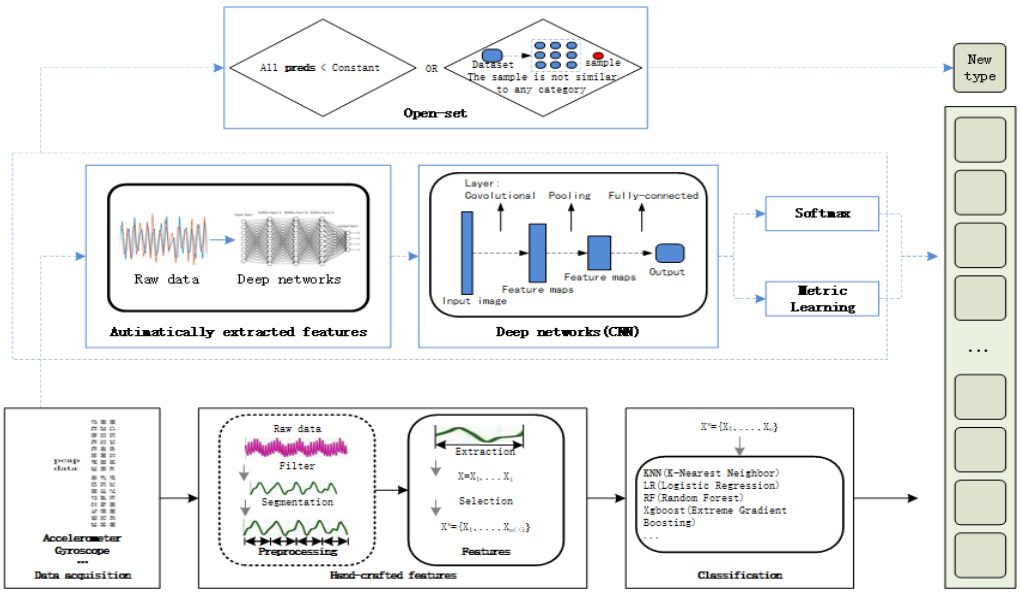

Figure 1: Network traffic classification framework

1) Traditional machine learning methods used on the datasets include KNN, logistic regression (LR) [Kleinbaum and Klein (2002)], random forest (RF) [Breiman (2001)], DT and XGBoost [Chen and Guestrin (2016)]. We converted network

traffic packets (PCAP) into hex values and then built some necessary features.

2) CNNs were used on three traffic datasets without acquiring any features. The hex values were sent to CNN directly.
3) The ArcMargin model can be regarded as a structure in which metric learning regularization is embedded in the objective function of a traditional CNN.
4) The cluster method is used in the classification model which improves open-set classification results.

After comparing different classification methods in general classification tasks and open-set classification tasks, some meaningful conclusions about network traffic data emerged, such as their distribution, different features and etc. Simultaneously, the results using different algorithms manifest that the metric learning improves the classification accuracy and also performs well with open-set tasks.

4 Methods

4.1 Traffic network model based on metric learning (ArcMargin)

4.1.1 Introduction of CNNs and ArcMargin

Before embedding metric learning (ArcMargin) in CNN models, we used CNNs only to solve the classification assignment as contrast group. Our model can be more easily understood if some basic concepts of CNNs and ArcMargin are first explained.

A. CNNs

CNNs have demonstrated impressive performance in many applications, such as sensing image analysis and abnormal performance detection [Chaib, Liu, Gu et al. (2017), Cheng, Zhou and Han (2016); Cheng, Li, Yao et al. (2017); Nogueira, Penatti and dos Santos (2017)]. As shown in Fig. 2, a typical CNN network consists of several network layers: convolutional layers, pooling layers, and fully connected (FC) layers.

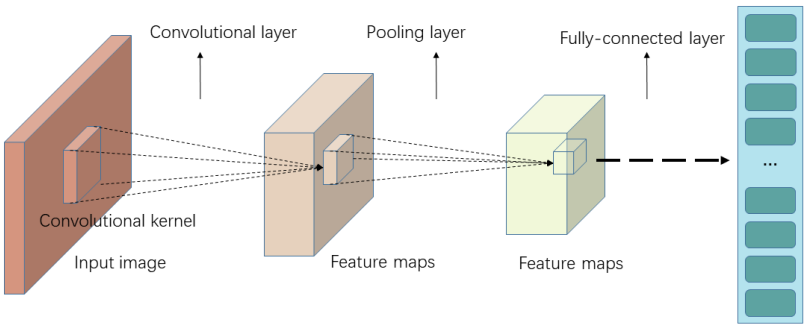

Figure 2: CNN structure

The convolutional layer is the most important layer in feature extraction. Its main purpose is to extract features automatically during training. The convolution layer in the front of the network structure extracts simple features, and the deeper convolution layer extracts the complex features that are computed from simple ones. Each unit in the convolution layer is connected to a local patch in the feature map of the previous layer by a set of

convolution kernels. This set of kernels is represented by the brown cube in Fig. 2. Then, the results output from the local weighted sum is mapped by a non-linear operation such as the rectified linear unit (ReLU) [Hara, Saito and Shouno (2015)]. All units in the feature maps share the same convolution kernel, while different convolution kernels can represent different features.

