

Visual Analytics for Network Events Classification in LAN With Deep Convolutional Neural Network

Yuwei Sun (✉ sywtokyo@hongo.wide.ad.jp)

Tokyo Daigaku <https://orcid.org/0000-0001-7315-8034>

Hideya Ochiai

Tokyo Daigaku

Hiroshi Esaki

Tokyo Daigaku

Research

Keywords: Cybersecurity, LAN, Visual analytics, Deep learning, Convolutional neural network

Posted Date: June 11th, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-34589/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Visual Analytics for Network Events Classification in LAN With Deep Convolutional Neural Network

Yuwei Sun^{1*}, Hideya Ochiai¹ and Hiroshi Esaki¹

Abstract

This article illustrates a method of visualizing network traffic in LAN based on the Hilbert Curve structure and the array exchange and projection, with nine types of protocols' communication frequency information as the discriminators, the results of which we call them feature maps of network events. Several known scan cases are simulated in LANs and network traffic is collected for generating feature maps under each case. In order to solve this multi-label classification task, we adopt and train a deep convolutional neural network (DCNN), in two different network environments with feature maps as the input data, and different scan cases as the labels. We separate datasets with a ratio of 4:1 into the training dataset and the validation dataset. Then, based on the micro scores and the macro scores of the validation, we evaluate performance of the scheme, achieving macro-F-measure scores of 0.982 and 0.975, and micro-F-measure scores of 0.976 and 0.965 separately in these two LANs.

Keywords: Cybersecurity, LAN, Visual analytics, Deep learning, Convolutional neural network

1 Introduction

Recently, with more and more objects connected into the internet, more information is shared through the format of digital data on the internet. In the era of big data, resilient and robust network systems with the ability of protecting privacy of users are critical. In the local area network (LAN), a malware delivered, for example by phishing e-mails, has the ability to intrude and expand into the other hosts, causing leaky of personal information. Through delivering malware and further spreading to social media, messaging services and applications, attacks like this can affect multiple aspects of personal life.

With an explosion of information, the manipulation of network systems is becoming more and more difficult. Moreover, the digital property of it adds up to the complicity of explaining certain events in networks. With respect to big data of network traffic, visual analytics is usually used to convert it into visual information, which has been adopted

in research for enhancing the explicitly of tasks such as anomaly detection from network traffic.

On the other hand, with the advancement of machine learning, especially deep learning, it is considerable to utilize deep learning for dealing with onerous analyses of enormous network traffic. In many studies, machine learning methods such as support vector machine and neural networks are used to detect anomaly in LAN, telling the status of the network is normal or abnormal, which is mainly focused on detecting of anomaly instead of digging into the hidden explicitly of network events.

In this research, we focus on the visualization of network traffic by selecting suitable discriminators and a further classification between different events in LAN with a deep learning method.

When malware intrude into a LAN, and try to expand into (or steal some data from) the other hosts in the LAN, it tries to access some specific TCP or UDP ports of all the hosts. This kind of activities is a network event. In this paper, we focus on ARP scan, TCP scan, UDP scan, and those scans to specific ports as the types of network events, which we target to classify by our scheme.

We propose a scheme that visualizes network traffic of different network events in LAN by generating feature maps based on a structure called Hilbert Curve, compressing protocol information during a certain duration into one

*Correspondence: sywtokyo@hongo.wide.ad.jp

¹Graduate School of Information Science and Technology, University of Tokyo, Tokyo, Japan

Full list of author information is available at the end of the article

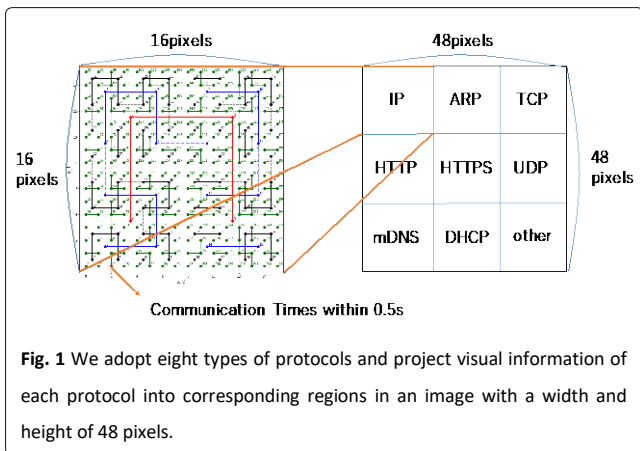


image (Fig. 1). Then we simulate eight types of network events in LAN, generating feature maps as the dataset for training a DCNN model thus solving this multi-label classification task. Our model detects and classifies these eight types of attacks using visual analytics, by which we could provide some explicitly for the occurrence of a specific type of attacks.

This paper is organized as follows. Section 2 discusses related works about network anomaly detection with machine learning. Section 3 provides an overview of the scheme including the visualization of network traffic based on protocol communication information and classification of network events using DCNN. Section 4 presents the performance evaluation of the scheme based on validation scores of the model. Section 5, we discuss the contribution of this research and show some limits which can be improved through future work. Section 6, we conclude the paper and the result of this research.

2 Related work

Anomaly detection in computer networks attracts many attentions, with more than 40 years of evolution [1]. With the rapid growth and increasing complexity of network infrastructures, and the evolution of attacks, identifying and preventing networks attacks are becoming more and more challenging. Traditional approaches to this issue include an application of several knowledge-based rules on network communication, and once these rules are satisfied, a network event can be considered malicious.

Several traditional machine learning methods such as support vector machine (SVM) and neural network (NN) have been used to address these issues of network anomaly detection in both personal computers and critical infrastructure [2][3]. Moreover, Yang et al. [4], they use

restricted Boltzmann machine (RBM) to extract high-level features of traffic data and train SVM with stochastic gradient descent (SGD) for classification of these features. Asmaa et al. [5] presented a comprehensive discussion of using RBM for feature learning and a classifier for anomaly detection. Salama et al. [6] presented an intrusion detection hybrid scheme using deep belief network (DBN) and SVM, classifying the intrusion into two clusters: normal or attack. They adopt DBN for reducing dimension of features and SVM for the classifier. They evaluated this scheme with the NSL-KDD dataset [7] and achieved an accuracy of 0.9.

However, limits of these methods in dealing with big data of network traffic and the lack of explainability have shown disadvantages in solving more complicated detection problems in networks. Furthermore, lots of studies adopt a supervised method, training a classifier with data labeled as normal or abnormal, so that knowledge about anomaly can be constructed thus detected, which however can be limited.

With the advancement of deep learning in recent years, large-scale data analyses on network traffic data have become feasible and been showing great performance. For instance, Saxe et al. [8] proposed a deep neural network (DNN) based malware detector that employs two-dimensional binary features to detect malware. Yousefi et al. [9] gave out a generative feature learning-based approach for malware classification, where latent features from the hidden layer of an autoencoder are used for anomaly detection.

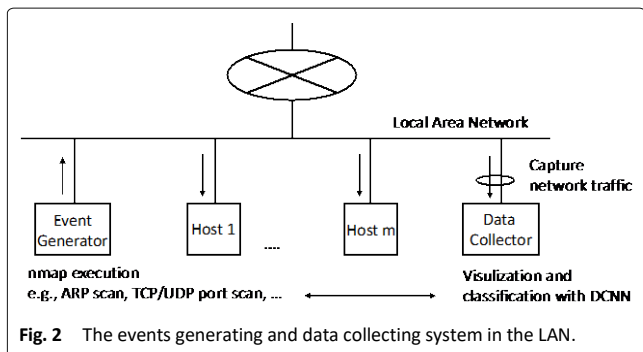
Different from the aim of detecting anomaly in networks in former research, we proposed an approach to classify different types of network events in LAN. And instead of only labeling data with the normal and abnormal, we adopt eight types of network event clusters, thus building a multi-label classifier in a dynamic way. Moreover, we focus on the representation of network traffic in LAN by 2-D images data based on protocol communication frequencies. Then we build a dataset to train a deep convolutional neural network (DCNN) for solving this multi-label classification problem. Consequently, this system is supposed to have the ability to deal with big data of network traffic, at the same time, have a stable training progress for the multi-label classification of network events in various LAN environments.

