Hindawi
Security and Communication Networks
Volume 2020, Article ID 8881760, 19 pages
https://doi.org/10.1155/2020/8881760

Research Article

Binary File's Visualization and Entropy Features Analysis Combined with Multiple Deep Learning Networks for Malware Classification

Hui Guo,[1] **Shuguang Huang**,[1] **Cheng Huang**,[2] **Fan Shi**,[1] **Min Zhang**,[1] **and Zulie Pan**[1]

[1]*College of Electronic Engineering, National University of Defense Technology, Hefei 230011, China*
[2]*College of Cybersecurity, Sichuan University, Chengdu 610065, China*

Correspondence should be addressed to Fan Shi; shifan17@nudt.edu.cn

Received 2 September 2020; Revised 6 November 2020; Accepted 20 November 2020; Published 4 December 2020

Academic Editor: Liguo Zhang

In recent years, the research on malware variant classification has attracted much more attention. However, there are still many challenges, including the low accuracy of classification of samples of similar malware families, high time, and resource consumption. This paper proposes a new method of malware classification based on multiple visual features of malware and deep learning algorithms. In prior research, visualization techniques and entropy demonstrated exemplary performance in many areas. This paper extracts numerous visual features from the raw bytes and entropy sequence of the malware, which makes it more sensitive to malware samples of similar families and endows it the ability to classify malware variants more accurately. To evaluate the proposed method, this paper conducted a series of experiments on two malware datasets with a total of more than 20,000 samples provided by the Malware Research Lab and Microsoft Research. Through experiments, the method showed its superiority compared with some leading malware visual classification methods, achieving good performance on the accuracy with at least 1% improvement. The accuracy of the method even could reach 99.73% and 99.54%, respectively, on the two datasets.

1. Introduction

In recent years, the exponential growth of malware has posed a serious threat to cyber security. According to the Symantec internet security threat report [1], 246,002,762 new malware variants were monitored in 2018, and the number of new malware variants has exceeded a billion in the past three years. Hackers are increasingly inclined to use techniques such as packing and encryption to slightly modify the original malicious code to create new malware variants. Therefore, the rapid and accurate identification of malware variants can effectively assist the cyber security personnel in grasping their harmfulness and other attributes, which has significant research value.

With the development of machine learning techniques, data mining methods are often used to analyze malware, and many features-based detection methods are proposed [2]. These methods first extract the features of malware and then detect malware by using these features. This approach has become the mainstream method of malware detection. Currently, malware detection methods consist primarily of two types of approaches: static-feature-based detection and dynamic-features-based detection. The malware detection based on static features mainly analyzes the raw bytes of malware or disassembles the malware to analyze its opcodes, file structure, and other file attributes. The dynamic detection methods often extract behavioral features such as API operation sequence, file operations, and network communication during its process by running malware samples in a virtual environment or sandbox.

1.1. Need for This Study

1.1. Need for This Study. However, code obfuscation technology could modify malware, increase the difficulty of code reverse, and reduce the performance of static malware detection. Dynamic detection is more robust, but it is still disturbed by different kinds of countermeasures (e.g., strict triggering conditions for malicious behavior or increasing waiting time to avoid dynamic detection), reducing the detection performance. Moreover, the time and resources consumption of malware execution is often expensive.

In recent years, with the rapid development of deep learning algorithms and image recognition technologies, rather than focusing on these nonvisual detection methods, many researchers have proposed new classification methods based on malware visualization methods [3–7]. They transform the raw bytes of malware into grayscale images and extract the malware texture features for classification. Besides, these visualization methods are also proven to be more robust than the static method [3] and provided performance roughly equivalent to dynamic detection methods, but in less time [5].

Although the visualization detection methods can handle some code obfuscation problems and show some superiority, they still have some limitations and challenges. The malware variant has many homologous parts with its ancestors, which makes it have a strong correlation in visual characteristics. However, the malware samples of similar families also have some correlations, showing some similarities in the visual features, which would cause some interference to the malware classifier and reduce the performance. The latest related methods, combined with deep learning algorithms, have partially alleviated the problem. But it is still not entirely resolved. Therefore, a critical challenge in the field is how to effectively distinguish the samples of similar malware families and improve classification performance.

1.2. Major Contributions of the Study. To address the above challenges, this paper proposes a new malware classification method based on malware visual features, entropy features, and deep learning algorithms. In the field of image recognition, the deep learning algorithm often performs better than traditional feature extraction algorithms. It can enable computers to automatically learn important features in images to improve the performance. Besides, the entropy, as a measure of randomness or uncertainty, has achieved good performance in detecting encrypted, compressed malware [8–11]. Therefore, the entropy feature and deep learning algorithms may assist us in learning more detailed features contained in malware samples to classify the malicious code more accurately and improve the classification performance of malware samples of similar families. Based on this hypothesis, this paper extracts multiple features of the raw bytes and entropy features of malware based on the deep learning algorithms.

First, we transform the raw bytes of malware into RGB images in bytes sequence order and extract bottleneck features based on the convolutional layer of a deep learning network. Then, the malware is converted into another RGB images using a visualization method containing location information, and the bottleneck feature is also extracted. Besides, this paper extracts the entropy sequence of the malware and then extracts the visualization features of the entropy sequence based on the grayscale image visualization method. At last, a machine learning malware classifier is trained based on these three types of features.

To evaluate the proposed method, this paper conducted cross-validation experiments on two different malware datasets with a total of more than 20,000 samples provided by the Malware Research Lab and Microsoft Research. The experimental results show that the proposed method could significantly improve the classification performance of malware samples of similar families and achieve better performance on the accuracy with at least 1% and 2% improvement on two datasets compared with the leading malware visual classification methods, and the accuracy of the method even could reach 99.73% and 99.54%, respectively, on the two datasets.

Overall, we make the following contributions:

A new malware classification method was proposed based on the raw bytes of malware. It transforms malware into three kinds of images based on multiple visualization methods, including visualization methods containing location information and entropy features. The combination of these three types of features could effectively improve the performance of malware classification.

This paper introduces a visualization method of the entropy sequence. Unlike other traditional methods, we mine entropy features based on deep learning algorithms, and the experimental results show that it can play an essential role in malware classification.

Extensive experimental results showed that the proposed method almost solved the problem of confusion of similar family samples in malware classification and effectively improved the performance of malware classification, which is of great significance for the future research.