The pooling layer reduces the dimension of the feature representation and ensures the invariance of small translations or rotations. It is of great significance to image and object detection, as well as network traffic classification. The pooling layer can be created by computing some local nonlinear operation over feature maps. The max-pooling operation in the pooling layer computes the maximum of local units in feature map.

The role of the FC layer is to better summarize the message transmitted by previous layers which include lower-level features, in order to make the final decision. It is always used as the last few layers in the model.

B. ArcMargin

Softmax is the most widely used classification loss function:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \tag{1}$$

where x_i denotes the deep feature of the i-th sample, which is belongs to the y_i-th class. We set the embedding feature to 128 here. N denotes the batch size and n denotes the number of classes, b_j denotes the bias term, W is the weight and W_j is the j-th column of W.

However, there is still room for improvement in feature expression when using the Softmax loss function. If there are large intraclass appearance variations in training samples, there will still be a performance gap in the data identification, which cause the model cannot indicate the diversity among interclass samples and higher similarities among intraclass samples. Deng et al. [Deng, Guo, Xue et al. (2018); Sengupta, Chen, Castillo et al. (2016)].

Consequently, we set the bias b_j=0 and convert the logit as $W_j^T x_i = ||W_j||\ ||x_i \cos\theta_j||$, where θ_j is the angle of W_j and the x_i denotes a feature. According to Liu et al. [Liu, Wen, Yu et al. (2017); Wang, Xiang, Cheng et al. (2017)], the individual weight $||W_j=1||$ can be fixed by l_2 normalization and the embedding feature $||x_2||$ can be fixed by l_2 normalization and is then re-scaled to s. Thus, the embedding features learning from the model will be distributed on a hypersphere whose radius is s.

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}} \tag{2}$$

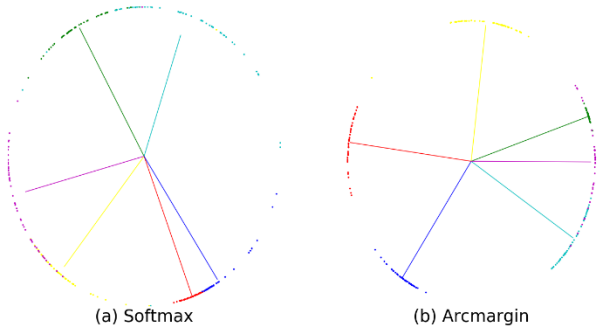

Figure 3: Illustration of the distance between six categories of sample sets under ArcMargin and softmax loss functions

As Fig. 3 shows, on the hypersphere, the embedded features are distributed regularly. An additive angular margin penalty m is added between x_i and W_{y_i}, which has the ability to enhance the intraclass compactness and interclass discrepancy. The additive angular margin penalty can be computed as the geodesic distance margin penalty gathering from the normalized hypersphere (ArcMargin) [Deng, Guo, Xue et al. (2018)].

In this paper, some network traffic from six different identities is used to train the feature-embedding model with the Softmax as well as ArcMargin loss. The results are displayed in Fig. 3.

4.1.2 ArcMargin classification model

Fig. 4 depicts the metric learning-based model using ArcMargin. The figure shows that after obtaining the raw input data, we send them into the CNN models. To date, CNNs have been mainly widely used in the computer vision field due to their superior ability to learn the spatial properties of an image pixel by pixel. In this paper, because the format of raw network traffic is a sequential format which is different from that of an image, a 1-D CNN is used in the structure rather than a 2-D CNN, which is widely used in the image. As shown above, the model consists of two convolutional layers, two max pooling layers, two local response normalization (LRN) layers and a densely connected layer with a final optional unit consisting of a Softmax classifier and metric learning part. These two different final output layers are used for a set of comparative experiments.

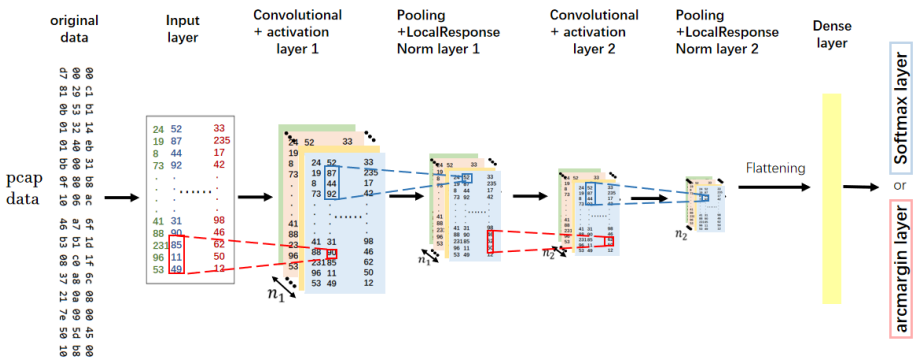

Figure 4: Experimental procedure illustration of CNN model with metric learning

In public processing layers, we first used 32 convolution kernels with the size of [25, 1] and step size of 1 to extract the first feature on the input data with the size of [1, 1, 784]. The results output from the convolutional layer is delivered to an activation function. ReLU is used here (described in Section 4). Then, the max pooling layer processes the results previously output.