3 Methods

3.1 Network traffic visualization in LAN with Hilbert curve

The system we use to collect network traffic with different events, mainly consists of two terminals for manipulating and any other hosts connected in a LAN. In detail, one terminal

is used as an event generator for implementing scan commands thus generating various network events in the LAN and another one is used to collect network traffic with these events simultaneously (Fig. 2).



In this research, we use a tool called tcpdump for network traffic collecting. And all the traffic data broadcasted in the LAN or data sent directly to the data collector are collected and processed with a daily base. For generating these network events, we adopt a tool called nmap, which can be used to send various commands of scans in LAN. Moreover, the implementing of a command for each network event lasts for two whole days. Then we extract protocol and time stamp information from the collected traffic data in order to further visualize time-series features hidden in the network traffic.

We generate a feature map for a constant recording time unit. To define this recording time unit, first, we introduce a concept called the fineness, which shows how finely we should analyze the information hidden in big data of network traffic. Then we define a constant time unit for recording traffic data and generating each feature map as in (1).

$$T = T_{st} \times \text{fineness} \times \left(\frac{\text{size}}{St}\right)^2 \tag{1}$$

Here, T_{st} (time standard) is a standard interval for the recording time unit, which is defined as 64 seconds in this research. St is a parameter of the basic segment, showing the standard size of a feature map with a value of 8 pixels. The parameter of size is used to compute the length and width of a feature map.

For example, when the fineness is 1.0, a feature map with a size of 16 pixels represents a recording time unit of 256 seconds (around four minutes). On the other hand, if we use a size of eight pixels instead, the recording time unit will be 64 seconds. As a result, it is possible to use these parameters (fineness and size) to bring features of traffic data during a time span with a fixed length into an image with different sizes. At the same time, we can use an image with a fixed

size to represent traffic data during time spans with different lengths.

In detail, we compute how many times of communication for each protocol has been recorded during every recording time unit, as the discriminators for representing features of network traffic with different events. Considering explainability of this network events classification, from all protocol information, communication frequency information from IP, ARP, TCP, HTTP, HTTPS, UDP, mDNS, DHCP and the others is extracted and clustered. Then, as a visualizing method, we convert communication frequency information of these protocols into pixel values using (2), where we use these values to show the frequency of a specific protocol’s communication during recording time units.

$$p_i = \frac{c_i}{\text{Max}(c)} \times 255 \tag{2}$$

Where c_i is the frequency of a protocol’s communication within a recording unit, and the denominator is the maximum from all frequency values within the duration for generating a feature map. p_i represents the value of the pixel point.

Considerable visual analytics should have properties including intense information representation, showing time-related relationships between data, and a computable two-dimension structure with respect to a DCNN model. As a result, a geometric structure called Hilbert curve is adopted in this research, by which we compress time-related features of network traffic data into a 2D image, keeping the hidden relationships between data as well. We project the pixel values computed from the communication frequencies as pixel points into an image. Here, a fineness value discussed above of 0.5 is adopted, with an image size of 16 pixels. Thus each feature map of a protocol consists of 1024 records, each of which shows features of a specific protocol’s communication within 0.5 seconds (Fig. 3).

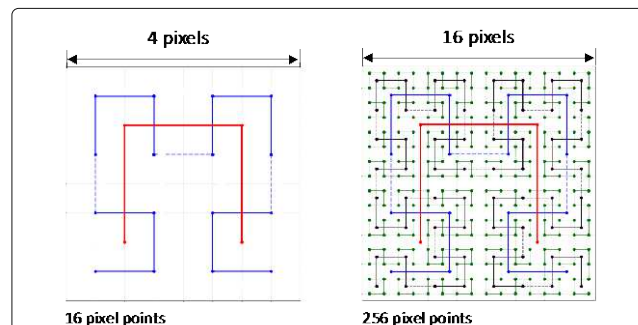


Fig. 3 The geometric structure of Hilbert curve, where frequency information is projected into each position of it through pixel values. Every four pixel points in the same neighborhood can be represented with a central point. Hence, we are able to compress big data of network traffic with the structure of the Hilbert curve.

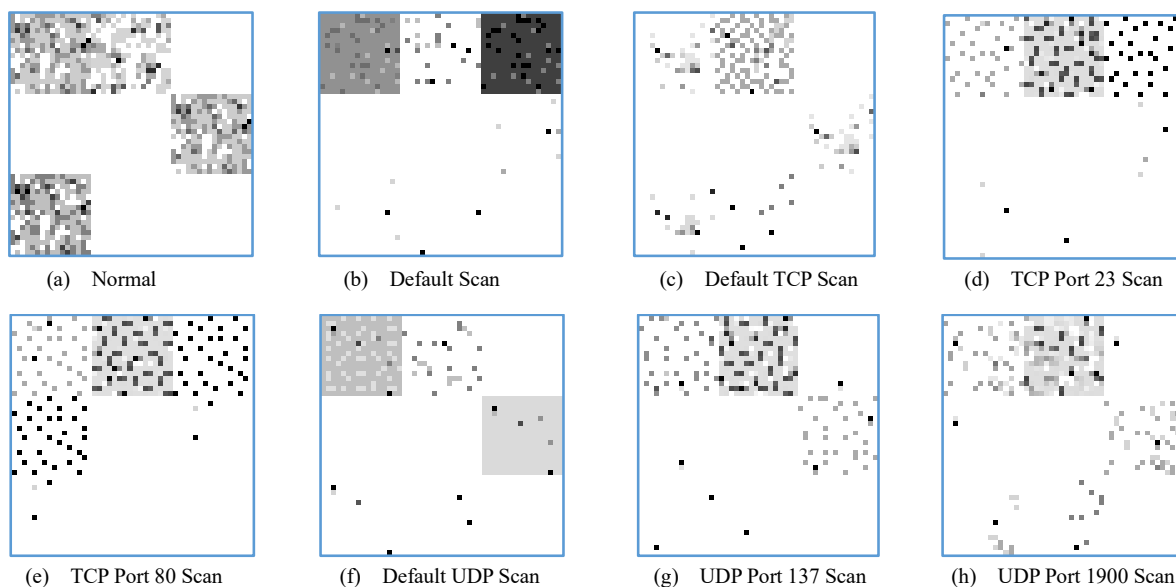


Fig. 4 Line 1, from left to right: Feature maps of events of the normal, default scan, default TCP scan, specific scan of TCP port 23; Line 2, from left to right: Feature maps of events of the specific scan of TCP port 80, default UDP scan, specific scan of UDP port 137, specific scan of UDP port 1900. (when the fineness is 0.5 and the size is 16)

Here, as a method of expressing time-related features of an event in LAN, we put feature maps of nine protocol clusters into an image (Fig. 1). That means, statistical information of nine types of protocols collected in the LAN can be represented in different regions of an image (48×48) through the array exchange and projection. Moreover, considering the computing cost of the DCNN model and the fineness of features representation, an image size of 16 and a fineness value of 0.5 is adopted in this research. At last, we can get a feature map representing time-sequential traffic data within 128 seconds in the LAN of a specific network event, by using information of nine types of protocols.

3.2 Network events generating and traffic data collecting

In this research, we manipulate seven types of scan commands in LAN covering arp scan, tcp scan, scan of tcp port 23, scan of tcp port 80, udp scan, scan of udp port 137 and scan of udp port 1900. We implemented each of these events by using corresponding nmap commands shown in the following through the event generator in the LAN.