2. Related Work

In this section, the related research of malware classification is presented, mainly including malware detection methods based on static features, dynamic features, and visualization features.

2.1. Malware Detection Based on Static Features. The static feature of malware includes raw bytes feature, opcode, PE header features, import and export table, CPU register features, and static API sequence. The malware static detection method extracts these features to train a classifier for malware classification. Piyanuntcharatsr et al. [12] processed the byte sequence and opcodes with the N-gram algorithm and trained a malware classifier based on the decision tree algorithm. Feng et al. [13] extracted the bytes features of the malware and trained the malware classifier based on the support vector machine (SVM) algorithm. Raff and Nicholas [14] processed the raw bytes of malware based on the k-nearest-neighbor (KNN) algorithm to train a malware classifier. Kong and Yan [15] disassembled the malware samples and extracted the function call graphs for malware classification. Upchurch and Zhou [16] extracted the raw bytes and disassembly features of malware and classified the malware based on multiple methods. The static detection method does not need to run malicious code files and would not cause harm to the system. However, it relies heavily on the file features of malware, which is less robust and vulnerable to code obfuscation technologies.

2.2. Malware Detection Based on Dynamic Features. Aiming at the problem of low robustness of static features, researchers have executed malware samples in virtual machines, sandboxes, and other virtual environments. They extract the malware behavioral features, including file operation behaviors, network communication behaviors, and runtime API call sequences, and train malware classifiers based on machine learning algorithms. Kawaguchi and Omote [17] extracted the dynamic API features of malware samples and classified the malware based on the features of the functions and machine learning algorithms. Vadrevu and Perdisci [18] proposed a novel malware testing framework and executed the malware samples in an analysis environment and extracted the network activity for malware classification. The experimental results in the paper showed that using their framework, they could reduce the malware execution time. Kim et al. [19], Dai et al. [20], and Lin et al. [21] also proposed malware classification methods based on the behavior features of malware samples. Malware detection technology based on dynamic features overcomes the impact of code obfuscation technology and has stronger robustness. However, these methods are still challenged by different kinds of countermeasures [5] (e.g., set waiting time measures and operation environment detection measures). Moreover, dynamic detection is time-consuming because it usually consumes time to execute malware samples, making it unsuitable for large datasets.

2.3. Malware Detection Based on Visualization Features. With the rapid development of image recognition technology, many scholars have utilized visualization technology for malware detection. Nataraj et al. [3] proposed a new method to extract malware visualization features and classify malware based on the k-nearest-neighbor (KNN) algorithm. They transformed the malware samples into grayscale images and extracted the texture features based on the GIST algorithm. Compared with the traditional feature-based methods, their method could provide more improved results [5]. Kosmidis and Kalloniatis [22] conducted more research on the visualization feature of malware and evaluated the performance of different machine learning algorithms. Inspired by these studies, Naeem et al. [6, 23], Xiaofang et al. [24], and Hashemi and Hamzeh [25] proposed many new malware classification methods based on different visualization features of malware images (e.g., DSIFT, LBP, and SURF). Moreover, the researchers also combined visualization technology with other malware characteristics for malware detection. Zhang et al. [26] visualized the opcode sequences of malware and achieved good accuracy for a small training set. Han et al. [27] visualized the dynamic features and opcode sequences of malware and extracted visualization features for malware classification.

Besides, with the rapid development of deep learning technology, Cui et al. [4], Rezende et al. [28], and Tang et al. [7] proposed better methods of malware classification based on deep learning algorithms. Cui et al. [4] visualized the malware samples and classified the malware based on convolutional neural networks. They also propose a method to improve the classification performance in the case of insufficient training samples. Tang et al. [7] also proposed a malware classification method based on deep learning algorithms. They significantly alleviated the lacking data problem and improved the performance of malware classification. Rezende et al. [28] proposed a malware classification method based on transfer learning algorithms. They extracted the visualization features of the malware based on the VGG16 network and achieved good performance. The visualization-feature-based detection methods are efficient and provide better performance relative to some traditional methods, but they still have some challenges. For example, the classification accuracy of malware samples of similar families is relatively poor. The latest related methods, combined with deep learning algorithms, have partially alleviated the problem. But it is still not entirely resolved, and the accuracy of classification needs to be further improved.

2.4. Malware Detection Based on Entropy Features. Besides, many scholars have detected the malware based on the entropy features. Wojnowicz et al. [8], Bat-Erdene et al. [9], and Liu et al. [10] proposed different kinds of methods for malware classification based on the entropy features of malware. Wojnowicz et al. [8] proposed a method for the detection of parasitic malware based on the entropy features and achieved good performance. Bat-Erdene et al. [9] proposed a method to detect the packing algorithm of malware based on the entropy features. Liu et al. [10] extracted the entropy sequence of malicious documents and detected the malware based on the machine learning algorithms. Moreover, Canfora et al. [11] extracted the entropy features of Android malware and proposed a malware detection method. Therefore, the entropy features of malware should also be one of the critical attributes, which may play an essential role in the research of malware classification.

3. Methods

3.1. Overview. First, we extract the raw bytes of malware to mine features hidden in the malware. The raw bytes of the malware are processed in two ways (i.e., extracting raw byte stream data and extracting entropy sequence data). The extracted data are processed based on three visualization methods to obtain different kinds of visualization representations. Then, three types of features (byte sequence level RGB feature, location information level RGB feature, and entropy sequence feature) are extracted based on the transfer learning algorithms. At last, a malware classifier is trained based on these features, and it would classify new malware samples to identify their families. An overview of our method is shown in Figure 1.