After that, an LRN layer is embedded to punish abnormal responses at the end of the first convolutional layer, with the aim of obtaining better generalized results. The results will subsequently cross the second convolutional layer the difference of which is that there are 64 neurons. Finally, the data across a densely connected layer and a fully connected layer with dropout. After the data goes through the above two convolutional layers, the output label is obtained by the Softmax classifier or a metric learning module.

4.2 Open-set classification model

Open-set assignment is a common problem in the task of data classification and occurs when the test dataset contains new classes of data that do not appear in the training samples. To solve this problem, the following two methods have been adopted.

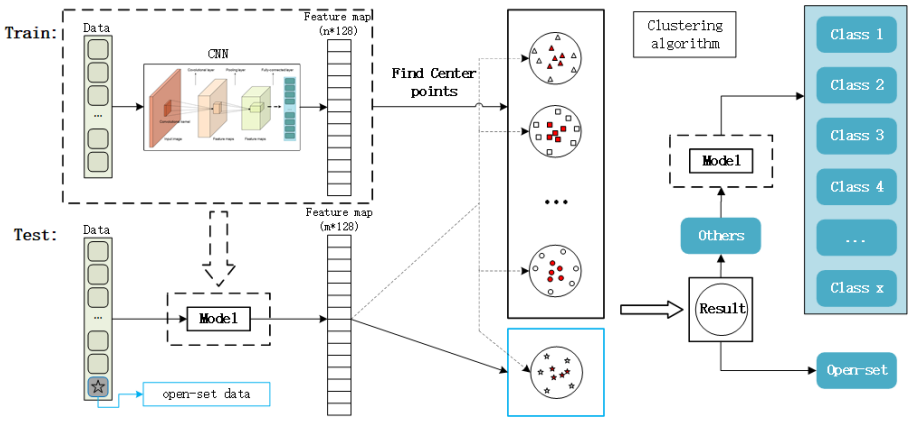

Figure 5: Structure of open set classification model based on deep features

A. Identify categories by setting thresholds

The model we proposed above is based on CNNs. In view of its structure, we attempted to output the probability of each category from the classification result. In our experiments, we used two methods to achieve the classification results: Softmax and Arcmargin. The probability was gained with both methods. After this step, we set a threshold for each category on the basis of the data distribution in each method. Clearly, the comparison of thresholds and the probability of sample data determine whether a sample belongs to the open-set category. In this method, the selection of threshold determines whether the data can be proposed divided. Data distribution and model effects also play a significant role. To overcome the disadvantages of this method, we proposed another model: a deep feature-based cluster model.

B. Deep features-based cluster model on open-set classification

Fig. 5 shows the open-set identification process. We acquired deep feature layer based on the CNN model as described above and used it as cluster base sets. This model can be explained in five steps as follows:

Step 1: The first step can be regarded as a training process. We use training data without open-set samples to train this model under CNNs. To build the cluster base sets, we acquire a deep feature layer from the output of the convolution layer in the completed training model.

Step 2: In this step, a few open-set samples (obtained through expert knowledge or business knowledge) are used to obtain deep feature layer from the model trained in Step 1. Until now, the cluster base sets are well constructed.

Step 3: Cluster and data center selection: In this step, cluster base sets will be used to select data centers in each category. In this paper, we apply the K-means++algorithm [Yu and Jian (2012)] and similarity analysis to obtain several center samples in each category.

Step 4: This step can be regarded as open-set sample identification. Test datasets are fed into the model previously trained model and their deep features are generated. Then, we calculate the similarities between test samples and center samples built in Step 3 in each category and obtain the average similarity result. According to the average similarity result and the set threshold, we can determine whether a sample belongs to open-set category.

Step 5: After Step 4, we keep samples that do not belong to open-set category in test dataset. And then we feed these samples into the classification model trained in Step 1 and obtain classification results. Finally, the results outputted in Steps 4 and 5 make up the final classification results.

Table 1: Data distribution of dataset ISCX-VPN-NONVPN-2016

Traffic	Content	Number	Percentage
Email	POP3, SMPT and IMAP	26844	14.94%
Chat	ICQ, Skype, AIM, Hangouts and Facebook	33978	18.92%
Streaming File	YouTube and Vimeo	26682	14.85%
Transfer	SFTP, FTPS and Skype using Filezilla	30000	16.70%
VoIP	Facebook, Skype and Hangouts voice calls (1h duration)	30000	16.70%
P2P	Transmission (BitTorrent) and uTorrent	32130	17.89%

5 Experiments and comparison of results

5.1 Data description

We used three different types of public datasets in the experiment which included both normal traffic and attack traffic. The attack traffic is divided into several categories. These datasets are described as follows:

Dataset 1 (ISCX-VPN-NONVPN-2016): The first dataset is ISCX-VPN-NONVPN network traffic. This dataset is meant to be a representative dataset of real-world network traffic in ISCX. The owners of dataset define a set of tasks to collect network information and assure the diversity and quantity of dataset at the same time. Tab. 1 presents the complete list of different types of traffic and applications included in the dataset. For each traffic type, the data captures a regular session and a session over VPN. There are 14 traffic categories in the dataset, such as VPN-P2P, VOIP, P2P and VPN-VOIP. The detailed distribution of the traffic is shown in Tab. 1.