- scan (arp): nmap [Network]
- scan (tcp): nmap -sT [Network]
- scan (tcp port 23): nmap -sT -p 23 [Network]
- scan (tcp port 80): nmap -sT -p 80 [Network]
- scan (udp): nmap -sU [Network]

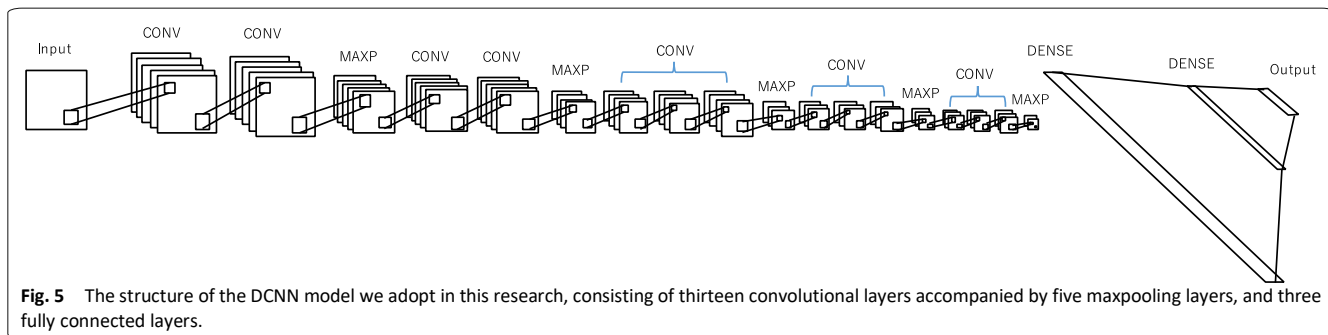
- scan (udp port 137): nmap -sU -p 137 [Network]
- scan (udp port 1900): nmap -sU -p 1900 [Network]

Network traffic from two network environments, LAN A and LAN B, was collected and used for generating feature maps. Here, LAN A is a network with a variable-length subnet mask with a length of 25 digits. And it is a network of the institute's critical infrastructures. On the other hand, LAN B is a network serving for general purposes in several labs, such as research and other daily operations.

We generated eight types of network events (including the normal state of a network) in LAN A and LAN B separately. Then we collected network traffic through the data collector, and generated feature maps of network events using the approach discussed above (Fig. 4). At last, we achieved 8120 feature maps in LAN A and 7125 feature maps in LAN B. Furthermore, we divided the datasets into a training set and a validation set with a ratio of 4:1 (Table 1).

Table 1 Datasets of traffic data's feature maps in LAN A and LAN B

| Event | Network A | | Network B | |
|----------------------|------------|-----------|------------|-----------|
| | Train Data | Test Data | Train Data | Test Data |
| normal | 576 | 144 | 1344 | 336 |
| scan (default) | 1040 | 260 | 496 | 124 |
| scan (tcp) | 560 | 140 | 1560 | 390 |
| scan (tcp port 23) | 1040 | 260 | 1600 | 400 |
| scan (tcp port 80) | 1040 | 260 | 1240 | 310 |
| scan (udp) | 652 | 163 | 992 | 248 |
| scan (udp port 137) | 1160 | 190 | 1184 | 296 |
| scan (udp port 1900) | 1160 | 190 | 1272 | 318 |



3.3 Network events classification using DCNN

Convolutional neural network (CNN) is one type of deep learning, with a characteristic of movement invariance with respect to the input of time-related data and is usually used to solve problems related to images such as multi-label image classification. A CNN model usually includes several convolutional layers accompanied with pooling layers in some cases and several fully connected layers at last. By using a kernel in each convolutional layer and the pooling layer in some of the layers, we can compress the information in the input data. It is thought that the information contained in the image can be expressed by combining these layers. Finally, in order to get the outcome as one-dimension information to solve multi-label classification, fully connected layers are combined in the CNN model and thus the output can be narrowed to a specific range. As a result, the input data here are the generated feature maps and the labels are the corresponding types of network events with respect to these feature maps.

We designed and built a DCNN model based on VGG-16 [11], which consists of thirteen convolution layers accompanied by five maxpooling layers and three fully connected layers, with an output of eight values (Fig. 5). By adopting this DCNN model, the hidden features of network events inside a feature map can be extracted. Then three fully connected layers are used to flatten the output matrix of the convolution layers and maxpooling layers to a one-dimension array and compress information into a matrix, giving out confidences of these eight types' network events.

Moreover, an activation function of the ReLU is used, which is defined as (3). After the computation of each layer, we use this function to convert the output to a non-linear distribution. And at the last layer, for this multi-label problem, we use an activation function called the softmax, which is defined as (4). Through using the softmax, each component including negative, greater than one, or might not sum to 1, will be in the interval (0, 1), with a sum of 1.

$$f(x) = \max(0, x) \quad (3)$$

Where x is the input data, $f(x)$ is the output data of a node, and the "max" is a function used to get a maximum between 0 and x .

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}} \quad (4)$$

Where x_i represents each element of the input vector x . And the softmax is used to normalize these values through dividing by the sum of all the exponentials of these elements.

The progress of training a deep learning model is basically an optimization problem, and a suitable learning function for updating weights of the model at every epoch is extremely significant for achieving a result of the classification. Since these considerations, two learning functions, the RMSprop and the Adam, are adopted individually to implement the training thus comparing the training results of them.

The RMSProp has a property that the emphasis is placed on the latest gradient information more than the past gradient information and gradually the past gradient information is forgotten, instead, the new gradient information is greatly reflected. Therefore, it has a great adaptation to time-related data. This learning function is defined as (5) (6).

$$h_t = \rho * h_{t-1} + (1 - \rho) * \frac{\partial L}{\partial w_t} \odot \frac{\partial L}{\partial w_t} \quad (5)$$

$$W_{t+1} = W_t - \eta * \frac{1}{\sqrt{h_t + \epsilon}} \odot \frac{\partial L}{\partial w_t} \quad (6)$$

Where L is the result of training loss, W is weights of the node, η is the learning rate which is used to control the extent we update weights at each time, and ρ is the decay rate with a value of 0.9, which shows the extent of past gradient information impacting on the current updating.

The other learning function of the Adam, which is defined in the following (7), with decay values of 0.9 and 0.999, is adopted to train the DCNN model as well, and the result of it is used to compare with that of the RMSProp.

$$\begin{aligned}
 m_t &= \rho_1 * m_{t-1} + (1 - \rho_1) * \frac{\partial L}{\partial w_t} \\
 V_t &= \rho_2 * v_{t-1} + (1 - \rho_2) * \frac{\partial L}{\partial w_t} \odot \frac{\partial L}{\partial w_t} \hat{m}_t = \frac{m_t}{1 - \rho_1^t} \quad (7) \\
 \hat{v}_t &= \frac{v_t}{1 - \rho_2^t} \\
 W_{t+1} &= W_t - \eta * \frac{1}{\sqrt{\hat{v}_t + \epsilon}} \odot \hat{m}_t
 \end{aligned}$$

Where L is the training loss, W is weights of the node, η is the learning rate, ρ_1 and ρ_2 are the decay rates with values of 0.9 and 0.999 individually, and ϵ is used to prevent the denominator from being zero.

Then we train this DCNN model from scratch with initial small random weights with values in an interval of [0, 1.0). Moreover, we adopt the mini-batch training, dividing the input data of feature maps into several small batches with a size of 40 for training, by which we stabilize the training progress as well as reducing the training cost. A learning rate (η) with a value of 0.00002 is used for updating weights at each epoch. And an early stopping is used to prevent the overfitting, a situation that validation accuracy keeps descending while training accuracy keeps increasing, by monitoring validation loss scores within recent five epochs.