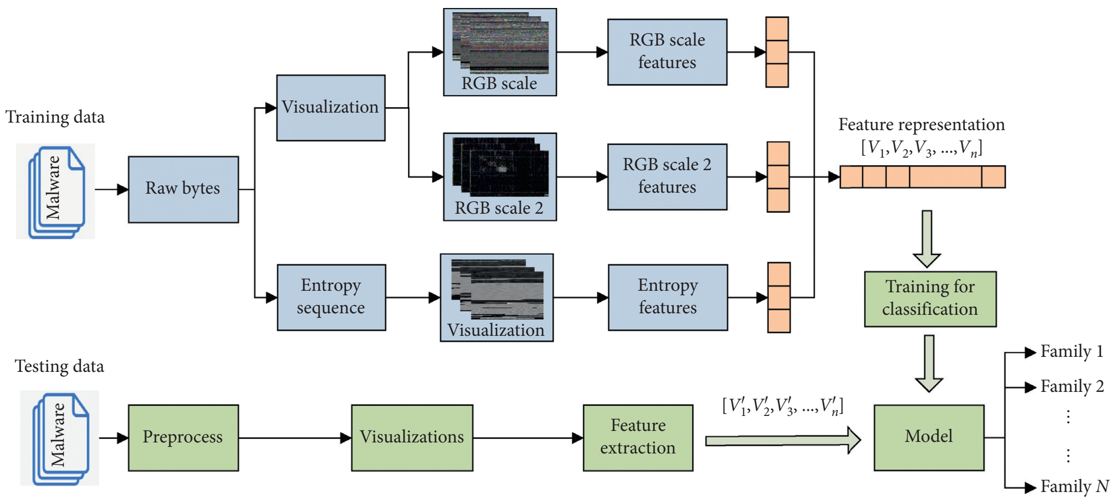


Figure 1: Overview of the method.

Specifically, for the byte sequence level RGB feature, we transform the raw bytes of malware into RGB image representation in bytes sequence order and extract the bottleneck features based on the convolutional layer of the deep learning network. For location information level RGB feature, we convert the malware into an RGB image representation using a visualization method containing location information. For entropy sequence feature, we first extract the entropy sequence of the malware, then convert the entropy sequence into a grayscale image representation, and extract the bottleneck features. Finally, a classifier is trained based on three types of features to classify malware.

3.2. The Byte Sequence Level RGB Feature. The features extracted from malware have a significant impact on classification performance. The method based on the malware visualization and texture features has been proved to be useful in classifying malicious code. In the field of image recognition, the deep learning model based on RGB images has better results than most traditional image recognition technologies. Therefore, this paper first extracts malware features based on the deep learning model and RGB images representation. This method mainly involves two steps: first, it transforms the malicious code into an RGB visual representation; second, it extracts the features contained therein. However, different malware visualization features would have different effects on the classification of malicious codes. In this section, we first visualize the malicious code based on byte sequence order and extract features as one of the critical elements of malicious code classification.

3.2.1. The Byte Sequence RGB Visualization Method. This paper visualizes the malware based on the raw bytes of malware samples. The raw bytes of the malicious code are a 1, 0-bit data stream, and each byte contains 8-bit data. Without processing malware samples, this paper directly extracts features based on the raw bytes of the malicious code to ensure that the malware image can contain all the features of the malicious code. At the same time, it can improve the efficiency of malware feature extraction and facilitate large-scale malicious code detection.

When the byte data sequence of the malware is obtained, it needs to be converted into RGB image representation, which involves two key issues: first, how to convert malicious code data into the pixel data of image; second, how to set the image format (i.e., height and width). For the first issue, the byte data of the malware are treated as 8-bit unsigned integer data, and the value range of each byte data is [0, 255]. The RGB channel data value range of each pixel is also [0, 255], and each pixel contains three channels of red, green, and blue. Therefore, every three bytes of data are combined into one group. According to the one-to-one correspondence relationship between each group of data and the three-channel data of pixels, the byte data sequence is converted into RGB image pixels in sequence order. The process is shown in Figure 2.

For the second issue, the range of malicious code file size is large. To ensure that the image has an appropriate aspect ratio, this paper formulates different image size setting standards based on the malware file size. The specific content is shown in Table 1.

The size of the malware is different, and the malware visualization image size is also different. To improve the performance of malware classification and facilitate the application of the deep learning algorithms, this paper normalizes the image to a uniform size after the transformation. As shown in Figure 3, there are some malware visualizations of malware samples. The image size of all malicious code samples is unified to $224 * 224 * 3$. It can be seen from the figure that the transformed images of malicious code samples of the same family have strong similarities, and there are some differences between malicious code samples of different families.

3.2.2. Feature Extraction of the Byte Sequence RGB Image. In the field of image processing, deep learning algorithms have been proven to handle various problems better than traditional feature extraction algorithms (such as GIST and LBP).

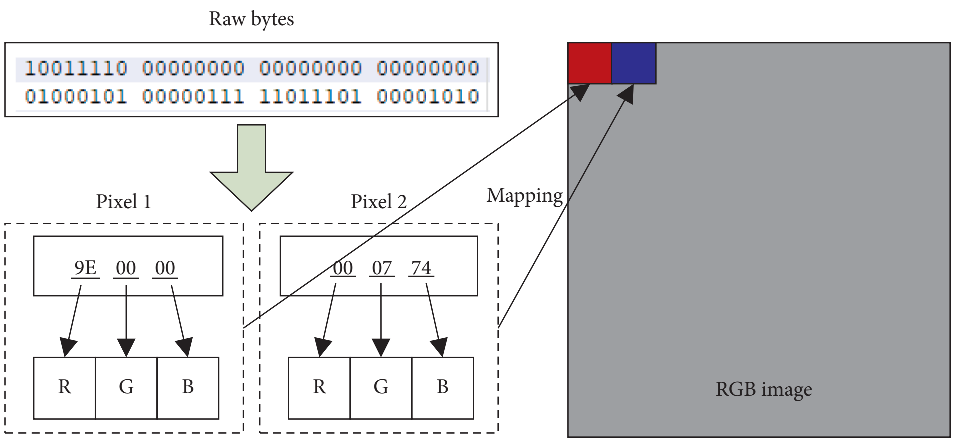
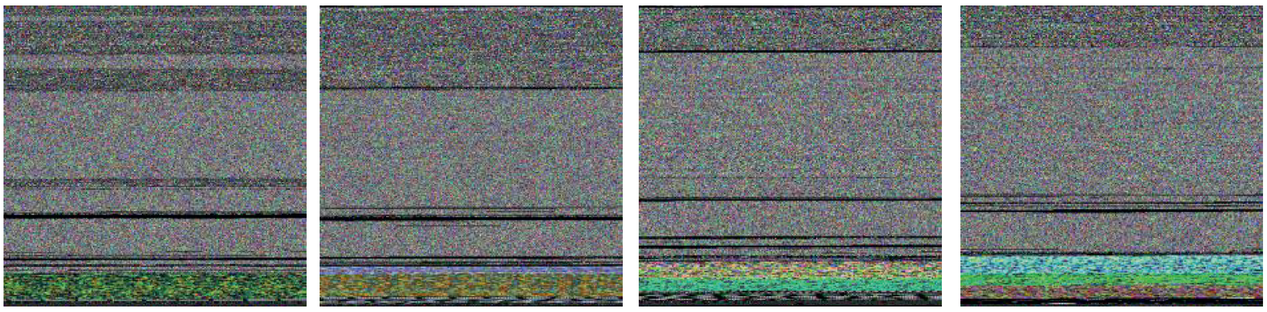


Figure 2: The transformation process of byte sequence RGB image.

