IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON EMERGING APPROACHES TO CYBER SECURITY

Received August 4, 2020, accepted September 1, 2020, date of publication September 8, 2020, date of current version September 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022722

A Malware Detection Method of Code Texture Visualization Based on an Improved Faster RCNN Combining Transfer Learning

YUNTAO ZHAO[1], WENJIE CUI[1], SHENGNAN GENG[2], BO BO[1], YONGXIN FENG[3], AND WENBO ZHANG[3]

[1]School of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China
[2]Beijing Institute of Astronautic Systems Engineering, Beijing 100000, China
[3]Graduate School, Shenyang Ligong University, Shenyang 110159, China

Corresponding author: Yongxin Feng (fengyongxin@263.net)

This work was supported in part by the China Postdoctoral Science Foundation under Grant 2016M590234, in part by the Department of Education of Liaoning Province under Grant LG201908, in part by the Department of Science and Technology of Liaoning Province under Grant 20180551066, in part by the Distinguished Professor of Liaoning Province, and in part by the Postdoctoral Foundation of Shenyang Ligong University.

ABSTRACT Today, with the continuous promotion and development of IoT and 5G technology, Cyberspace has become an important pillar of economic and social development, and also a foundational domain of national security. Cyberspace security is attracting more and more attention. Therefore, detecting malware and its variants is of great significance to Cyberspace. However, the increasing sophistication of malicious variants, such as encryption, polymorphism and obfuscation, makes it more difficult to identified malware effectively. In this article, a malware detection method of code texture visualization based on an improved Faster RCNN (Region-Convolutional Neural Networks) combining transfer learning is proposed. We utilize visualization technology to map malicious code into corresponding images with typical texture features, and realize the classification of malware. Firstly, in order to quickly acquire and locate the representative texture of malware, we adopt CNN to extract the global and deeper features of malicious code images. Then with RPN (Region Proposal Network) we generate the target image frame, which is used to locate the core texture of malware file (.text file), to realize the accurate positioning of malicious features. Secondly, we preprocess and train Faster RCNN model with ImageNet set, and then transfer the model to the malware classification model to accelerate the convergence of the first model and promote generation performance. Thirdly, we construct an improved objective function in which a novel multi-label of classification proportion is added to solve the problem that the texture change of ''.text'' section and other sections in malicious code image is not obvious after transfer learning. We collect code samples of six malware families from Kaggle platform, and compared the experimental results before and after transfer. The results show that the novel method can accelerate the convergence of loss function, and obtain higher accuracy (92.8%), lower FPR (6.8%) and better P-R (precision-recall) curve.

INDEX TERMS Cyberspace security, faster RCNN, malware detection, code classification, transfer model.

I. INTRODUCTION

In recent years, with the development of Internet of Things and artificial intelligent technology, there has been more attack methods for Cyberspace such as automatic malware generation tools, obfuscation and polymorphism technology. The emergence of the new tools and technology allow attackers to do more damage at a lower cost. The number of malicious code grows exponentially every year. According to the 2018 ''State of the Internet Center briefing'' report, there were 222,000 hosts infected with malicious code within one week [1]. In 2018, WannaCry, Saturn, and other ransomware, and their variants attacked the Internet, causing great losses to Internet users [2]. Malware or malicious code is a piece of code that is intentionally compiled or set up to steal privacy, obtain information, extort money and destroy the

The associate editor coordinating the review of this manuscript and approving it for publication was Luis Javier Garcia Villalba.

system, such as computer virus, Trojan, worms, backdoors, logic bombs [3] etc. researchers have done a lot of studies on the analysis and detection methods of malicious code. Traditional commercial antivirus products typically rely on a signature-based approach [4] that requires a local signature database to store patterns extracted from malware. However, the minor changes of malicious code can lead to failure of the signature-based approach. Increasingly, malicious code can easily evade signature-based detection by encrypting, obfuscating, or packaging. Hence such approach, which uses a countable signature-based codes against infinitely growing variants, is rather limited. In order to solve the detection problem of malware being continuously recompiled, changed, packaged and disguised, a lot of studies are required to keep up with the ever-changing malware.

Deep learning, the core drive of AI (artificial intelligence), is a kind of representation learning model based on deep neural network, which adopts layer by layer abstract mechanism from low-level data objects to high-level feature representation. Through building a complex neural network to simulate the analysis process of human brain, the characteristics of low-level localization are constantly summarized and abstracted into the higher-level neural network layer to finally get a highly generalized global information representation, which is an extension of the artificial neural network method. Based on deep learning technology, we can deal with various complex data types and learn from the original data to high-level feature expression adaptively. Compared with the traditional methods, the processing of massive data with deep learning has great advantages. Now many kinds of deep learning models, such as deep confidence network, convolutional neural network and recurrent neural network, have been applied to computer vision, voice recognition, natural language processing, bioinformatics and other fields, and have achieved great success. It can be said that deep learning technology has led to a scientific and technological revolution beyond the field of computer science itself. As an efficient algorithm, it has entered many research fields.

In this article, a malware detection method of code texture visualization based on an improved Faster RCNN combining transfer learning is proposed. In order to fight against malware, we adopt code visualization to represent malicious behaviors. Because a tiny variation of malicious code will lead to the failure of signature-based methods, but it will not change the texture visualization features essentially. Therefore, the detection method based on texture features can effectively combat code obfuscation. We adopt CNN to extract deeper features of malware texture visualization and use RPN to locate the core features of visualization texture. Then we transfer Faster RCNN model to the malware classification model to accelerate the convergence. At the same time we propose an improved objective function to solve the image blurring and overfitting after transfer learning. We adopt the dataset from Kaggle of Microsoft Malware Classification Challenge (BIG 2015), which contains enough malicious samples and their variants from different families.

The main contributions of this article are summarized here.

1) We adopt Faster RCNN to acquire and locate the representative texture of malware, the RPN of which can realize the accurate positioning of malicious features. Compared with LBP algorithm and Gist algorithm, the Faster RCNN can extract the global and deeper features of malicious code images. The results show that the method can obtain higher accuracy (91.4%), lower FPR (9.8%).

2) Transfer learning is used to preprocess and train Faster RCNN model with ImageNet set. We transfer the model to the malware classification model to accelerate the convergence of the first model and promote generation performance. The results show that the transfer learning method can acquire higher accuracy (92.8%), lower FPR (6.8%) and better P-R (precision-recall) curve.

3) We reconstruct an improved objective function in which a novel multi-label of classification proportion is added to solve the problem that the texture change of ''.text'' section and other sections in malicious code image is not obvious after transfer learning.

The rest of the paper is structured as follows. Section 2 presents the related work. Section 3 describes our detection method and introduces our proposed deep learning framework for malware detection. Section 4 evaluates the performance of our proposed method in comparison with other alternative methods. Finally, Section 5 concludes.