Dataset 2 (CIC-IDS-2017): Open-sourced by Canadian Institute for Cybersecurity in 2017. CICIDS2017 is a dataset for intrusion detection and intrusion prevention. Sharafaldin et al. [Sharafaldin, Lashkari and Ghorbani (2018)] designed a real attack application scene to extract traffic data from both the attack network and the victim network. It extracts normal traffic and different kinds of attack traffic in a week and produces real-world PCAP file data as output. On Monday, there is only normal traffic. From Tuesday to Friday, several attacks occur and the collectors extract all information generated in this period. This paper extracts eleven types of attack flows from the CICIDS2017 and uses them as the training and test samples to build the classification model. The traffic information can be extracted in two ways: by directly extracting the flow raw data and by extracting the statistical features. The distribution of flows is indicated in Tab. 2.

Dataset 3 (CIC-IDS-2012): With the evolution of network behavior patterns and intrusions, network behavior is becoming more diversified. This dataset generates dynamic, rather than static network performance data. ISCX proposed a systematic approach to generate the required datasets. Various multi-stage attack scenarios were applied to complete this task. These anomalous portions fall into four categories: DDoS, Brute_Force_SSH, Infiltrating_Transfer and HTTPDoS. The intrusion detection dataset,

UNB ISCX 2012, offers features such as data labels, realistic network traffic, total interaction captures and complete capture. The distribution of the traffic is shown in Tab. 3.

Table 2: Data distribution of dataset CIC-IDS-2017

Flow Types	Number	Percentage
botnet	66049	9.16%
ddos	110639	15.34%
ftp Transfer	83019	11.51%
goldeneye	18897	2.62%
heartbleed	26585	3.69%
hulk	166451	23.08%
portscan	68523	9.50%
slowhttptest	27994	3.88%
slowloris	34983	4.85%
sql_injection	56469	7.83%
ssh	61668	8.55%

Table 3: Data distribution of dataset CIC-IDS-2012

Flow Types	Number	Percentage
Brute_Force_SSH	14056	7.61%
DDos	45019	24.37%
HTTPDos Transfer	6533	3.54%
Infiltrating_Transfer	19156	10.37%
Normal	100000	54.12%

5.2 Data processing

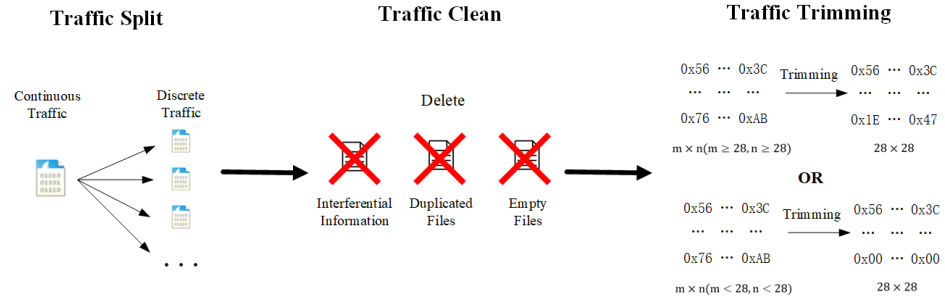

Figure 6: Data processing flow

Before feeding data into the classification model, traffic packets need to be converted into the vectors with the same length of 784. As Fig. 6 shows, we convert raw traffic data into CNN input data. This is a three-step process of traffic split, traffic clean and traffic trimming.

Traffic Split: This step splits a continuous raw traffic into multiple discrete traffic units that refer to each flow's information, including protocol, source IP, source port, destination IP and destination port.

Traffic Clean: This is the step to remove the interferential information in traffic

packages. First, we randomize MAC addresses in the data link layer and the IP addresses in the IP layer. Next, we refine the data to eliminate duplicate and empty files, since they do nothing other than interfere with the learning ability of the network.

Traffic Trimming: The purpose of this step is to trim all files into a uniform length. Since all layers contain some traffic feature information, for this paper, we extract information from all layers, which means that we trim file data and layer data. If the file is larger than 784 bytes, it will be trimmed to 784 bytes. If the file is smaller than 784 bytes, we add 0x00s at the end of the file to make it 784 bytes.