We show the graph of validation loss with the RMSProp in LAN A as an example of the training progress with the early stopping (Fig. 6), where through updating weights at each epoch, the validation loss descends gradually, and at last, it terminates since the last five epochs' validation loss values don't achieve a better result through the updating.

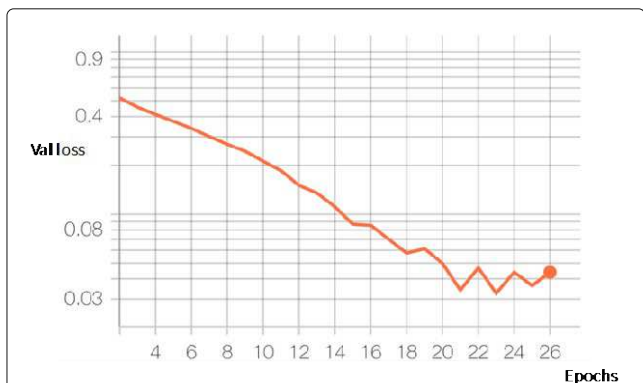


Fig. 6 The validation loss at each epoch with the RMSProp in LAN A, using the method of the early stopping: It shows a stable learning progress, where the loss descends gradually and stabilizes at the last several epochs.

After training, the model is supposed to have the ability to identify eight types of network events in LAN based on the confidence scores of model prediction. We trained our model using the datasets from two LAN environments, the LAN A and the LAN B, and adopted two types of learning functions, the RMSProp and the Adam as a comparison. As a result, the corresponding training graphs of the two LAN environments using two different learning functions are shown below (Fig. 7).

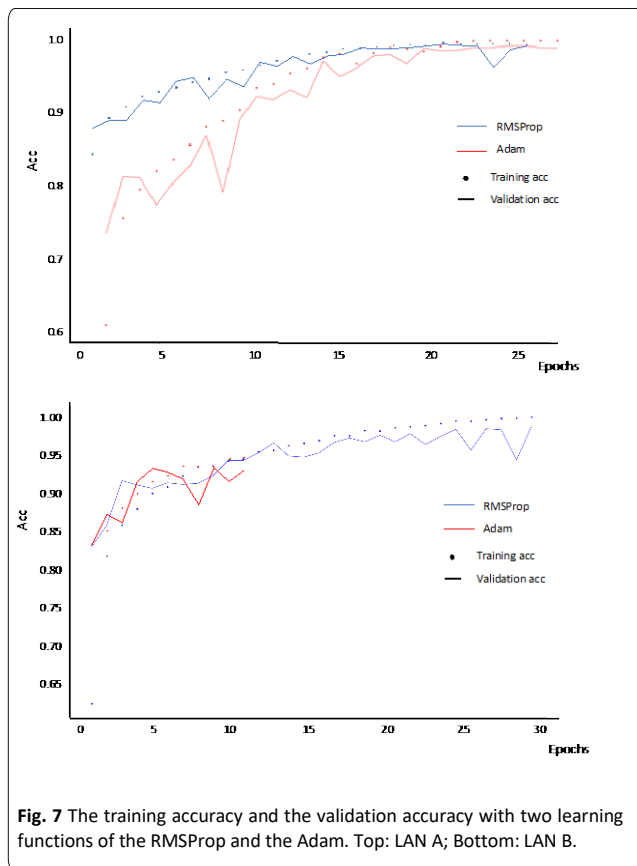


Fig. 7 The training accuracy and the validation accuracy with two learning functions of the RMSProp and the Adam. Top: LAN A; Bottom: LAN B.

From the graphs, we can see that in LAN A, both two learning functions achieve a progress with more than 25 epochs and a validation accuracy of around 0.98 at last, even though the Adam seems to outperform the RMSProp at first. On the other hand, in LAN B, the Adam completes the training progress with only 15 epochs while the RMSProp completes that with 30 epochs in total. However, the final result of the RMSProp is obviously better than that of the Adam, with both a better training accuracy and a better validation accuracy. And it also has a more stable training progress compared with the Adam. Hence, based on these comparisons, we choose the RMSProp in this research and further evaluate the performance of the scheme in the multi-label network events classification problem.

Table 2 Performance evaluation on the validation set (RMSProp learning function)

| Event | Network A | | | Network B | | |
|-----------------------|-----------|--------|-----------|-----------|--------|-----------|
| | Precision | Recall | F-measure | Precision | Recall | F-measure |
| normal | 0.931 | 1.000 | 0.964 | 0.994 | 1.000 | 0.997 |
| scan (default) | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| scan (tcp) | 1.000 | 1.000 | 1.000 | 0.994 | 1.000 | 0.997 |
| scan (tcp port 23) | 0.988 | 1.000 | 0.994 | 1.000 | 0.996 | 0.998 |
| scan (tcp port 80) | 0.993 | 0.989 | 0.991 | 1.000 | 1.000 | 1.000 |
| scan (udp) | 0.951 | 0.885 | 0.916 | 1.000 | 0.995 | 0.998 |
| scan (udp port 137) | 0.969 | 1.000 | 0.984 | 0.906 | 0.938 | 0.921 |
| scan (udp port 1900) | 1.000 | 0.977 | 0.988 | 0.913 | 0.861 | 0.886 |
| <i>Macro averages</i> | 0.982 | 0.981 | 0.982 | 0.976 | 0.974 | 0.975 |
| <i>Micro averages</i> | 0.981 | 0.972 | 0.976 | 0.971 | 0.968 | 0.965 |

Furthermore, by visualizing the compressed representation of different network events' feature map at the last fully connected layer of the DCNN model, we obtained compressed feature maps of these eight types of network events (Fig. 8).

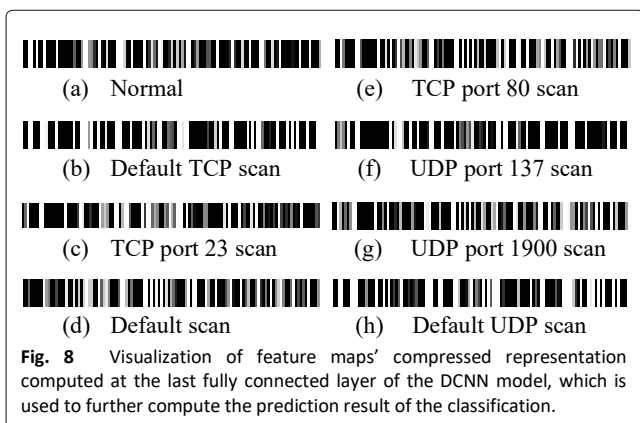


Fig. 8 Visualization of feature maps' compressed representation computed at the last fully connected layer of the DCNN model, which is used to further compute the prediction result of the classification.

4 Results

We evaluate the scheme in two active networks, LAN A and LAN B, each of which includes the event generating and data collecting system we discussed before. We evaluate the performance of it using the precision, recall, and F-measure.

The precision is a parameter used to show how many events are successfully classified in all test data; the recall is a parameter that is used to show how many times a specific event is successfully classified in all test data of that event. They are defined as (8) and (9) individually.

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

$$Recall = \frac{TP}{TP+FN} \quad (9)$$

Where TP (True Positives) indicates the number of events successfully detected, and FN (False Negative) represents the number of events unsuccessfully classified.

And the F-measure defined as (10) is a parameter used to show the comprehensive evaluation of a model's performance.

$$F\text{-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (10)$$

Here, we use the macro averages of each evaluation standard as the result, which take the average of each class's metric, thus treating all classes equally. Moreover, considering the existing imbalance in the datasets (e.g. fewer examples of tcp scan than the other classes in LAN A), the micro average method is also adopted to further evaluate the scheme, which aggregates the contributions of all classes to compute the average metric instead.