II. RELATED WORK

In recent years, deep learning has been used to detect and classify malware families of unknown samples due to its expansibility, rapidity, and flexibility. Saxe and Berlin [5] introduced a deep learning based malware detection approach that achieves a true positive rate of 95% and a false positive rate of 0.1%. They calculated the entropy histogram of binary files, and extracted the number of calls to the context byte data, and collected the metadata imported by the execution class and DLL. They also converted the four types of features into 256-dimensional vectors and used the eigenvectors in the four-layer neural networks to classify unknown samples. Kolter and Maloof [6] achieved good results by using n-gram instead of non-overlapping byte sequence features for data mining. The experimental results showed that the optimal decision can be obtained by using the boost decision tree.

With the development of computer vision technology, there are some new ideas about malware detection. Nataraj *et al.* [7] proposed a visual malware classification method through image processing. They converted malware binary data into grayscale images, used KNN model to classify, and calculated Euclidean distance. The experimental results showed that this method is a fast detection method. But because they used global image features, attackers can still escape detection by local transformation. Therefore, in the subsequent paper

[8], they compared the methods based on image feature and dynamic analysis methods. The experimental results showed that the image methods are effective and extensible, whose accuracy is comparable to that of dynamic analysis. They also found that the improved approach can handle both the packaged malware and the unpackaged. Zhang and Peng [9] put forward a malicious code classification method based on feature fusion. The method extracted the local texture features of malicious code images and the operation code instructions of disassembly file. According to the fusion feature vector of malicious code, an RF (random forest) classification model is constructed to realize the malware classification. Upchurch and Zhou [10] proposed a method of code reuse to detect malware. This method extends n-gram analysis into target blocks, which are basic blocks extracted from compiled code or entire text portions. Luo *et al.* [11] proposed a stacked auto encoder. They combined the texture features of malicious code and the frequency of instruction sentences to train this stacked auto encoder and a softmax classifier. Liao and Liu [12] proposed a malicious code detection framework based on data mining and machine learning, used this method to extract the code features in the text structure, and achieved the detection of malicious code using multiple classification methods. Li *et al.* [13] described a method to detect malware through the use of permissions. They used mining permission data to identify the most important permissions, and eventually found 22 effective permissions. Using support vector machine classification, the detection accuracy of the malware reached more than 90%. Wuechner *et al.* [14] proposed a malware detection method that uses data compression mining on the data flow graph to improve accuracy. Kong and Yan [15] proposed an automatic classification framework for malware (a function call graph) based on unsupervised clustering learning of structured information. After extracting the fine-grained features for function call graphs, they calculated the similarity of malware and clustered the samples from homogeneous malware or the same malware family by the discriminative learning method of distance matrix. In 2018, Liu *et al.* [16] proposed a malicious code visualization method based on information density enhancement. The method can enhance the information density of malicious code image by visualizing the operation sequence information of the function section in the ".text" section of malicious code file, so as to improve the efficiency of malicious code visualization.

The above malware detection and classification methods have achieved good results, but most of these methods relied heavily on professional knowledge or expertise in malware. At the same time, with the ever-increasing and dynamic changes in malware, there is a huge price to pay for traditional methods in response to new malware. And most of methods belong to shallow learning. Therefore, in the paper we attempt to adopt deep neural network to automatically learn the characteristics of malicious code to reduce the manpower cost of malware expert. With visualization methods we can represent and extract the spatial structure characteristics of malicious code. And we use transfer learning to accelerate convergence and improve detection accuracy.

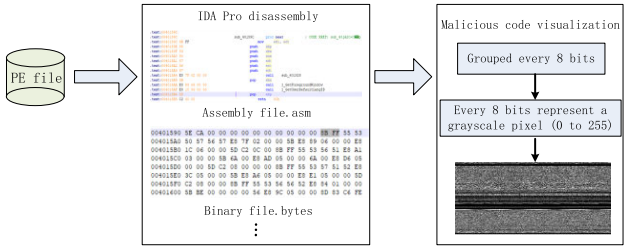


FIGURE 1. The schematic diagram of malicious code visualization.

III. DETECTION METHOD OF MALICIOUS CODE IMAGES

A. MALICIOUS CODE IMAGES

The concept of malicious code images was first proposed by Nataraj and Karthikeyan of the University of California in 2011 in their paper [7]. The idea is that the binary files are converted to grayscale images, and use the texture features in the images to cluster malicious codes.

Therefore, in order to generate a malicious code image, the malicious sample needs to be visualized. Firstly, we perform a static disassembly operation to get two source files of malicious code, i.e. the binary files (.bytes files) and assembly files (.asm files). Secondly, we take the binary executable files as input data, and convert them to the corresponding ".bytes" binary files by disassembly. And we convert every 4 binary digits into 1 hexadecimal digit, and combine the 2 hexadecimal digits (8 binary digits) that correspond to the 0-255 gray-scale value of image. Then we split the binary stream sequence every 8-bit and arrange them in order to form a corresponding gray image of the malicious code. The schematic diagram of malicious code generation is shown in Fig. 1.

The information entropy [17] of malicious code we usually use is to relate characteristics of each section of PE files. The PE file of the malicious code uses a flat address space, and the code and data are stored in different areas according to a certain format. The PE file includes a PE header and a series of sections. The file header contains metadata about the file itself. And each section after the header is actual parts of the file, which contains useful information [18]. Typically, the data of sections is logically related, and each section has a different name depending on its purpose, as shown in Table. 1.

The different sections of the malicious code PE file correspond to different texture features in the converted grayscale image [19]. These texture features reflect differences by the arrangement of pixel values. Nataraj L described the corresponding positions of the various sections after the PE file is converted into a malicious code image in the paper [7]. As shown in Fig. 2, the ".text" section contains the core feature Opcode and is located in the front of the grayscale image.

TABLE 1. Sections of the malicious code PE file.

Section name	Section content
.text	The .text section contains the CPU execution instructions, data of other sections supporting information. In general, this is the only section that can be executed and contains code.
.rdate	The .rdate section usually contains import and export function information, as well as other read-only information used by the program. (Contains read-only information that is globally accessible in the program).
.date	The .date section contains the global data of the program and can be accessed from anywhere globally. Local data is not stored in this section.
.idate	The .idate section sometimes shows and stores imported function information. If this section does not exist, the import function information is stored in the .rdate section.
.edate	The .edate section sometimes shows and stores exported function information. If this section does not exist, the export function information is stored in the .rdate section.
.pdate	The .pdate section exists only in 64-bit executables, storing exception handling information.
.rsrc	The .rsrc section contains the resources used by the executable, but these resources are not executable.
.reloc	The .reloc section contains information to relocate library files.

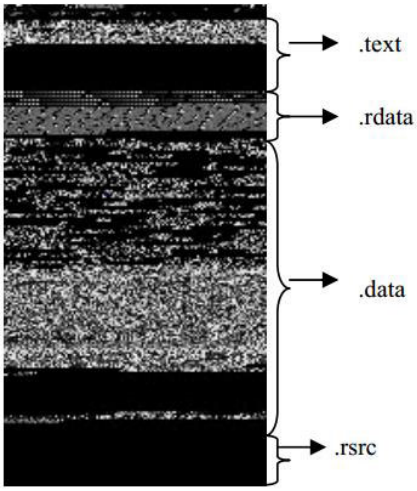


FIGURE 2. Individual sections of malicious code grayscale images.