5.3 Classification results

5.3.1 Results output from ArcMargin model

To evaluate the experiment's results, we collected four parameters: Recall, Precision, Macro-f1 and Weighted-f1. They are explained as follows:

$$\text{Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP}{TP+FN} \tag{3}$$

$$\text{Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP}{TP+FP} \tag{4}$$

$$\text{Macro}-f1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2*\text{ Precision }*\text{Recall}}{\text{Precision+Recall}} \tag{5}$$

$$\text{Weighted}-f1 = \sum_{i=1}^{N} w\frac{2*\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \tag{6}$$

- **True positive (TP) refers to** the number of correctly identified positive samples.
- **True negative (TF) refers to** the number of correctly identified negative samples.
- **False positive (FP) refers to** the number of wrongly identified positive samples.
- **False negative (FN) refers to** the number of wrongly identified negative samples.

N is the number of categories of data and *w* is the weight of each sample quantity to the total data quantity.

Table 4: Experimental results of dataset ISCX-VPN-NONVPN-2016

Model name	Recall	Precision	Macro-f1	Weighted-f1
CNN	0.9548	0.9558	0.954	0.9543
KNN	0.6864	0.6719	0.6715	0.6812
LR	0.2684	0.2025	0.228	0.2399
RF	0.8526	0.8521	0.8491	0.8557
DT	0.8458	0.8404	0.8416	0.8488
XGBoost	0.8	0.8493	0.8046	0.8132
CNN+metric learning	**0.9857**	**0.9853**	**0.9853**	**0.9856**

Tab. 4 shows the comparison with the traditional machine learning methods. It can be seen that the four indicators of CNN network perform better, the f1-score of which can reach around 95%. When we add metric learning on the basis of CNN network, we find that the four indicators are increased by about 3%, inferring that the addition of metric

learning has a positive influence on the model.

In order to verify the generality of the proposed metric learning model, we conduct the same test on two other datasets. As Tabs. 5 and 6 show, the results are not much different from the previous ones, which prove that adding metric learning can help to better classify network traffic.

Table 5: Experimental results of dataset CIC-IDS-2017

model name	recall	precision	macro-f1	weighted-f1
CNN	0.9661	0.9693	0.9659	0.9659
KNN	0.5995	0.6031	0.5969	0.597
LR	0.1113	0.0275	0.0378	0.041
RF	0.9393	0.9398	0.9393	0.9401
DT	0.9504	0.9498	0.9496	0.9504
XGBoost	0.9065	0.9111	0.9039	0.9067
CNN+metric learning	**0.9933**	**0.9934**	**0.9933**	**0.9933**

Table 6: Experimental results of dataset CIC-IDS-2012

model name	recall	precision	macro-f1	weighted-f1
CNN	0.9896	0.9896	0.9896	0.9896
KNN	0.9582	0.9593	0.9586	0.9593
LR	0.6089	0.7483	0.5419	0.4697
RF	0.9775	0.9762	0.9768	0.9766
DT	0.9504	0.9498	0.9496	0.9504
XGBoost	0.9749	0.9781	0.9764	0.9759
CNN+metric learning	**0.9971**	**0.9971**	**0.9971**	**0.9971**

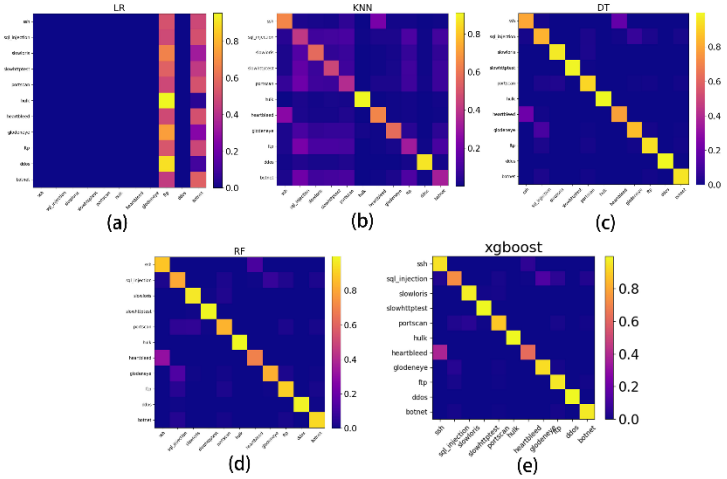

Figure 7: Confusion matrices of classification results using different algorithms

5.3.2 Comparison of experiments and results

To compare and validate the results of the proposed classification model, we designed several comparative experiments with traditional machine learning methods. We also

used a CNN model without ArcMargin to accomplish the classification task. (Aside from not having the final ArcMargin layer the other processing layers are the same as those of the ArcMargin model, so this will not be explained here.) This paper is mainly concerned with five algorithms: KNN, LR, DT, RF and Xgboost).

In the traditional machine learning process, it is important to send the right feature into the model for building a good classifier. However, it is usually time- and resource-consuming to extract complete and effective features from network traffic data. Consequently, we tried to find some universal features based on a novel method to acquire the most effective features in different malicious traffic types.