While training, we adopted the evaluating methods above at each epoch. We computed the precision, recall and F-measure scores of each event as a measure of the classification ability of the scheme. The corresponding result is shown above (Table 2). From the result, we can see that the classification between normal, arp scan, tcp scan and udp scan shows relatively great performance. Whereas, the classification between scans to some specific ports such as tcp port 80 and udp port 137 has relatively low F-measure scores, which means that it is more difficult to classify between scans to these specific ports than between only

normal and abnormal in LAN. Moreover, at last, we achieved a macro-F-measure score of 0.982 in LAN A and that of 0.975 in LAN B. We achieved a micro-F-measure score of 0.976 in LAN A and that of 0.965 in LAN B.

5 Discussion

The visualization of network traffic data allows some explainability to anomaly detection and classification in LAN. And a DCNN model is adopted to classify these reoccurring patterns in feature maps of various network events. Moreover, the experiments under two different networks are conducted to evaluate the scheme.

On the other hand, it is still possible that an adversary could forge these features inside a feature map by such as adjusting communication frequencies. Therefore, a more delicate experiment in a real-world setting needs to be verified. Furthermore, besides the proposed eight types of network events, the influence of additional, non-explicit network events on the classification result should be considered.

6 Conclusion

In this research, we are aimed to visualize traffic data in LAN by generating feature maps thus classifying these different network events with a DCNN model. We adopt nine types of protocol information as the discriminators for feature representation in feature maps. Then we evaluated the scheme using the recall, the precision and the F-measure. At last, we achieved macro-F-measure scores of 0.982 and 0.975, and micro-F-measure scores of 0.976 and 0.965 separately in two different LAN environments, as the comprehensive evaluation of the scheme's performance in network events classification.

Abbreviations

Not applicable

Declarations

Availability of data and materials

The datasets used for the evaluation of the algorithm are available online at <https://github.com/yuweisunn/LANSecurity>

Competing interests

The authors declare that they have no competing interests

Funding

The work has been funded by the graduate school of Information Science and Technology, the University of Tokyo.

Authors' contributions

Y. S. developed and implemented the core concepts of the algorithm presented within this manuscript, H. O. provided refinements and performed data acquisition and generation as well as further supplemental programming, H. E. provided further technical knowledge and support. All authors read and approved the final manuscript.

Acknowledgements

Not applicable

Authors' information

Yuwei Sun is a PhD's student in the Department of Information Science and Technology at the University of Tokyo. His research interests lie in intelligent systems and artificial intelligence. He received B.E. in Computer Engineering in 2018 from North China Electric Power University. In 2019, he was involved in cybersecurity research at the United Nations University. He is also a fellow of the advanced study program at the Massachusetts Institute of Technology.

Hideya Ochiai is an associate professor of the University of Tokyo, Japan. He received B.E. in 2006, M.E. in 2008, and Ph.D. in 2011 from the same university. His research interests have been sensor networking, delay tolerant networking, and building automation systems, IoT protocols, and cyber-security. He involves in the standardization of facility information access protocol in IEEE1888, ISO/IEC and ASHRAE.

Hiroshi Esaki received Ph.D. from University of Tokyo, Japan, in 1998. In 1987, he joined Research and Development Center, Toshiba Corporation. From 1990 to 1991, he has been at Applied Research Laboratory of Bell-core Inc., New Jersey, as a residential researcher. From 1994 to 1996, he has been at Center for Telecommunication Research of Columbia University in New York. From 1998, he has served as a professor at the University of Tokyo, and as a board member of WIDE Project. Currently, he is the executive director of IPv6 promotion council, vice president of JPNIC, IPv6 Forum Fellow, and director of WIDE Project.

References

1. J. P. Anderson, Computer Security Threat Monitoring and Surveillance, Technical Report, National Institute of Standards and Technology, 1980.
2. Nilamadhab Mishra, Sarojananda Mishra, Support Vector Machine Used in Network Intrusion Detection, National Workshop on Internet of Things (IoT), 2018
3. T. Omrani, A. Dallali, B. C. Rhaimi and J. Fattahi, Fusion of ANN and SVM classifiers for network attack detection, 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, 2017
4. J. Yang, J. Deng, S. Li, Y. Hao, Improved traffic detection with support vector machine based on restricted boltzmann machine, *Soft Computing* 21(11) (2017) 3101–3112, 2017
5. Asmaa Elsaedy, Kumudu S. Munasinghe, Dharmendra Sharma, Abbas Jamalipour, Intrusion detection in smart cities using Restricted Boltzmann Machines, *Journal of Network and Computer Applications*, 135 76–83, 2019.
6. M.A. Salama, H.F. Eid, R.A. Ramadan, A. Darwish, A.E. Hassanien, Hybrid intelligent intrusion detection scheme, *Soft Computing in Industrial Applications*, pp.293–303, 2011
7. Dhanabal L., S. P. Shantharajah, A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms, 2015
8. J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, *Proceedings of the 10th*

International Conference on Malicious and Unwanted Software (MALWARE), 2015

9. M. Yousefi-Azar, V. Varadharajan, L. Hamey, U. Tupakula, Autoencoder-based feature learning for cyber security applications, Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), 2017
10. Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, Mohsen Guizani, Deep Learning for IoT Big Data and Streaming Analytics: A Survey, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, 2018.
11. Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition, ICLR 2015.
12. Yann LECun, Patrick Haffner, Leon Bottou, and Yoshua Bengio, Object Recognition with Gradient-Based Learning, Shape, Contour and Grouping in Computer Vision, p.319, 1999.
13. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, Francis Bach, The MIT Press, 2016.
14. Eric Krokos, Alexander Rowden, Kirsten Whitley, and Amitabh Warshney, "Visual Analytics for Root DNS Data" IEEE, 2018.
15. Y.LeCun, K.Kavukcuoglu, and C.Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. IEEE, 2010.
16. Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Detection and Classification of Network Events in LAN Using CNN. IEEE International Conference on Information Technology (InCIT), 2019.