B. FASTER RCNN NETWORK

Faster RCNN [20] is a deep learning network model based on the RPN (Region Proposal Network). The RPN is used to achieve target location.

Faster RCNN structure diagram shown in Fig. 3. The convolutional neural network was used to obtain the feature mapping of the image. The obtained feature mapping was inputted into the RPN network to generate target candidate regions (Proposals), and then inputting feature mapping and target candidate regions through a subsequent ROI network (Region of Interest Pooling), and finally obtaining the feature expression of each target candidate region of feature mapping. The Faster RCNN network supports inputting images of any size. We set the normalization scale of pictures before entering the Faster RCNN, such as M × N. Then we can input a P × Q size picture. If P × Q is greater than M × N, this picture will be rescaled, and if P × Q is less than M × N, the edges of this picture will be filled with 0.

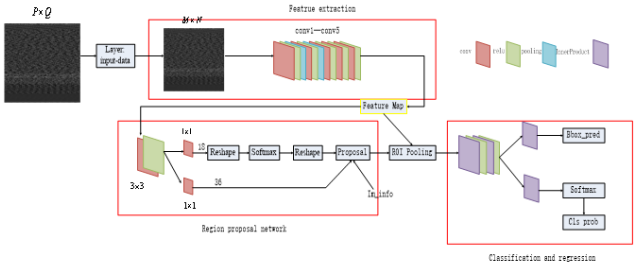


FIGURE 3. Faster RCNN structure diagram.

The "text" section containing the core feature code in the malicious code image is labeled as the target area. A malicious code image dataset is produced, and the malicious code feature is automatically extracted from the network to visualize the malware classification.

Because of the structural characteristics of the Faster RCNN network, its loss function is also a multi-tasking loss, that is, the sum of the prediction loss of the target frame and the regression loss of the target frame.

The objective function of an image in Faster RCNN is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{1}$$

p_i is the probability that the anchor is predicted to be target.

$$p_i^* = \begin{cases} 0 & negative\_lable \\ 1 & positive\_lable \end{cases} \tag{2}$$

$t_i = \{t_x, t_y, t_w, t_h\}$ is a vector, which represents the four parameter coordinates of the predict bounding box.

t_i^* is the coordinate vector of the ground truth bounding box corresponding to the positive anchor.

$L_{cls}(p_i, p_i^*)$ is a cross-entropy loss of binary classification (target & non-target).

$$L_{cls}(p_i, p_i^*) = -\log[p_i^* p_i + (1 - p_i^*)(1 - p_i)] \tag{3}$$

$L_{reg}(t_i, t_i^*)$ is the regression loss, calculated by $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$, R is the smooth L1 function.

$$smoothL1(x) = \begin{cases} 0.5x^2 \times {}^1\!/\!_{\sigma^2} & |x| < {}^1\!/\!_{\sigma^2} \\ |x| - 0.5 & otherwise \end{cases} \tag{4}$$

C. MALWARE DETECTION METHOD BASED ON TRANSFE LEARNING

1) TRANSFER LEARNING

Transfer learning is an important method of machine learning. The process of transfer learning refers to applying the knowledge, models or relationship structures learned in a certain field to another field that is related but different. The core idea of transfer is that experience gained in learning to perform one task can help improve learning performance in a related, but different, task [21]. For example, a trained images

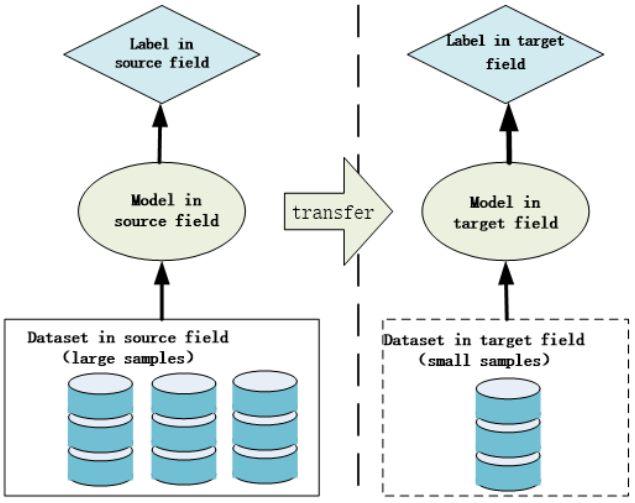


FIGURE 4. Schematic diagram of transfer learning.

classification network can be used for another task related images. The schematic diagram is shown in Fig. 4.

In theory, the more network parameters are, the higher the complexity of the model is, the greater the probability of over fitting phenomenon becomes. As the number of network layers increases, the error of forward propagation training will diverge, and the complexity of the time to calculate the error will also increase. To solve these problems, there are two transfer schemes for neural networks, feature extraction and fine-tuning. Feature extraction is to add a simple classifier to the pre-trained network on the source task, and take this network as a feature extractor. Fine-tuning allows modification of pre-trained network parameters for learning target. Fine-tuning usually freezes the underlying network parameters, and adjusts the high-level network parameters, and attaches the new initialization fully connected layer, and then uses a smaller learning rate to update the network parameters. Because the features learned from the underlying parameters of the network are general, freezing those parameters, which are transferred to the new model without change, can avoid a large amount of parameters updating and training, and can also reduce the occurrence of overfitting, and can also reduce the occurrence of overfitting.

2) AN IMPROVED TRANSFER LEARNING NETWORK

The Faster RCNN has a complex structure and requires a large dataset, for example ImageNet includes 14 million pictures. But the number of malicious samples collected is limited. Therefore, we adopt transfer learning of Faster RCNN that has been trained by a large dataset (ImageNet), and then train our image dataset of malicious code grayscale. The technical solution flow chart is shown in Fig. 5.

We transfer the pre-trained Faster RCNN model with ImageNet to a visualized malware classification model, which is a model-based transfer learning method. The pre-trained Faster RCNN is a multi-class detection network spending a lot of time and resources. Although the parameters of this network may not be optimal parameters for images in other fields

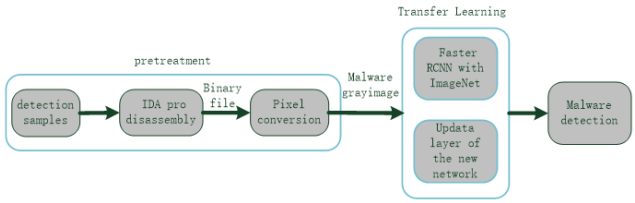


FIGURE 5. Diagram of detection method for malicious code based on transfer learning.

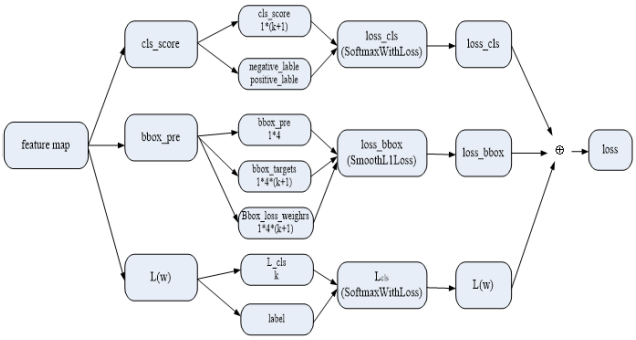


FIGURE 6. Schematic diagram of the total loss function.

and their weights may not identify pictures in other fields, they have strong feature extraction capabilities for images. The underlying parameters of the model can well extract the shallow features of the image, which is universal for most images. The transfer model from ImageNet can ensure excellent generalization of small samples in the malware detection field after transferring.