In this paper, the network traffic is regarded as single direction flow, one which can be described as from attacking host to the attacked host. Tab. 7 describes several details of traffic features. ByteDis (called byte dis in Tab. 7) is a 256-dimension vector, and the value of each dimension can be regarded as the number of corresponding bytes in each flow. For instance, ByteDis 0×63 (99) is equal to 25, which signifies that 0×63 (99) appears 25 times in all flow samples. In addition, the total number of dimensions in the traffic feature is 643. Subsequently, we feed feature vectors into the machine learning models mentioned above.

After processing the data, we fed the data into several traditional machine learning algorithms. Fig. 7 displays the confusion matrices of all the algorithms based on Dataset 2 described above. A confusion matrix is a method for summarizing the prediction results of the classification model, revealing the prediction and real label distribution among all the classes. If all the data were classified accurately, the matrix would be shaped as a diagonal. Therefore, the more elements, the worse the results. As shown in Fig. 6, DT is the best performing algorithm, and LR is the worst performer. Fig. 8 displays the comparison of the f1-score in the classification results between different algorithms on Dataset 2. The results are also presented in Tab. 7.

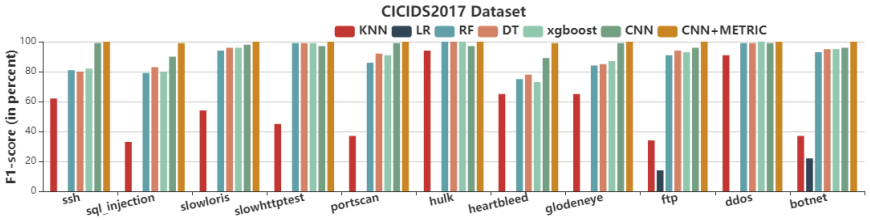

Figure 8: Comparison of the classification effect of different algorithms

5.4 Open-set classification results

As shown in Tabs. 4-6, indicators of recall, precision, macro-f1 and weighted-f1 are all over 0.98, which can be regarded as a good performance for the model. However, when open-set samples are added into test data, a model that can only distinguish existing categories will not work. As a result, we used the open-set classification model described in Section 4.1.1. First, we identified categories by setting thresholds, so as to solve open-set problem by using the probabilities output from Softmax and ArcMargin in the CNN model. The results of classifying Dataset 1 are listed in rows 1 and 2 in Tabs. 8 and 9.

Table 7: Details of the feature repository

Feature name	Description
sp	Source port
dp	Destination port
inbyte_cnt	Inbound bytes
outbyte_cnt	Outbound bytes
inpkt_cnt	Inbound packets
outpkt_cnt	Outbound packets
pkt_length	Long sequence packets in the first hundred packets in a flow
pkt_time	First hundred packets of arrival sequence in a flow
byte_dis	Number of per byte in a flow
byte_persec	Number of bytes per second
max_pkt	Number of the maximum packet
min_pkt	Number of the minimum packet
mean_pkt	Average length of the packets
std_pkt	Variance of the length of the packets
max_bd	Max value in in byte_dis
min_bd	Minimum value in the byte_dis
entropy_bd	Entropy of byte distribution
std_bd	Variance of byte distribution
duration	Duration of a flow

As can be seen from the tables, the threshold setting method used in the model that performed well in previous classification task does not operate as well in the open-set task. The results in Tab. 9 show that the precision, recall and f1-score of the open-set category are only 0.10 in the CNN model and 0.30 in the ArcMargin model. From this, it can be concluded that there are no obvious performance differences between the Softmax and ArcMargin probability outputs in each category.

Next, we added the cluster model into open-set classification task. As can be seen in row 3 of both Tabs. 8 and 9, on average, precision, recall and f1-score all improved significantly in both open-set categories as well as in all other categories. To support this crucial result, we output the feature similarities of each sample class with open-set class in Fig. 9, where the transverse axis gives the similarities between each sample and center samples in every category, and the size of the colored circle gives the number of samples in the similarity interval. Fig. 9 shows that most open-set samples have similarities with open-set centers close to 1, but most other classes are below 0.8, indicating that we could use the cluster method to distinguish the open-set class from others. These results also proved our hypothesis.

Table 8: Experimental results of all categories in open-set experiments

Model name	Recall	Precision	Macro-f1	Weighted-f1
CNN	0.7097	0.7045	0.7008	0.6955
CNN+metric learning	0.7246	0.7374	0.7317	0.7219
CNN+metric+cluster	**0.8482**	**0.8567**	**0.8523**	**0.8619**

Table 9: Experimental results of open-set category in open-set experiments

Model name	Recall	Precision	f1-score
CNN	0.0077	0.0160	0.0105
CNN+metric learning	0.3158	0.3043	**0.3040**
CNN+metric+cluster	**0.7580**	**0.6402**	**0.6308**