Table 1: The byte sequence image conversion width.

File size (kB)	Image width	File size (kB)	Image width
<10	16	100–200	192
10–30	32	200–500	256
30–60	64	500–1000	384
60–100	128	>1000	512



(1) C2LOP.gen!g (2) C2LOP.P

Figure 3: The examples of byte sequence RGB visualization of malware samples.

Moreover, in malware classification, deep learning algorithms can obtain better classification results than texture feature extraction algorithms. Therefore, this paper does not use traditional texture feature extraction algorithms but processes malicious code images based on deep learning algorithms. However, the number of variants of different malware families is various. In some cases, the number of malicious code variants of some families is small (for example, newly emerged malicious code variants), which is not enough to form a training sample set, making the classifier unable to effectively learn the characteristics of the new family. Cui et al. [4] have pointed out that the classification performance of the deep learning classifier is not good enough when the malware samples of some families are insufficient. Therefore, this paper does not directly use deep learning algorithms but trains the classifier based on transfer learning algorithms. Combined with the characteristics of malware variant dataset, this paper trains the malware classifier based on the deep learning network that has been trained on a large-scale image dataset.

First, this paper extracts malicious code image features based on a deep network model. In recent years, some deep learning networks with better recognition effects have been proposed (e.g., ResNet50 [29], Xception [30], and Inception [31–34]). Different deep learning networks have different structures and different performances. The VGG16 deep network and VGG19 deep network [35] include 16-layer and 19-layer networks, respectively, which have achieved good performance in the field of image recognition. Rezende et al. [28] extracted the visualization features of the malware based on the VGG16 network and achieved good performance. However, as the network depth increases, the deep learning network would face some problems, such as the degradation problem [29], which refers to the fact that adding more layers to the suitably deep network would lead to higher training errors. But these problems are address by the ResNet network. This makes it possible to train a network with deeper layers on the dataset, and the number of network layers could increase to more than 50, which has achieved better results than the VGG16 deep network in the field of image recognition. Besides, these deep networks have many different advantages and have achieved better performance than the VGG16 network in the field of image recognition. Therefore, these networks may be able to achieve better classification performance in malware classification. We evaluate the classification performance of these deep networks and finally extract byte sequence RGB image features based on the ResNet50 deep learning network.

The deep learning algorithms have achieved good performance in the field of image recognition, but as the network deepens, the application of deep learning networks will encounter two limitations: vanishing/exploding gradients problem [36–38] and degradation problem [29]. The problem of vanishing/exploding gradients has been addressed by normalized initialization [39, 40] and intermediate normalization layers [32]. However, the deep learning network still faces another problem: the degradation problem. As the network depth increases, the accuracy will gradually increase to saturation, and then, there will be a problem of rapid decline. However, the problem is not caused by overfitting [29]. In the same training round, a network with a deeper network (degraded network) has a higher error rate than a network with fewer layers [41]. But the degradation problem could be addressed by the application of the residual network. In the residual network, the residual unit is introduced, and its structure is shown in Figure 4.

In the residual network, the residual mapping is defined as the following:

$$H(x) = F(x) + x, \tag{1}$$

where $F(x)$ is the residual function. In the deep learning network, the additional residual unit connection method is called shortcut connection. In the residual network, this connection is defined as the following:

$$y = F(x, W_i) + x, \tag{2}$$

where $F(x, W_i)$ represents the residual mapping to be learned. For the example in Figure 4, the $F(x, W_i)$ in which σ represents ReLU [42], $F + x$ represents the shortcut connection. If the dimensions of X and $F(x)$ are different in the deep learning network (e.g., when changing the input/output channels), this connection is defined as the following:

$$y = F(x, W_i) + W_s x, \tag{3}$$

where W_s is a linear projection of the x to make it consistent with the dimension of $F(x)$. By importing residual values and residual units into the network, the correlation between the shallow and deep networks is enhanced, and the influence of the degradation problem is reduced. To evaluate the performance of the Reset50 network, we conducted a lot of experiments. The experimental results show that the residual deep learning network ResNet50 can effectively

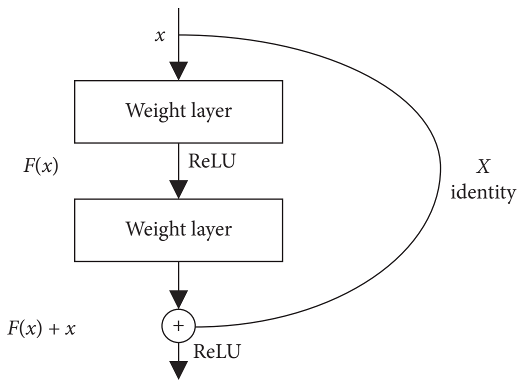


Figure 4: The residual unit.

extract byte sequence RGB image features. The structure of the ResNet50 network is shown in Figure 5. The ResNet50 network includes 50 layers and about 3.8×10^9 parameters in total. Inspired by the transfer learning algorithms, we remove the fully connected layers of the network and use the convolutional layer in the ResNet50 network to extract the byte sequence RGB image features.

3.3. The Location Information Level RGB Feature. The byte sequence level RGB image visualization method uses the raw byte data of the malicious code as the original data of the image pixels, and the locations of the pixels are arranged in byte order, without further considering the location information of the pixels. In this section, an image transformation method containing location information is proposed. The raw bytes of the malware are divided into two parts: location information and pixel information. In this way, the converted RGB image could contain richer information to enhance the performance of malicious code classification.

3.3.1. The Location Information Level RGB Visualization Method. In RGB images, the location of the pixel also has an essential influence on the characteristics of the image. In this section, the malware visualization is also based on the raw bytes of malware, which also involves two key issues: first, how to convert malicious code data into the pixel data of image; second, how to set the image format. In this section, the raw byte of the malware is still regarded as 8-bit unsigned integer data, and every five bytes of data are a group. The first two bytes of data in each group of data represent location information, and the last three bytes of data represents the pixel information of the RGB image. The process of location information level RGB image is shown in Figure 6.