The objective function of Faster RCNN is a compound loss function, which consists of bounding box regression and classification. RPN selects the candidate region of suggestion box in the ''.text'' section. The ROI Pooling layer combines the malware feature mapping by the ZF convolutional neural network and the information of the candidate regions in the ''.text'' section by the RPN, then obtains the coordinates and scales of candidate regions on each feature mapping. In the process of back propagation and data update, the back-propagation error was widely regressed on the candidate suggestion box of the.text section, resulting in a relatively low classification proportion. For malware image samples, the texture change between the ''.text'' section and other sections is not very obvious, which will cause errors in the regression of the candidate suggestion box in the.text section with back propagation, which affects classification proportion and leads to an unsatisfactory performance. As a result, in order to improve the classification accuracy and classification proportion based on the original objective function, we define a novel objective function by adding a multi-label loss function, and use it to adjust network parameters during back propagation. The schematic diagram is shown in Fig. 6.

For the malicious code image samples $s_i = (x_i, y_i)$, x_i represents the input, and y_i represents the category label of the malicious code, and the novel objective function

formula is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_1, t_1^*) + L(w) \quad (5)$$

Among them,

$$L_{cls\_score}(p_i, p_i^*) = \frac{1}{N_{cls}}\sum_i -\log[p_i^* p_i + (1-p_i^*)(1-p_i)] \quad (6)$$

$$p_i^* = \begin{cases} 0 & negative\_lable \\ 1 & positive\_lable \end{cases} \quad (7)$$

p_i is the probability that the anchor is predicted to be target.

$$L_{bbox\_score}(t_i, t_i^*) = \lambda\frac{1}{N_{reg}}\sum_t p_i^* R(t_i - t_i^*) \quad (8)$$

t_i^* is the coordinate vector of the ground truth bounding box corresponding to the positive anchor.

$t_i = \{t_x, t_y, t_w, t_h\}$ is a vector, which represents the four parameter coordinates of the predict bounding box.

R is the smooth L1 function.

$$smoothL1(x) = \begin{cases} 0.5x^2 \times {}^1\!/\!_{\sigma^2} & |x| < {}^1\!/\!_{\sigma^2} \\ |x| - 0.5 & otherwise \end{cases} \quad (9)$$

$$L(w) = \sum_{m=1}^{M}\sum_{j\in S^m} \alpha_m l(x_i, y_i\,|w) \quad (10)$$

M is the different Branch. Each Branch weight is different, used α to represent. Then:

$$l(x_i, y_i\,|w) = L_{cls}(p(x_i), y_i) = -\frac{1}{N}\sum_i \log(soft\ \max(x_i)) \quad (11)$$

In this way, the proportion of classification is increased, thereby improving the accuracy of classification.

The specific implementation mainly includes the following steps.

1) ***Dataset production***: The collected malicious samples are subjected to static disassembly processing and converted into malicious code images, then are placed in the ''JPEG Images'' folder. Furthermore, the malicious code images are manually labeled, and the ''.text'' section containing the core feature code is used to generate the image. The xml file of the information is placed in ''Annotations'' folder. The training dataset and the test dataset data are saved in txt format and placed in the ''ImageSets / Main'' folder. The processed dataset is prepared for input layer of the model.
2) ***Model pre-training***: We adopt the ImageNet dataset for model pre-training, which can ensure a good generalization of small image sample data in new fields after transferring. And the underlying parameters have shallow feature representation capabilities and can extract images well on malware texture. The features are universal for most images.
3) ***Fine-tuning***: We adjusted appropriate weights of pre-trained model after many forward propagation and back propagation. In this way, we get a model that stores the feature data and weight information, which are closely related to the classification object. Some of these information have more common features that can be shared between different tasks or objects. For example, the low-level convolutional layer extracts shallow features, which are general and universal, so the weights of this part of the network can be shared between different tasks or objects. Furthermore, we fine tune the previous Faster RCNN network and retrain the model with the malware images dataset. The fine-tuning and retraining scheme is as the followings: First, modify the convolutional layer for feature extraction. It is important to choose an appropriate initial learning rate [22] for fine-tuning. Generally, the better range of learning rate is between 0.01 and 0.00001. The shallow features extracted by the lower-level convolutional layers (conv1 and conv2) are general, and the weights of transfer model have little influence on the final goal, so the parameters of this layer are frozen. The middle of convolutional layers (conv3 and conv4) extract deep features, and the weights affect the target, so we fine-tuned the learning rate to 0.001. The highest level features extracted by the highest convolutional layer (conv5) are only valid for specific data sets, concentrate on the target, and have no generalization ability for other types of images. Therefore, we fine-turned the learning rate to 0.0001. Finally, we adjusted the output dimension of the fully connected layer according to the number of categories.
4) ***Training network***: Train the network with a gradient descent (SGD) based on back propagation algorithm. After setting the parameters, we retrained the network model according to the new configuration file to obtain a classification model that is suitable for malware detection. The experimental flowchart is shown in Fig. 7.

IV. EXPERMENTAL RESULTS AND ANALYSIS

A. LAB ENVIRONMENT

The experiment is carried out under the Caffe framework of Linux system. Caffe [23] is a convolutional neural network framework based on C++/CUDA/Python developed by Berkeley Vision and Learning Center, which is suitable for feature extraction of 2D image data. It has fast training speed and provides a complete set of tools for model training, forecasting, fine-tuning, publishing, data pre-processing, and good automatic debugging. Experimental hardware and software environment is shown in Table. 2.

B. DATA SET

This article selected malicious samples provided by Microsoft Corporation [24], and collecting 6 common families of malicious code samples, a total of 1821, from the Windows platform. Besides, we expend double the malicious

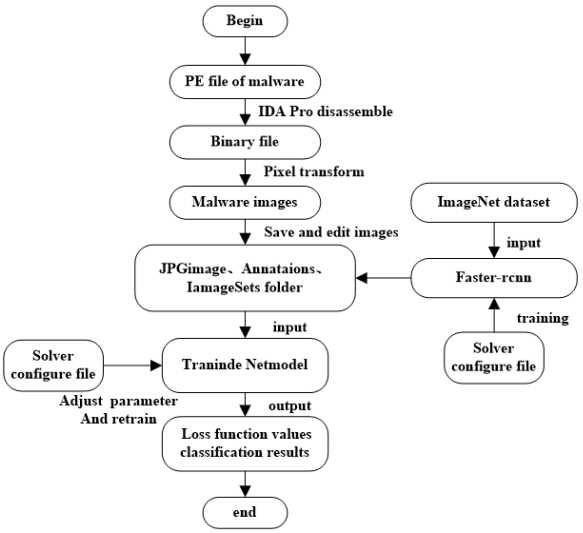


FIGURE 7. Experimental flowchart.