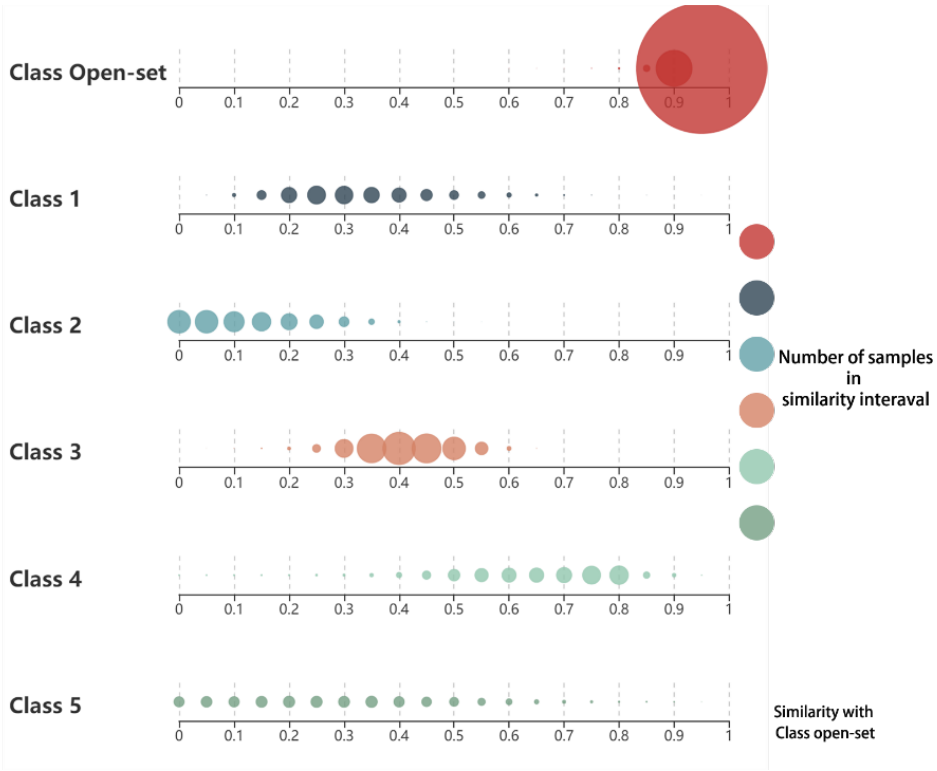

Figure 9: Similarities in open-set of each class

6 Conclusions

In this paper, we propose a new method to identify malicious and other abnormal behaviors in network traffic. With the purpose of improving the classification results, we built a CNN model with metric learning (ArcMargin) embedded. As the results show, our model performed well in different indicators. To further verify the effectiveness of this model, we designed a classification verification framework that applies several machine learning algorithms and the original CNN model on different datasets. The experiments demonstrate that embedding metric learning into CNNs improves the performance of the traffic classification model. We also dealt with the open-set classification problem and achieved improved results by adding a deep feature cluster method into the above metric model.

In the future, we hope to design a better network traffic classification model based on metric learning and make it applicable to a wider range of network traffic data and train it on other networks including RNN and LSTM. At the application level, we also intend to test our model on real traffic data and in actual production. Finally, we will continue to improve the open-set classification results using different methods, in order to increase the possibility of applying this model into actual and real-time network traffic.

Funding Statement: This work was supported by the National Natural Science Foundation of China (61871046).

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

References

Alhawi, O. M. K.; Baldwin, J.; Dehghantanha, A. (2018): Leveraging machine learning techniques for windows ransomware network traffic detection. *Cyber Threat Intelligence*, vol. 70, no. 1, pp. 93-106.

Breiman, L. (2001): Random forests. *Machine Learning*, vol. 45, no. 1, pp. 5-32.

Chaib, S.; Liu, H.; Gu, Y.; Yao, H. (2017): Deep feature fusion for VHR remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4775-4784.

Chen, T.; Guestrin, C. (2016): Xgboost: a scalable tree boosting system. *Proceedings of the ACM SigKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794.

Cheng, G.; Li, Z.; Yao, X.; Guo, L.; Wei, Z. (2017): Remote sensing image scene classification using bag of convolutional features. *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1735-1739.

Cheng, G.; Yang, C.; Yao, X.; Guo, L.; Han, J. (2018): When deep learning meets metric learning: remote sensing image scene classification via learning discriminative CNNs. *IEEE Transactions on Geoscience Remote Sensing*, vol. 56, no. 5, pp. 2811-2821.

Cheng, G.; Zhou, P.; Han, J. (2016): Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405-7415.

Chopra, S.; Hadsell, R.; LeCun, Y. (2005): Learning a similarity metric discriminatively, with application to face verification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 539-546.

Deng, J.; Guo, J.; Xue, N.; Zafeiriou, S. (2018): Arcface: additive angular margin loss for deep face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690-4699.

Di, C. (2018): Learning automata based SVM for intrusion detection. *Proceedings of the International Conference in Communications*, pp. 2067-2074.

Finsterbusch, M.; Richter, C.; Rocha, E.; Muller, J. A.; Hanssgen, K. (2014): A survey of payload-based traffic classification approaches. *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1135-1156.

Gao, N.; Gao, L.; Gao, Q.; Wang, H. (2014): An intrusion detection model based on deep belief networks. *Proceedings of the International Conference on Advanced Cloud and Big Data*, pp. 247-252.