Specifically, the value range of each byte is [0, 255], and we set the size of each image to $256 * 256$. The data coordinates of the bottom left corner of the image are set to $(0, 0)$. All pixels in the image are initialized to 0. For each group of data, the first byte represents the X coordinate of the pixel in the image, the second byte represents the Y coordinate, and the last three bytes of data respectively represent the values of the R, G, and B channels of each pixel in order. Then, each group of data can be converted into a pixel in the image. The

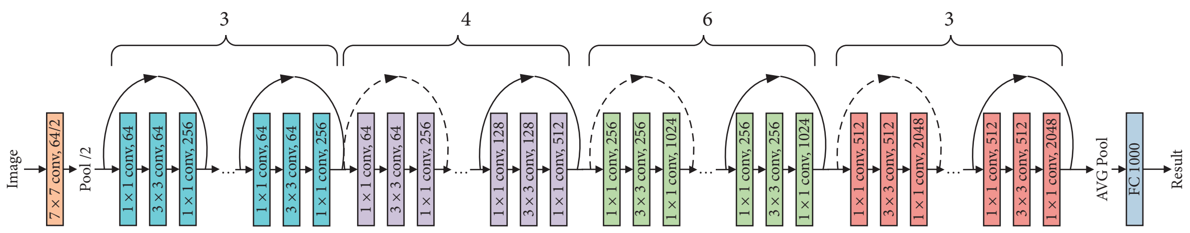


Figure 5: The structure of the ResNet50 network.

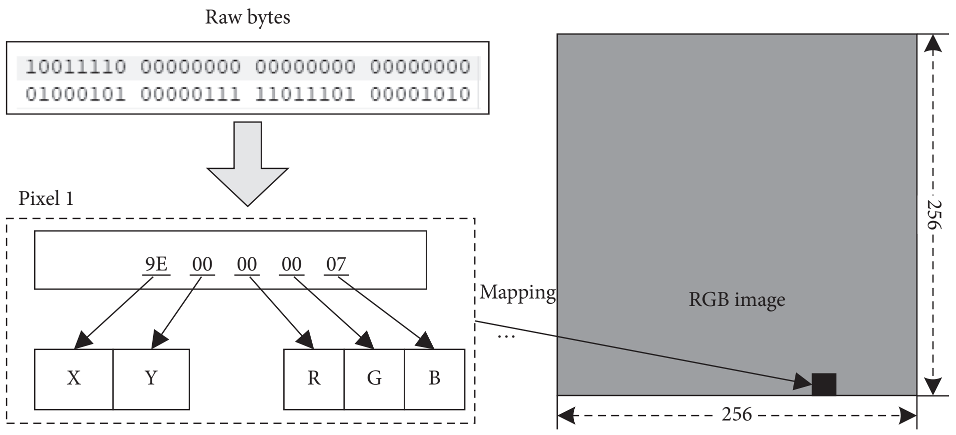


Figure 6: The transformation process of location information RGB image.

```
    The RGB image $M_i$ contains the information of the malware feature, and $(X_m, Y_m, R_m, G_m, B_m)$ is a group of raw byte data of
    malware to be processed.
    **Input:** image $M_i$, $(X_m, Y_m, R_m, G_m, B_m)$
(1) Find the pixel $(R_i, G_i, B_i)$ to the coordinates $(X_m, Y_m)$ in $M_i$
(2)     $R_i = (R_i + R_m)\%255$
(3)     $G_i = (G_i + G_m)\%255$
(4)     $B_i = (B_i + B_m)\%255$
(5) Modify the pixel data of $(X_m, Y_m)$ in $M_i$ to $(R_i, G_i, B_i)$
(6) **Return:** image $M_i$
```

Algorithm 1: Image pixel transformation.

image pixel processing method is shown in Algorithm 1. Each group of data is added to the pixel data of the corresponding coordinate separately, and the sum value is used as the final data of the image pixel. If the data are greater than 255, the 255 remainder operation is performed on the data.

The size of the RGB image is 256 ∗ 256, and there is no need to perform standardized operations on these images. Some examples of the location information level RGB image are shown in Figure 7. It can be seen that the location information RGB level image seems to contain "noise," and there are not many obvious blocks in the image. However, some images still include noticeable lines and blocks. As shown in Figure 7, the samples of similar families C2LOP.gen!g and C2LOP.P exhibit different image features, so it can be inferred that some similar family samples could be classified based on these image characteristics.

3.3.2. Feature Extraction of the Location Information RGB Image. The location information level RGB image shows a big difference from the byte sequence level image, and the performance of different deep learning networks is also different. The ResNet network makes it possible to train a network with deeper layers on the dataset, and the number of network layers even could increase to 200, which has achieved good performance. It has also been found that different convolution kernels have different performance on image recognition. In the process of image recognition, it is

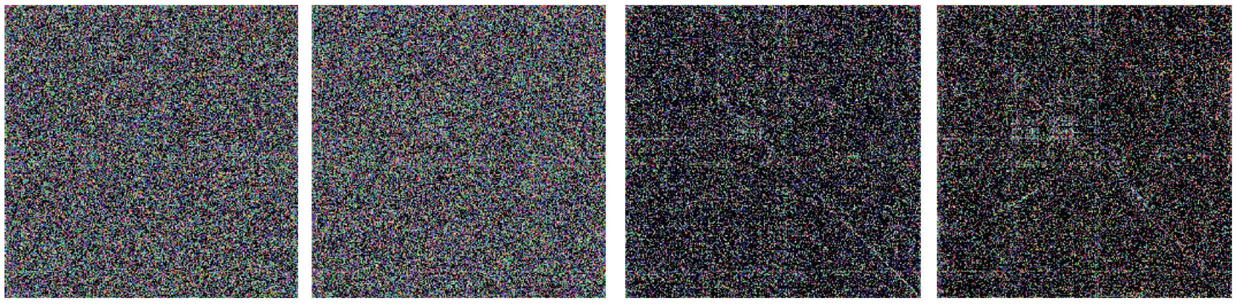


Figure 7: The examples of location information RGB visualization of malware samples.

of great significance to quickly find the best convolution kernel, and the problem is addressed by the Inception network and Xception network. They use multiple filters of different sizes to make the network "wider." For the same image, different convolution kernels extract different features, so through the network training process, the computer can automatically adjust the weight for the image to find the optimal feature extraction method. This method enables the deep learning network to achieve better performance under the same depth. Therefore, the Xception deep network may achieve better performance than VGG16 deep network in malicious code classification. Based on this hypothesis, we evaluate the classification performance of these deep networks and finally extract the location information level RGB image features based on the Xception deep learning networks.