TABLE 2. Experimental hardware and software environment.

Hardware environment	Software environment
graphics card: RTX 2080Ti	operating system: ubuntu16.04
driver: Nvidia 430.34	frame: Caffe
CUDA: 9.0	language: Python

TABLE 3. Malicious code data set of training samples.

Malicious code family name	The number of training samples	Type
Ramnit	342	Worm
Lollipop	269	malicious advertisement
Vundo	281	Trojan
Tracur	278	Trojan
Obfuscator.ACY	400	Malicious advertisement
Gatak	251	back door

code image data set by flipping, cropping, and changing the brightness of the RGB color channel, for training and testing of classification models. The samples information is shown in Table. 3.

C. ANALYSIS OF RESUITS

Experiment 1 (A Faster RCNN): We preprocess the malicious samples, and make them into a malware image dataset according to the above, which is inputted into the shown Faster RCNN. Then we train the classification network. In the experiment, malware images set is randomly divided into training set and testing set, 70% of which is training set and 30% of which is testing set. And we use the cross-validation method to train the model. After the training model is completed, the model's loss function is used for judgment and evaluation.

As shown in Fig. 8, is the graph of the total loss function during the training process. As training iteration increases, the total loss value decreases and gradually stabilizes. When the iterations reach 70,000 times, the total loss value has become flat. The result shows that the training model based on Faster RCNN is successful, which can effectively detect malware.

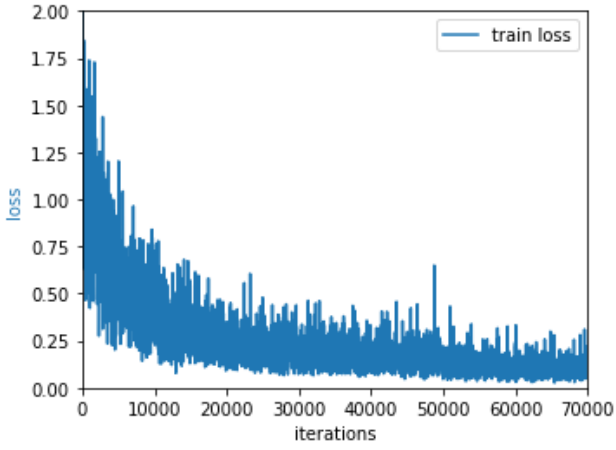


FIGURE 8. Total loss function curve.

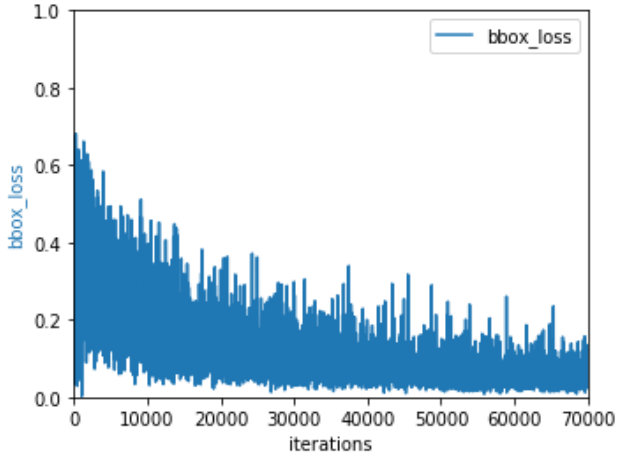


FIGURE 9. The bounding box regression loss function curve.

Fig. 9 is the bounding box regression loss (***bbox_loss***) function curve. Fig. 10 is the classification loss (***cls_loss***) function curve of the fully connected layer. Because the texture change between the various sections of the malicious code image is not very obvious, the ***cls_loss*** value and ***bbox_loss*** value fluctuate greatly, but the overall trend is continuously decreasing.

Fig. 11 is the regression loss function curve of the RPN. Fig. 12 is the classification loss function curve of the RPN. The results show that as the number of iterations increases, the two loss functions converge gradually.

Experiment 2 (An Improved Faster RCNN Combining Transfer Learning): We used the ImageNet image set to pre-train Faster RCNN, and then transferred the model to a malware classification. In the new model, we fine-tuned the parameters of the convolutional layer and the fully connected layer, and modify the relevant configuration file to retrain the network according to the newly defined objective function.

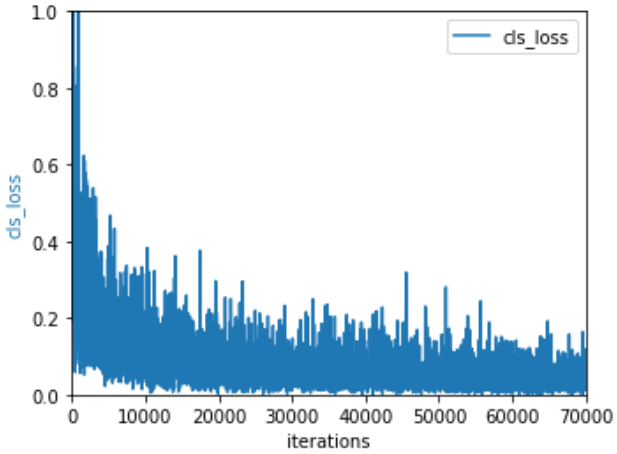


FIGURE 10. Classification loss function curve.

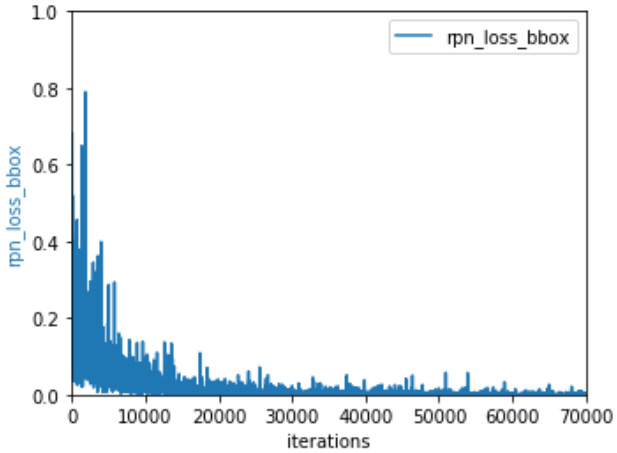


FIGURE 11. Regression loss function curve of the RPN.

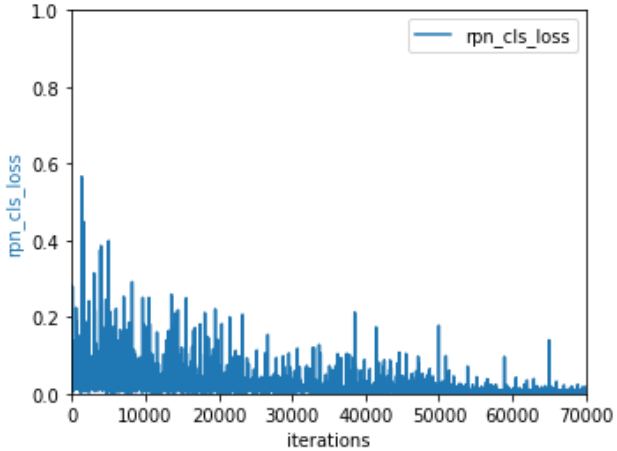


FIGURE 12. The classification loss function curve of the RPN.