Gul, A. B.; Yoan, M.; Aapo, K.; Ian, O.; Silke, H. et al. (2018): Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space. *Cognitive Computation*, vol. 10, no. 5, pp. 848-863.

Hara, K.; Saito, D.; Shouno, H. (2015): Analysis of function of rectified linear unit used in deep learning. *Proceedings of the International Joint Conference on Neural Networks*,

pp. 1-8.

Huang, G. B.; Lee, H.; Learned-Miller, E. (2012): Learning hierarchical representations for face verification with convolutional deep belief networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2518-2525.

Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. (2016): A deep learning approach for network intrusion detection system. *Proceedings of the EAI International Conference on Bio-inspired Information and Communications Technologies*, pp. 21-26.

Kleinbaum, D. G.; Klein, M. (2002): Logistic regression (a self-Learning text). *Technometrics*, vol. 45, no. 1, pp. 109.

Kumar, S.; Dharmapurikar, S.; Fang, Y.; Crowley, P.; Turner, J. (2006): Algorithms to accelerate multiple regular expressions matching for deep packet inspection. *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 339-350.

Lin, Q.; Zhang, H.; Lou, J. G.; Yu, Z.; Chen, X. (2016): Log clustering-based problem identification for online service systems. *Proceedings of the IEEE/ACM International Conference on Software Engineering Companion*, pp. 102-111.

Lins, F.; Damasceno, J.; Silva, B.; Medeiros, R.; Souza, A. et al. (2012): Towards an approach to design and enforce security in web service composition. *International Journal of Web Engineering Technology*, vol. 7, no. 4, pp. 323-357.

Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B. et al. (2017): Sphereface: deep hypersphere embedding for face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 212-220.

Nejad, M. B.; Shiri, M. E. (2019): A new enhanced learning approach to automatic image classification based on Salp Swarm Algorithm. *Computer Systems Science and Engineering*, vol. 34, no. 2, pp. 91-100.

Nogueira, K.; Penatti, O. A.; dos Santos, J. A. (2017): Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, vol. 61, no. 1, pp. 539-556.

Peterson, L. E. (2009): K-nearest neighbor. *Scholarpedia*, vol. 4, no. 2, pp. 1883.

Safavian, S. R.; Landgrebe, D. (1991): A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660-674.

Salakhutdinov, R. (2015): Learning deep generative models. *Annual Review of Statistics and Its Application*, vol. 2, no. 1, pp. 361-385.

Schultz, M. G.; Eskin, E.; Zadok, F.; Stolfo, S. J. (2002): Data mining methods for detection of new malicious executables. *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 38-49.

Sengupta, S.; Chen, J. C.; Castillo, C.; Patel, V. M.; Chellappa, R. et al. (2016): Frontal to profile face verification in the wild. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 1-9.

Sharafaldin, I.; Lashkari, A. H.; Ghorbani, A. A. (2018): Toward generating a new Intrusion detection dataset and intrusion traffic characterization. *Proceedings of International Conference on Information Systems Security and Privacy.*

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. et al. (2014): Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9.

Tariq Usman. (2019): Intrusion detection and anticipation system (IDAS) for IEEE 802.15.4 devices. *Intelligent Automation and Soft Computing*, vol. 25, no. 2, pp. 231-242.

Wang, F.; Xiang, X.; Cheng, J.; Yuille, A. L. (2017): Normface: 2 hypersphere embedding for face verification. *Proceedings of the ACM International Conference on Multimedia*, pp. 1041-1049.

Wang, P.; Ye, F.; Chen, X.; Qian, Y. (2018): Datanet: deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, vol. 6, pp. 55380-55391.

Wang, W.; Zhu, M.; Zeng, X. W.; Ye, X. Z.; Sheng, Y. Q. (2017): Malware traffic classification using convolutional neural network for representation learning. *Proceedings of the International Conference on Information Networking*, pp. 712-717.

Yang, L.; Jin, R. (2006): *Distance Metric Learning: A Comprehensive Survey*. Michigan State University, vol. 2, no. 2, pp. 4.

Yu, X.; Jian, Y. (2012): Partitive clustering (kÃÂÂmeans family). *Wiley Interdisciplinary Reviews Data Mining Knowledge Discovery*, vol. 2, no. 3, pp. 209-225.

Zeng, Y.; Gu, H.; Wei, W.; Guo, Y. (2019): Deep-full-range: a deep learning-based network encrypted traffic classification and intrusion detection framework. *IEEE Access*, vol. 7, pp. 45182-45190.

Zhang, H.; Yi, Y.; Wang, J.; Cao, N.; Duan, Q. (2018): Network security situation awareness framework based on threat intelligence. *Computers, Materials and Continua*, vol. 56, no. 3, pp. 381-399.

Zhang, J.; Qin, Z.; Yin, H.; Ou, L.; Hu, Y. (2016): IRMD: malware variant detection using opcode image recognition. *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, 1175-1180.