Xception network is another improved model after the Inception V1, V2, and V3 models proposed by Google. Compared with the traditional convolutional neural network, the Inception network model imports the Inception module. It uses multiple filters of different sizes (including $1 * 1$ convolution, $3 * 3$ convolution, $5 * 5$ convolution, and maximum pooling) to make the network "wider.. For the same image, different convolution kernels extract various features; so, through the network training process, the computer can automatically adjust the weight for the image to find the optimal feature extraction method. This method enables the deep learning network to achieve better performance under the same depth. Besides, the Xception network uses the depth-wise separable convolution module in the Inception network to further enhance the performance of the deep learning network. As shown in Figure 8, it is an extreme version of the Inception model. For the input, a $1 * 1$ convolution kernel is used for convolution operation to obtain N channel data, and then, N $3 * 3$ convolution kernels are used to perform the convolution operation. Each channel data are convolved so that each $3 * 3$ convolution kernel is convolved with one channel. At last, all the results are merged as the output.

The depth-wise separable convolution used in Xception is different from the Inception network in the following two points: (1) the order of operations is different. The Xception network first uses M $3 * 3$ convolution kernels to convolve the input data. It then uses N $1 * 1$ convolution kernels and M output results to perform convolution operations to generate N results. However, the Inception network first performs a $1 * 1$ convolution operation. (2) All processes in the Inception network are followed by nonlinear operations (i.e., ReLU). However, most of the depth-wise separable convolution does not require nonlinear operations. The improvements of these two aspects have enabled the Xception network to have a better performance in the field of image recognition. Moreover, the residual module is also imported into the Xception network. The Xception network has 36 convolutional layers for extracting image features, which also uses multiple residual module connections. At the end of the Xception network, a fully connected layer and a logistic regression layer are used to classify images. The network structure is shown in Figure 9. In the figure, "Spconv" is the abbreviation of SeparableConv, which refers to the use of a depth-wise separable convolution module. We remove the fully connected layer in the Xception network and extract the features of the image based on the convolutional layers of the Xception. These features are the second part of the input features of the malicious code variant classifier.

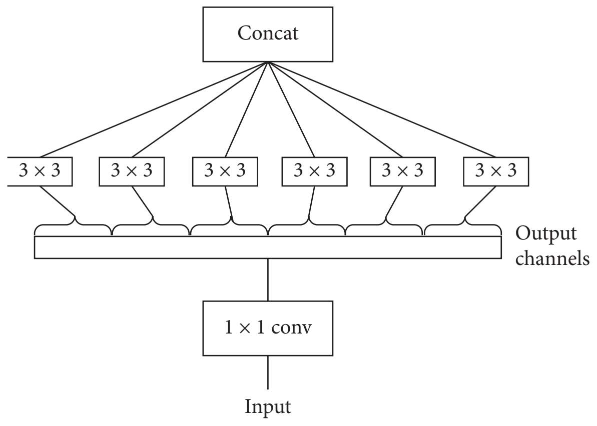


Figure 8: An extreme version of Inception module.

3.4. *The Entropy Sequence Feature.*

The entropy distribution of the raw byte data of malicious code is also one of the essential characteristics of malicious code. In related research, various entropy feature processing methods (e.g., linear features, the bag of words model) have achieved good

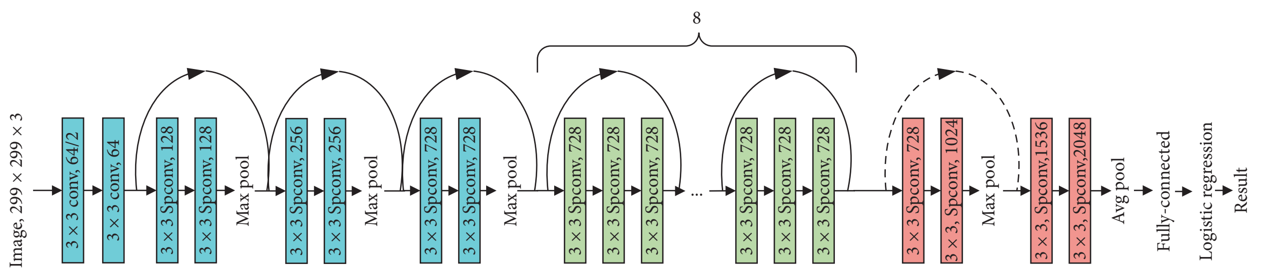


Figure 9: The structure of the Xception network.

results in the field of malicious code detection. Similar to the development of image recognition technology, deep learning algorithms enable the network to automatically learn the features contained in images and can often achieve better results. Therefore, we first extract the entropy sequence of the malware, then visualize the entropy, and process the entropy visualization based on the deep learning algorithms to extract features for malware classification.

3.4.1. Entropy Sequence Extraction. The entropy is a measure of the randomness and uncertainty of data distribution. To compute the entropy sequence of the malware, we first divide the raw bytes of the malware into continuous data blocks (the data are represented in hexadecimal: 00h-FFh), then compute the entropy of each block, and finally connect the entropy of each block according to the order of the blocks to form the entropy sequence. The value range of each byte is [0, 255]. It is crucial to ensure that all data in the block can be used to compute the entropy value. So, the size of the block is set to 256. When computing the entropy sequence, if the length of the last block is less than 128 bytes, the block will be discarded. Otherwise, the block is supplemented with data zero to make its length reach 256.