The graph of the total loss function in the training process of transfer learning is shown in Fig. 13. As the iterations increase, the convergence speed of the objective function is faster. When the iterations reach about 20,000 times,

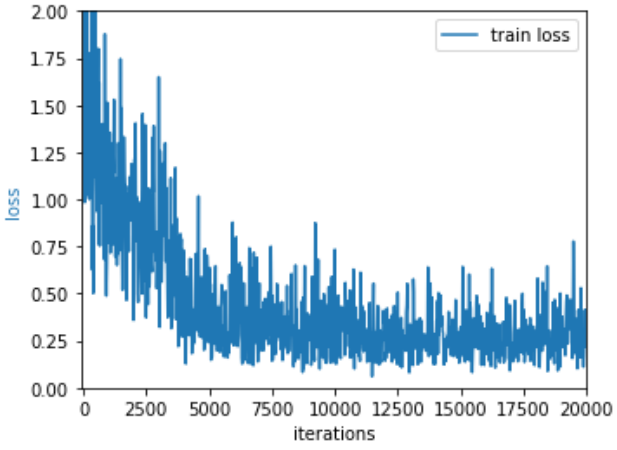


FIGURE 13. The total loss function curve after improved transfer learning.

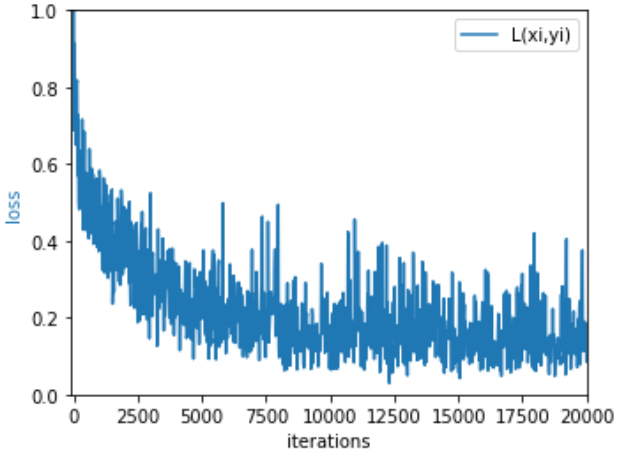


FIGURE 14. Curve of a novel multi-label loss function L(w).

the total loss curve has become flat, which shows that the model after transfer learning can quickly converge to a stable state.

Because there is no obvious texture boundary between the ".text" section and others in the malicious code image samples, it will cause an error in the regression of the candidate suggestion box of the ".text" section during back propagation, which can make the classification proportion decrease and then affect the classification accuracy. Therefore, we redefined the new objective function to improve the parameter update in back propagation. As shown in Fig. 14, is the curve of a novel multi-label loss function $L(w)$.

Fig. 15 is the regression loss function curve of the fully connected layer. Fig. 16 is classification loss function curve of the fully connected layer.

D. COMPARISON OF EXPERIMENTAL RESULTS

We analyze the classification results of malicious code between the two models trained in Experiment 1 and Experiment 2. The accuracy curves of the two detection models on the malicious code image data set are shown in Fig. 17. The model 1 represents the detection model trained in experiment

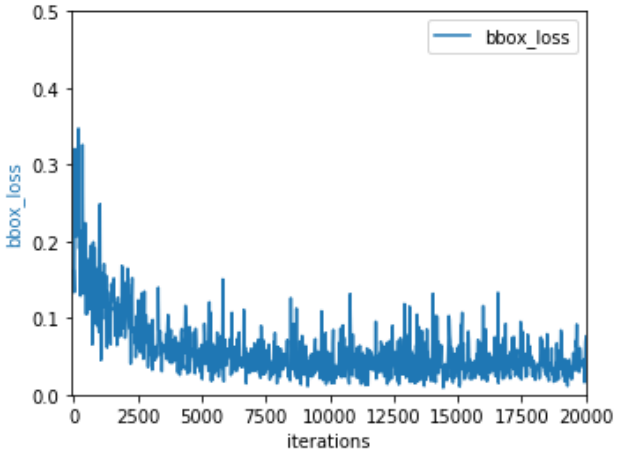


FIGURE 15. Regression loss function curve of the fully connected layer.

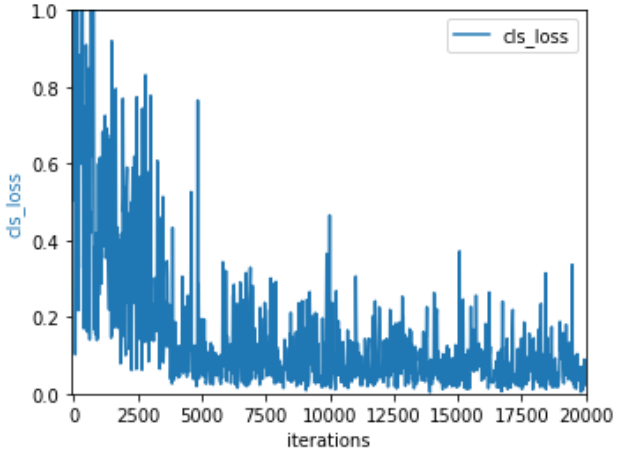


FIGURE 16. Classification loss function curve of the fully connected layer.

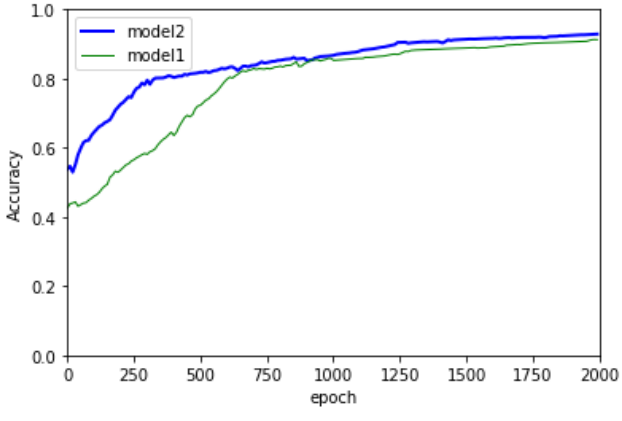


FIGURE 17. Accuracy curves of two models.

one, and the model 2 represents that in experiment two. As can be seen from the accuracy curve, with the increase of iteration times, the parameters tend to be stable and the accuracy is also continuously improved. In experiment 2, because of the fast convergence speed of transfer model, the model parameters tend to be stable faster than those in

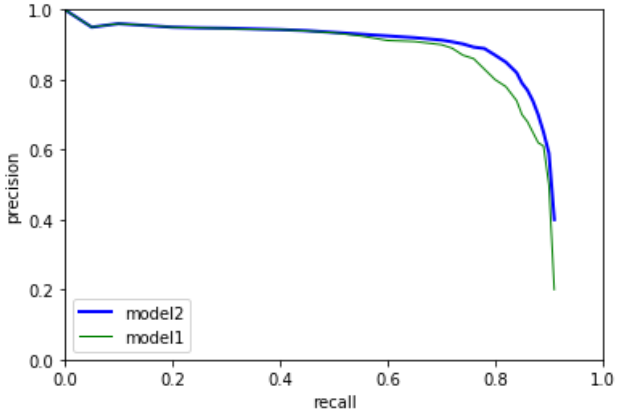


FIGURE 18. P-R curves of two classification models on malware images dataset.