Figures

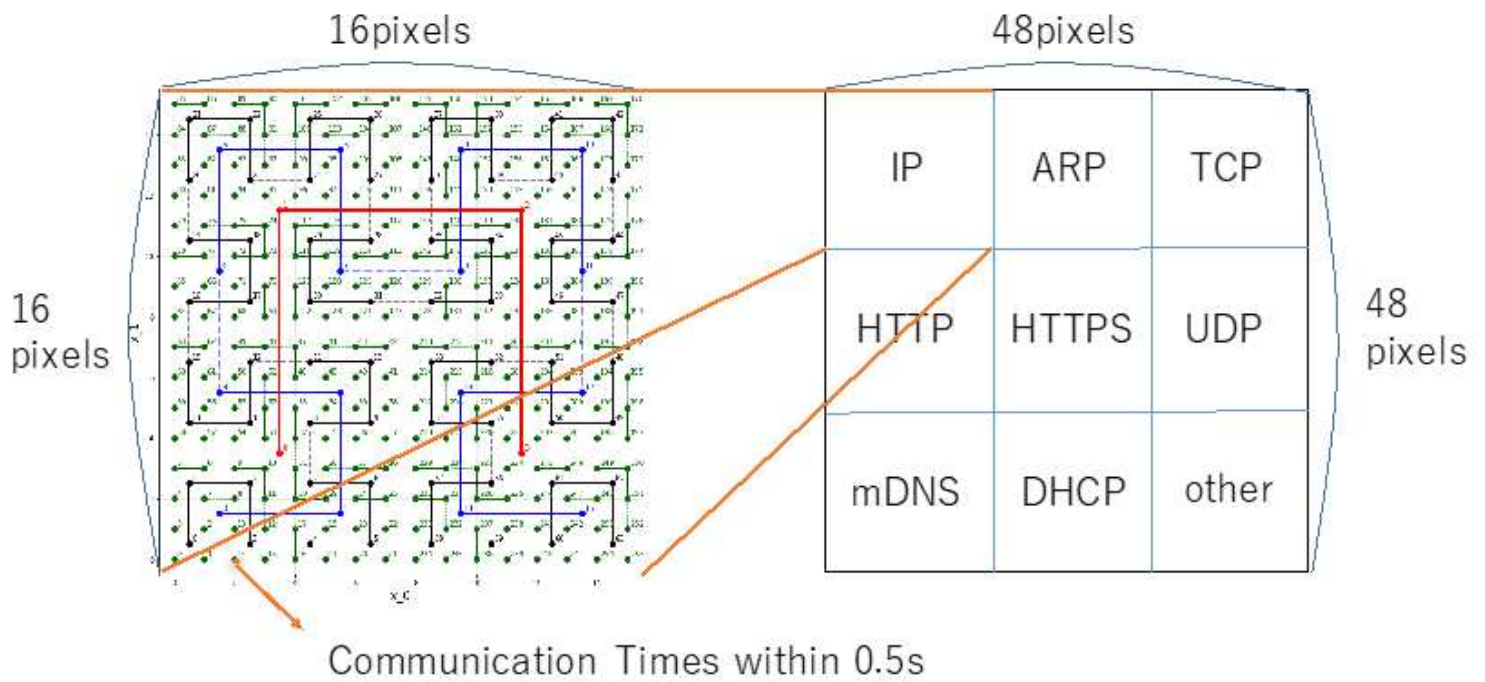


Figure 1

We adopt eight types of protocols and project visual information of each protocol into corresponding regions in an image with a width and height of 48 pixels.

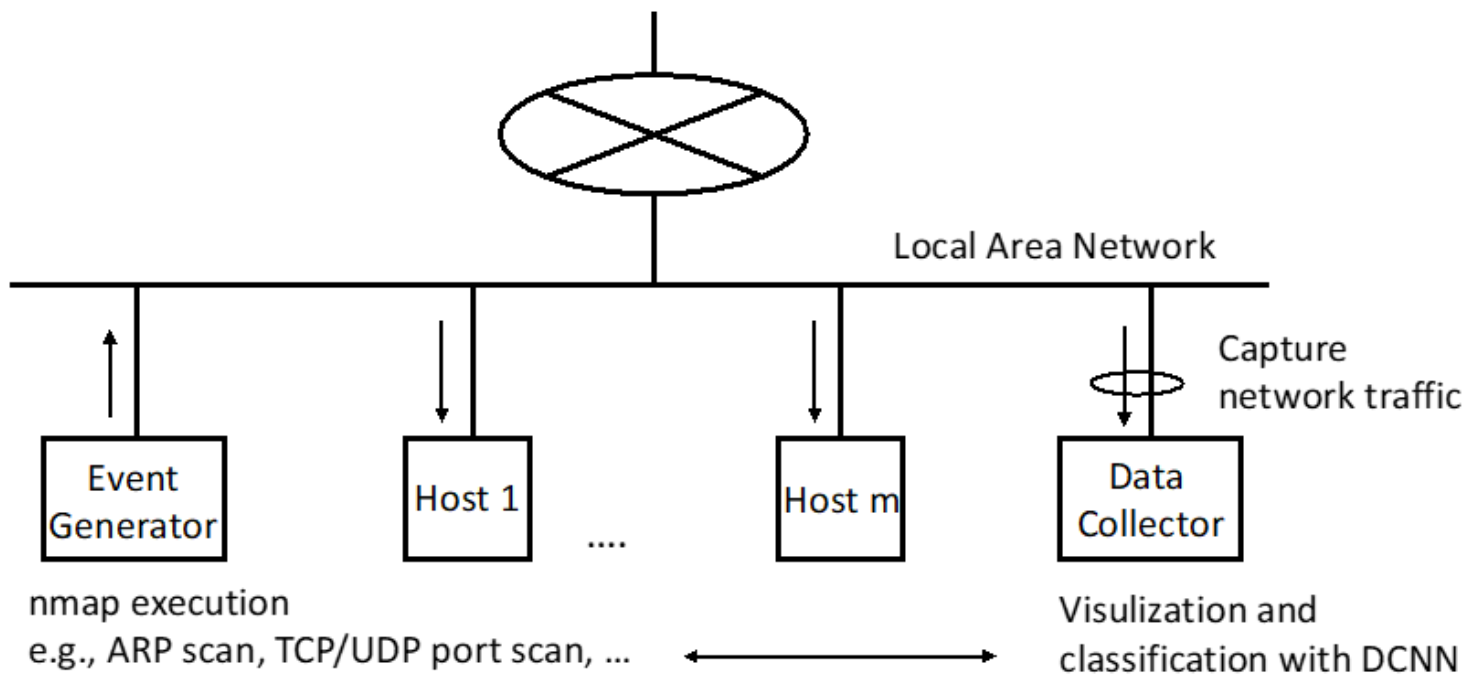


Figure 2

The events generating and data collecting system in the LAN.

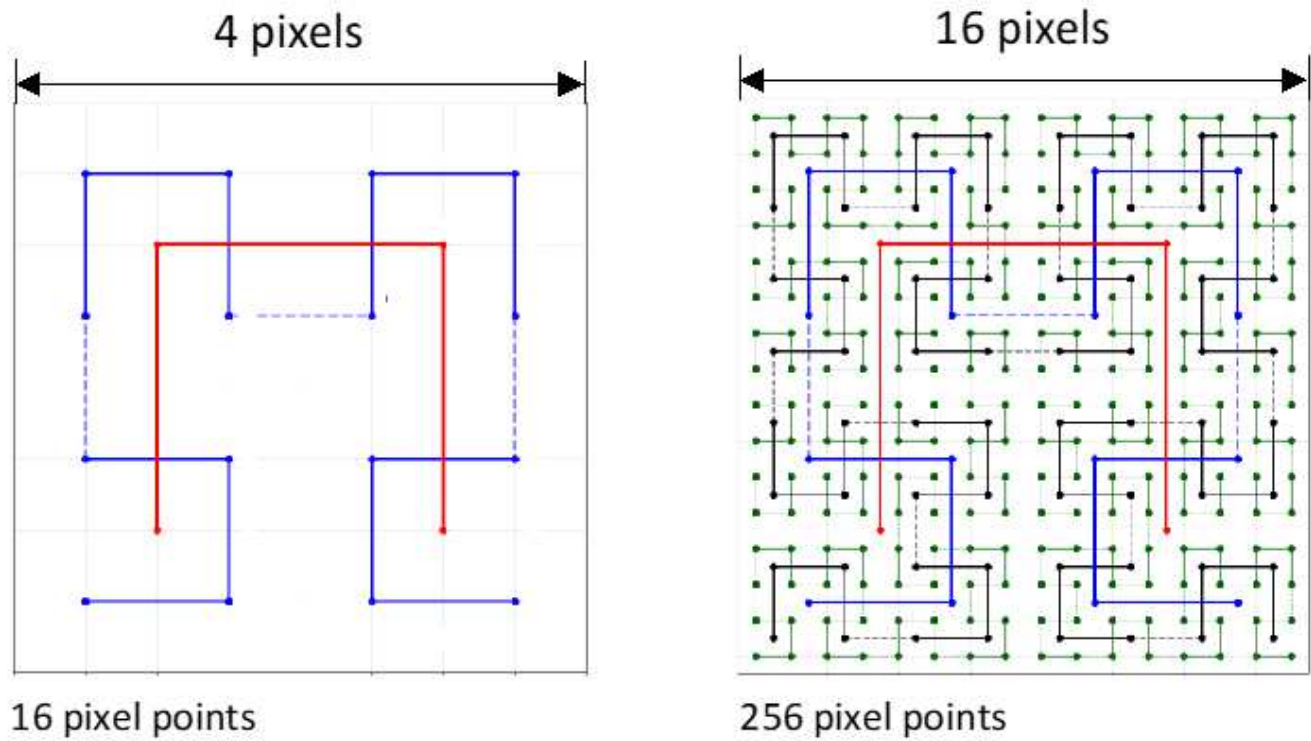


Figure 3

The geometric structure of Hilbert curve, where frequency information is projected into each position of it through pixel values. Every four pixel points in the same neighborhood can be represented with a central point. Hence, we are able to compress big data of network traffic with the structure of the Hilbert curve.