For each block, the method of computing entropy is as follows:

$$H(X) = -\sum_{i=0}^{255} p(x_i) * \log 2\, p(x_i), \tag{4}$$

where x_i represents a specific raw byte value and p_i represents the probability (frequency) of this value in the block, $H(x)$ represents the entropy value of the block, and the range of its value is zero to eight. When all bytes in the block are equal, the value of entropy is zero. If all the values in the block are different, the value of entropy is eight. If the raw byte of the malware is divided into N blocks, we represent the entropy sequence as $H_s = h_1, h_2, h_n$. Some examples of the entropy sequence are shown in Figure 10. Figures 10(a) and 10(b) show the entropy sequence of the malware samples of family Rbot!gen. Figures 10(c) and 10(d) show the entropy sequence of the malware samples of family Adialer.C. It can be seen that the entropy sequences of the same malware family samples are very similar, but the entropy sequence distributions of different family samples are quite different.

3.4.2. Entropy Sequence Visualization. This paper visualizes the entropy sequence, which also involves the two critical issues mentioned above: first, how to convert malicious code data into the pixel data of image; second, how to set the image format. For the first issue, the value range of the entropy data is [0, 8], and the value range of image pixels is [0, 255]. Therefore, this paper amplifies the entropy value and processes each entropy value according to the following formula:

$$P_{h_i} = 2^{h_i} - 1, \tag{5}$$

where h_i represents the entropy value of the block i in the malware, P_{h_i} represents the enlarged value, and its value range is [0, 255], which is the same as the value range of image pixels. This paper uses P_{h_i} as the pixel value of the entropy image. For the second issue, this paper uses the grayscale visualization method to process the entropy sequence. The image format of the entropy sequence is shown in Table 2.

Moreover, the length of the entropy sequence of the malware is different, and the size of the visualization image is also different. To enhance the classification performance, this paper also standardizes the grayscale entropy image and modifies the image format to the same size. Some grayscale images of the entropy sequence are shown in Figure 11. The image size is unified to 224 ∗ 224. It can be seen that the entropy images of malware samples of the same family have strong similarities, but the images of different families have different features.

3.4.3. Entropy Grayscale Image Feature Extraction. Several scholars have achieved good performance in extracting grayscale image features based on deep learning algorithms. Therefore, this paper also extracts the features of the entropy grayscale images based on the deep learning algorithms. After several experiments, the ResNet50 deep learning network introduced in Section 3.2 could effectively extract the features of the entropy image. In this section, the features of the grayscale image of the entropy sequence are also extracted based on the ResNet50 deep learning network which is used as one of the input vectors of the final malicious code classifier.

We extract features from the byte sequence RGB image and the location information RGB image. Besides, we also extract the entropy features from the raw bytes of the

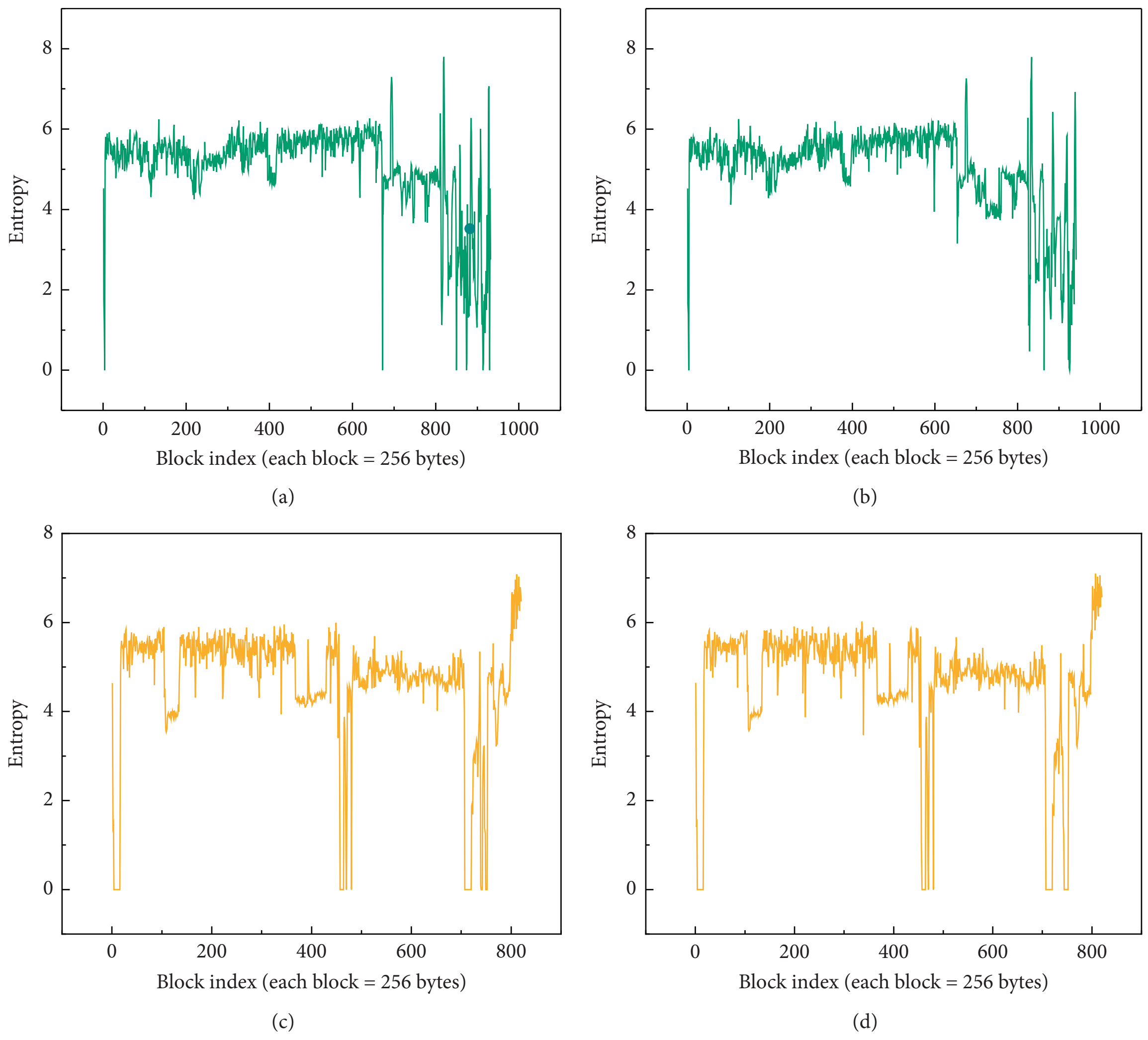


Figure 10: The entropy sequences of malware samples from different families. (a) Sample 1 of family Rbot!gen. (b) Sample 2 of family Rbot!gen. (c) Sample 3 of family Adialer.C. (d) Sample 4 of family Adialer.C.