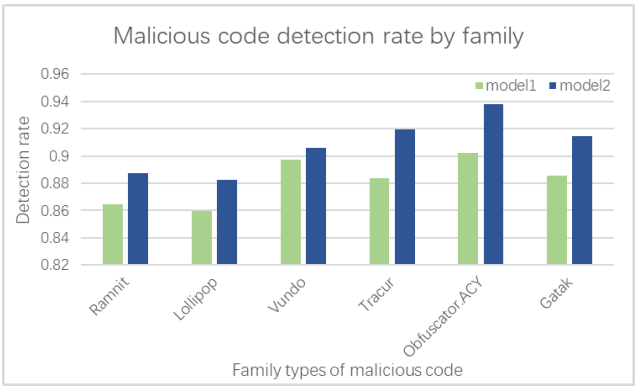


FIGURE 19. Comparison of detection rates of two models.

Experiment 1, and the accuracy after transfer is higher than before.

The P-R curves of the two detection models on malware image dataset are shown in Fig. 18.

As shown in Fig. 18, the detection result of model 1 is slightly worse than that of model 2. This is because the back-propagation error of the Faster RCNN return to a wide range on the candidate suggestion box of the ".text" section, and the classification proportion is relatively low. The texture change between the ".text" section and others is not very obvious. After transfer learning the classification proportion of multiple labels is increased, and the model continuously learn the malicious code image features during the back propagation process, which increases the detection performance of the model.

The comparison results of detection rates of the two detection methods are shown in Fig. 19. The detection rate of model 2 is higher than that of model 1. This is because training a deep neural network classifier requires a large amount of data samples, the transfer model that has been trained from ImageNet dataset obtains lots of image features, which contribute to higher accuracy and generalization in an improved model. On the other hand, the novel lost function of multiple labels reduces the overfitting produced by fast RCNN in the large-scale regression of location in the back

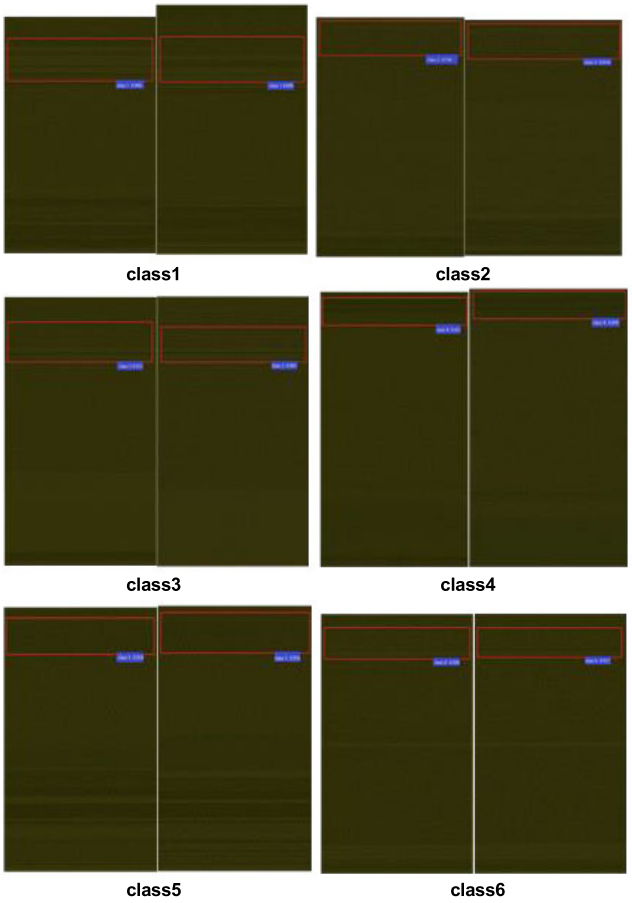


FIGURE 20. Visualization results of malicious code.

propagation, optimizes the classification proportion, and thus improves the detection rate.

We randomly select malware images in the testing set for verification. The output results of texture images from 6 malware families are shown in Fig. 20. The red box in the figure represents the ''.text'' section of the detected malicious code, and the blue box indicates the category to which the predicted malicious code belongs.

We compare accuracy rate of classification, false alarm rate and training time. The result is shown in Table. 4.

As shown in Table. 4, the improved model achieves an accuracy rate of 92.8%, a false positive rate of 6.8%, and training time of 85 min.

Furthermore, we compare our improved deep learning models with the traditional image methods. We use three traditional image processing methods [25]–[27] to extract the texture features of malicious code images, then use machine learning methods to classify malware, and finally generate malicious code detection model. The comparison results are shown in Table. 5.

The experimental results show that the improved Faster RCNN model is better than the traditional methods. Firstly, CNN can better extract the deep-level features of the malicious code image. Secondly, the RPN of Faster RCNN can locate the texture features of the ''.text'' section of malicious code images. Thirdly, the transfer model that has been trained

TABLE 4. Performance comparison of two models.

Method	Accuracy (%)	False positive rate (%)	Training time (m)
Faster RCNN	90.4	7.6	106
An improved Faster RCNN combining transfer learning	92.8	6.8	85

TABLE 5. Comparison of experimental results of different methods.

method	Accuracy (%)	Detection rate (%)	False positive rate (%)
LBP+KNN	88.5	90.3	11.3
Nataraj	91.8	90.3	6.9
Gist+RF	91.0	91.6	8.3
Faster RCNN	91.4	93.5	9.8
An improved Faster RCNN combining transfer learning	92.8	95.6	6.8

from ImageNet dataset can obtain lots of basic image features, which contribute to higher accuracy, generalization and speeds up convergence in the new model.

V. CONCLUSION AND OUTLOOK

This article proposes a detection method of malicious code visualization based on an improved Faster RCNN with transfer learning. We convert the PE files of the malicious samples into a binary files of static disassembly. Combined with computer vision technology, we map these binary files into the corresponding the malicious codes grayscale images. There is a big difference in the image texture between different families of malicious codes, while the malicious codes of the homogeneous family have a large similarity in image texture. Therefore, we adopt code visualization technology to display malicious code samples in the form of grayscale images. We pre-train the Faster RCNN model with the ImageNet image sets, and then transfer the model, fine-tune the parameters, and build a multi-label loss function. In addition, we train this model using 1,200 malware samples from 6 families provided by Microsoft Corporation. The experimental results show that the improved method proposed in this article can accelerate the model's convergence and achieve 92.8% accuracy and 6.8% false positive rate. We also compared three traditional malicious code classification methods. The improved method of Faster RCNN with transfer learning is better than other methods. In the future, we intend to expand the detection capabilities of Faster RCNN model. First, we will further study the information enhancement processing to make texture features of the ''.text'' section more obvious and expand sample types. Meanwhile, we plan to focus on domestic and foreign research on optimization algorithms for deep feature analysis to speed up the computing power of deep learning model.