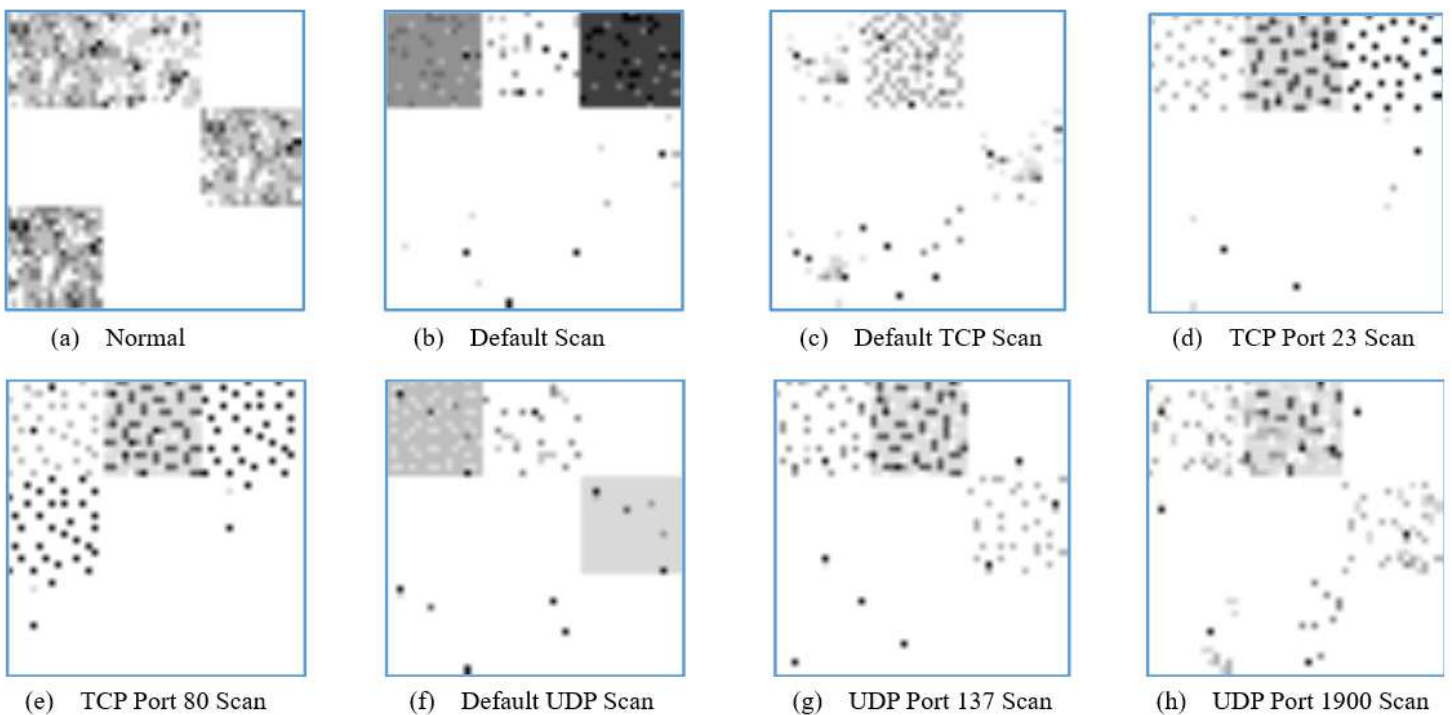


Figure 4

Line 1, from left to right: Feature maps of events of the normal, default scan, default TCP scan, specific scan of TCP port 23; Line 2, from left to right: Feature maps of events of the specific scan of TCP port 80, default UDP scan, specific scan of UDP port 137, specific scan of UDP port 1900. (when the fineness is 0.5 and the size is 16)

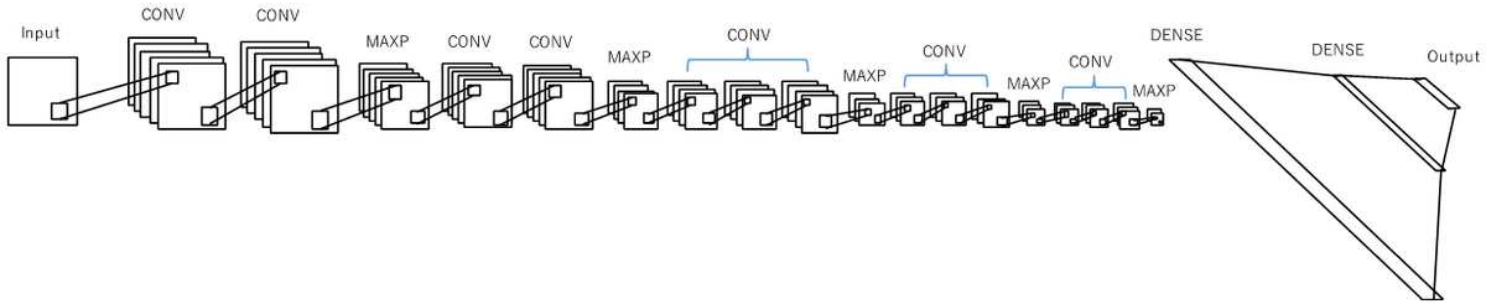


Figure 5

The structure of the DCNN model we adopt in this research, consisting of thirteen convolutional layers accompanied by five maxpooling layers, and three fully connected layers.

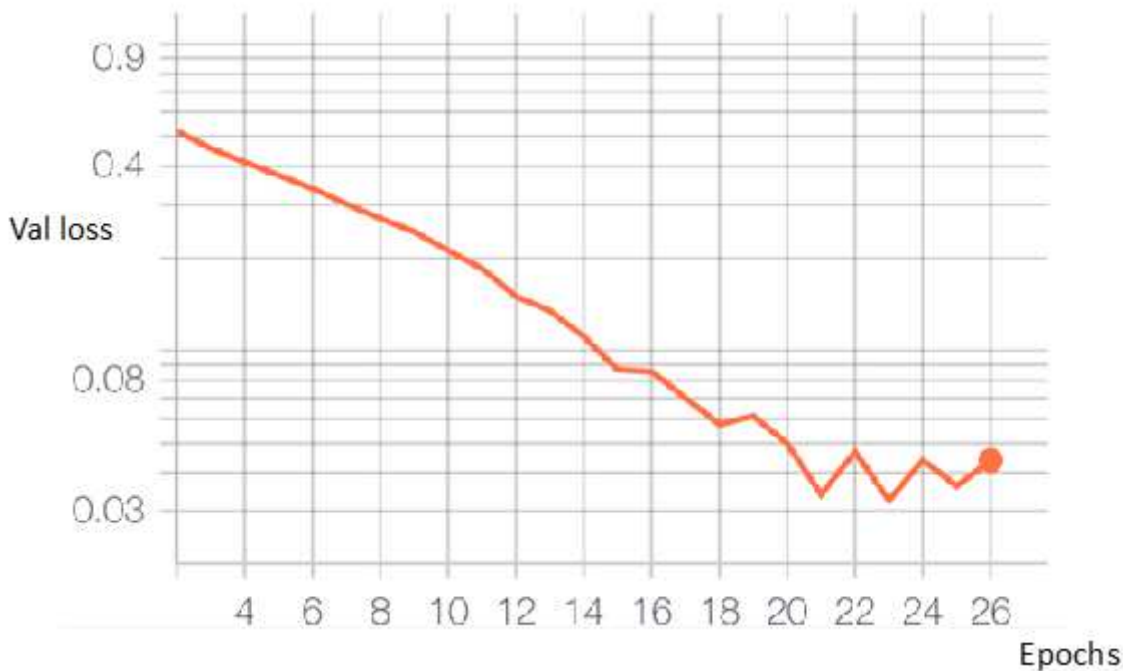


Figure 6

The validation loss at each epoch with the RMSProp in LAN A, using the method of the early stopping: It shows a stable learning progress, where the loss descends gradually and stabilizes at the last several epochs.