REFERENCES

[1] National Internet Emergency Center. *Network Security Information and Dynamic Weekly Report*. Accessed: Jul. 2018. [Online]. Available: http://www.cert.org.cn/publish/main/upload/File/2018CNCERT12.pdf

[2] Virus Total. (2017). *File Statistics*. [Online]. Available: https://www.virustotal.com/en/statistics/
[3] E. Antonakakis, "Method and system for detecting malware," U.S. Patent 8 578 497, 2013.
[4] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.*, vol. 7, no. 4, pp. 247–258, Nov. 2011.
[5] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.
[6] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 470–478.
[7] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Visualizat. Cyber Secur. (VizSec)*, 2011, pp. 1–7.
[8] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proc. 4th ACM workshop Secur. Artif. Intell. (AISec)*, 2011, pp. 21–30.
[9] J. Zhang and Y. Peng, "Classification of malicious code based on feature fusion," *Comput. Eng.*, vol. 45, no. 8, pp. 1–7, Jul. 2019.
[10] J. Upchurch and X. Zhou, "First byte: Force-based clustering of filtered block N-grams to detect code reuse in malicious software," in *Proc. 8th Int. Conf. Malicious Unwanted Softw., Amer. (MALWARE)*, Oct. 2013, pp. 22–24.
[11] L. Shiqi, T. Shengwei, S. Hua, and Y. Long, "Research on malicious code classification algorithm of stacked auto encoder," *Appl. Res. Comput.*, vol. 35, no. 1, pp. 262–265, Jan. 2011.
[12] G. Liao and J. Liu, "A malicious code detection method based on data mining and machine learning," *J. Inf. Secur. Res.*, vol. 2, no. 1, pp. 74–79, Jan. 2016.
[13] J. Li *et al.*, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.
[14] T. Wuchner, A. Cislak, M. Ochoa, and A. Pretschner, "Leveraging compression-based graph mining for behavior-based malware detection," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 1, pp. 99–112, Jan. 2019.
[15] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug.2013, pp. 1357–1365.
[16] Y. Liu, Z. Wang, Y. Hou, and H. Yan, "Visualization and automatic classification of malicious code with enhanced information density," *J. Tsinghua Univ. (Natural Sci. Ed.)*, vol. 59, no. 1, pp. 9–14, 2019.
[17] D. Baysa, R. M. Low, and M. Stamp, "Structural entropy and metamorphic malware," *J. Comput. Virol. Hacking Techn.*, vol. 9, no. 4, pp. 179–192, Nov. 2013.
[18] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* (Advanced Technology Publications). Amsterdam, The Netherlands: Elsevier, 2014.
[19] G. Conti, S. Bratus, A. Shubina, B. Sangster, R. Ragsdale, M. Supan, A. Lichtenberg, and R. Perez-Alemany, "Automated mapping of large binary objects using primitive fragment type classification," *Digital Invest.*, vol. 7, pp. 3–12, Aug. 2010.
[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[21] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, no. 2, 2012, pp. 1097–1105.
[23] Y. Zhao, "Deep learning: 21 days of practical caffe," Electron. Ind. Press, Beijing, China, Tech. Rep., 2016, pp. 20–43.
[24] *Kaggle*. Accessed: Jul. 2017. [Online]. Available: https://www.kaggle.com/datasets
[25] X. Guan *et al.*, "Research on signature based virus scanning technology," *Inf. Netw. Secur.*, vol. 4, no. 4, pp. 8–13, 2013.
[26] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variantdetection," *IEEE Trans. Depend. Sec. Comput.*, vol. 11, no. 4, pp. 307–317, Jul./Aug. 2014.
[27] I. Ghafir and V. Prenosil, "Malicious file hash detection and drive-by download attacks," in *Proc. 2nd Int. Conf. Comput. Commun. Technol.*, 2016, pp. 661–669.
[28] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, Apr. 2019.
[29] Y. Li, K. Xiong, T. Chin, and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection," *IEEE Access*, vol. 7, pp. 32765–32782, Jan. 2019.

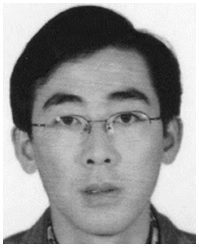
YUNTAO ZHAO received the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2013. He is currently a Postdoctoral Researcher in pattern recognition and artificial intelligence with Northeastern University, Shenyang, China, in 2015. He is also a Professor with the Communication and Network Institute and the School of Information Science and Engineering, Shenyang Ligong University, Shenyang. He has authored over 30 papers published in related international conference proceedings and journals. He is the holder of ten patents and software copyrights. His main research interests include deep learning, AI algorithm, cyberspace security, protocol analysis, and data mining.

WENJIE CUI was born in Zibo, Shandong, China, in 1996. She received the B.S. degree in communication engineering from Binzhou University, in 2018. She is currently pursuing the M.S. degree in communication engineering with Shenyang Ligong University, Shenyang, China. Her research interests include network communications and security, deep learning, and malicious code classification technology.

SHENGNAN GENG received the Ph.D. degree in instrument science and technology from Tsinghua University, Beijing, China, in 2012. She is currently a Senior Engineer with the Beijing Institute of Astronautic Systems Engineering, China. She has authored over ten papers published in international conference proceedings and journals. Her main research interests include sensing technology, measurement and control, AI algorithm, and data mining.

BO BO received the master's degree in communication and information from the Shenyang University of Technology, Shenyang, China, in 2020. She has authored two articles published in related journals. Her main research interests include deep learning, artificial intelligence, and data analysis.

YONGXIN FENG received the M.S. degree in computer science from Northeastern University, in 2000, and the Ph.D. degree in computer science and technology from the School of Information Science and Engineering, Northeastern University, in 2003. She is currently a Professor with Shenyang Ligong University. She has authored over 60 papers in related international conferences and journals. Her research interests include network management, wireless sensor networks, and communication and information systems. She was a recipient of the ICINIS 2011 Best Paper Awards and 15 Science and Technology Awards, including the National Science and Technology Progress Award and the Youth Science and Technology Awards from the China Ordnance Society.



WENBO ZHANG received the Ph.D. degree in computer science and technology from Northeastern University, China, in 2006. He is currently a Professor with the School of Information Science and Engineering, Shenyang Ligong University, China. He has authored over 100 papers in related international conferences and journals. His current research interests include *ad hoc* networks, sensor networks, satellite networks, and embedded systems. He was a recipient of the ICINIS 2011 Best Paper Awards and nine Science and Technology Awards, including the National Science and Technology Progress Award and the Youth Science and Technology Awards from the China Ordnance Society. He has served on the Editorial Board for up to ten journals, including the *Chinese Journal of Electronics* and the *Chinese Journal of Astronautics*.