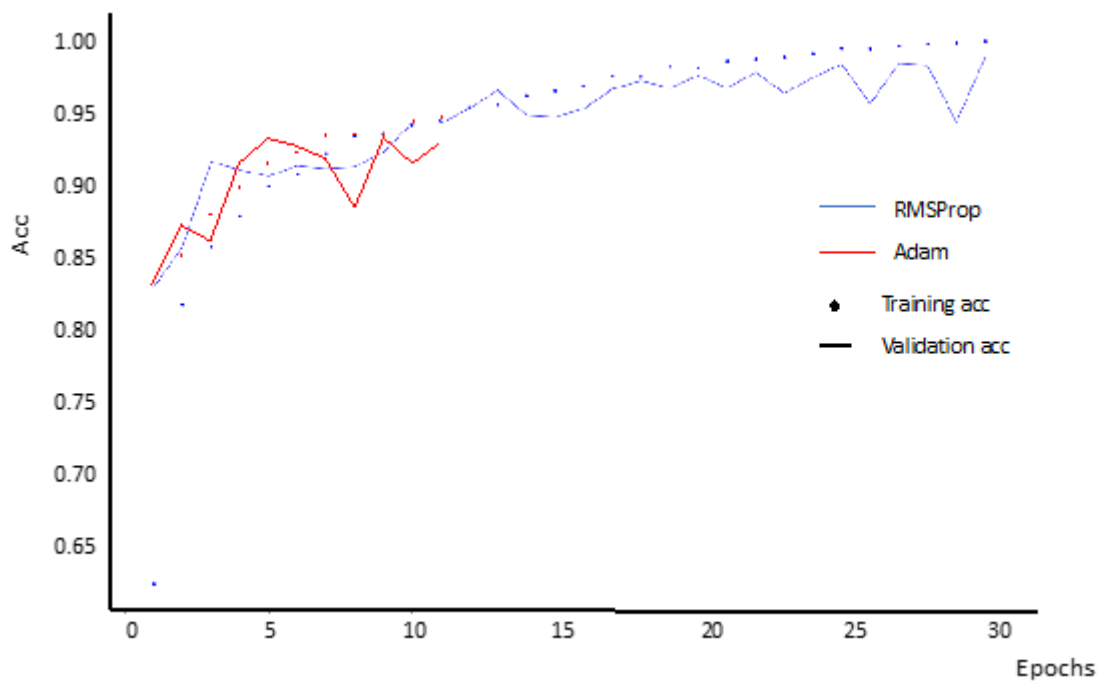
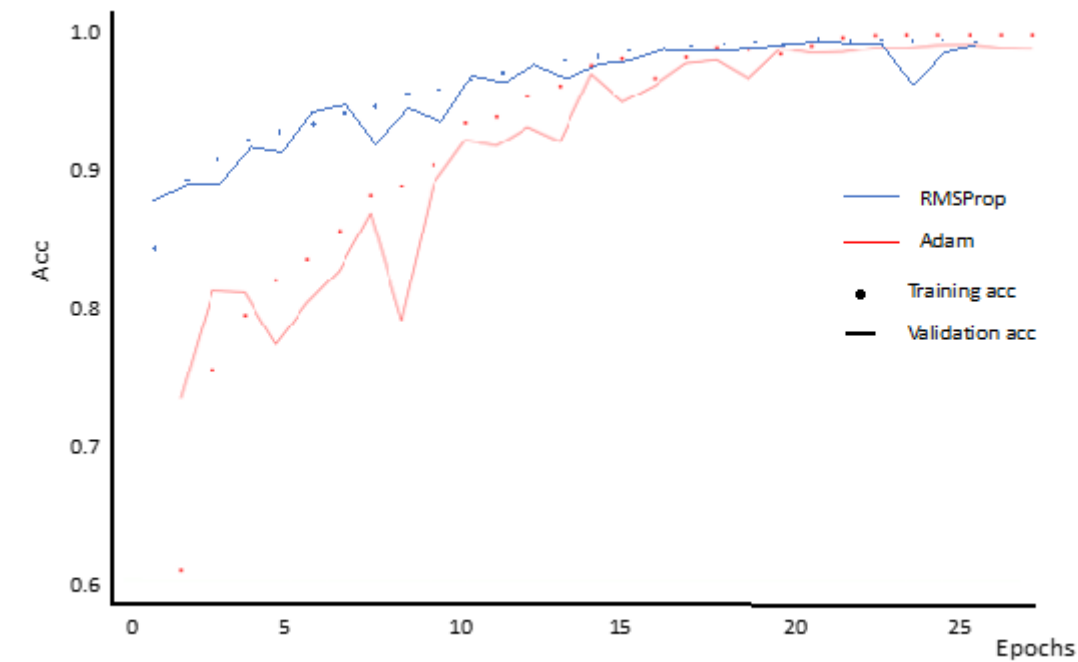


Figure 7

The training accuracy and the validation accuracy with two learning functions of the RMSProp and the Adam. Top: LAN A; Bottom: LAN B.

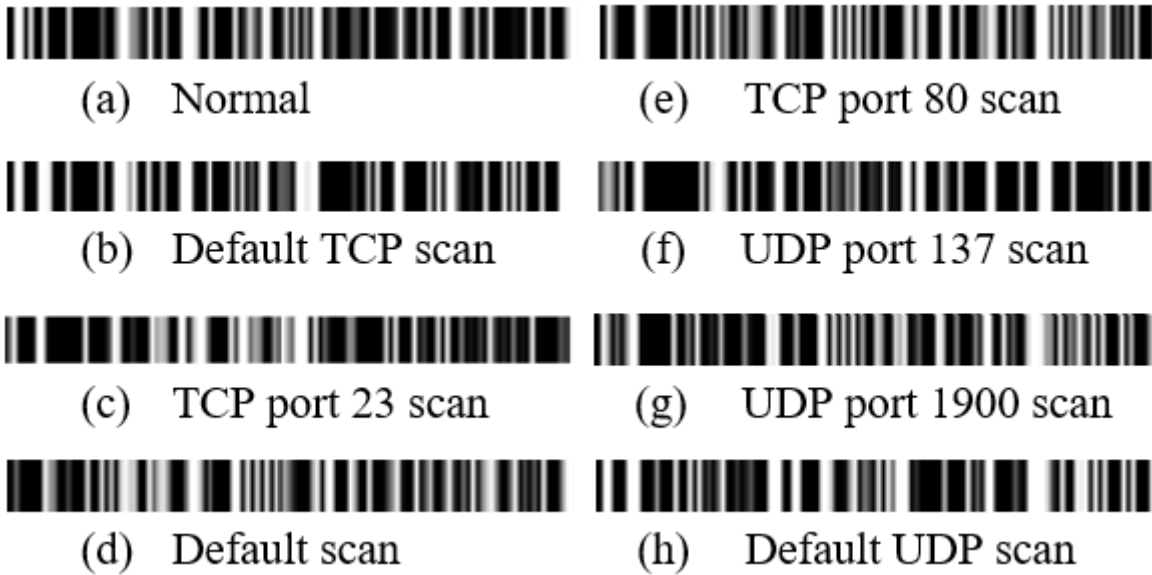


Figure 8

Visualization of feature maps' compressed representation computed at the last fully connected layer of the DCNN model, which is used to further compute the prediction result of the classification.