Table 2: Entropy sequence image conversion width.

Sequence (KB)	Image width	Sequence (KB)	Image width
<10	32	100–200	384
10–30	64	200–500	512
30–60	128	0.500–1000	768
60–100	256	>1000	1024

malicious code. Finally, these three types of features are combined to train a machine learning model for malware classification.

4. Evaluation

4.1. Implementation and Setup. To evaluate the performance of the proposed method, we implemented a prototype system. The system is programmed in Python 2.7. The deep learning part of the system is programmed by the Keras library, and the machine learning part is programmed by the Sklearn library. The system is deployed on a PC with an Intel(R) Xeon(R) Gold 6139 CPU (2.3 GHz, 72 cores), a TITAN RTX graphics card and 188 GB RAM.

4.1.1. Dataset. To evaluate the proposed method, our experiments are conducted based on two malware datasets: malimg dataset [3] and Big 2015 dataset [43]. The malimg dataset consists of 9,339 malicious samples from 25 families. All the malicious samples are classified by the Microsoft security platform. The detailed distribution of the samples in

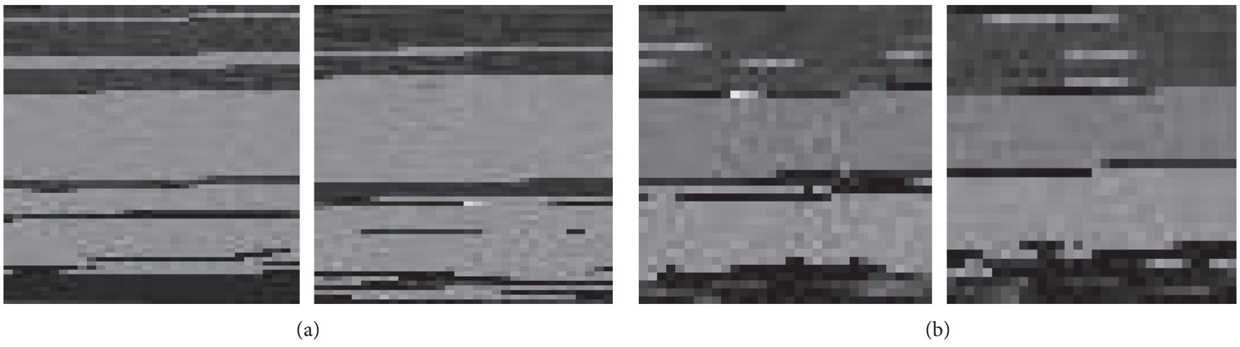


Figure 11: The examples of byte sequence RGB visualization of malware samples. (a) C2LOP.gen!g. (b) C2LOP.P.

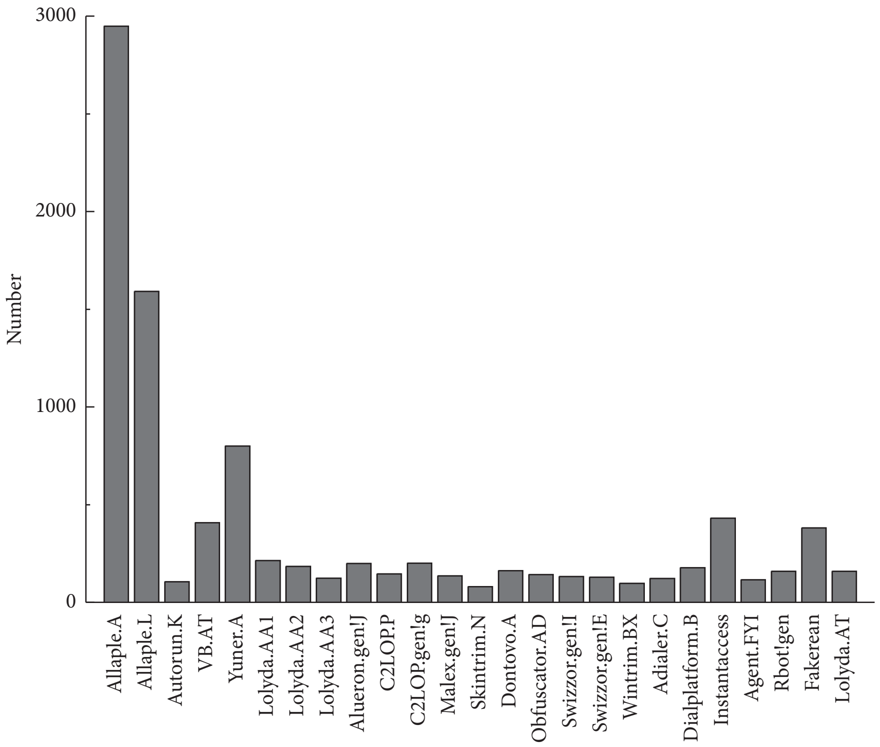


Figure 12: The distribution of samples in the dataset.

the dataset is shown in Figure 12. The big 2015 dataset is provided by the Microsoft Malware Classification Challenge (BIG 2015) [43]. The dataset consists of 10,868 samples from 9 families (Ramnit, Lolipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, and Gatak). All samples are provided by Microsoft Research and Northeastern University.

To ensure the accuracy and reliability of the experimental results, this paper uses a 10-fold cross-validation method. In the experiment, the dataset is divided into ten parts. In each test process, nine parts are selected as the training set, and the remaining one part is used as the test set. A total of ten experiments are conducted, and finally, the results of all experiments are combined to evaluate the method.

4.2. Evaluation Metric. The evaluation matric in this paper includes accuracy, precision, recall, receiver operating characteristic (ROC) curve, and area under ROC (AUC) curve. Accuracy: the proportion of the number of correct classification of malware samples to the entire samples. Precision: the ratio of true positive samples to the positive samples classified by the classifier. Recall: the proportion of true positive samples to entire positive samples in the dataset. F1-measure is the weighted harmonic average of precision and recall. ROC curve: the ordinate of the ROC curve is the true positive rate and the abscissa is the false positive rate. The true positive rate is also called recall. When evaluating the generalization ability of two classifiers, the area under the ROC curve (AUC) is usually compared. The size of the AUC area represents the generalization ability of the classifier. The related definition formulas are shown as follows:

$$\begin{aligned}\text{Acc} &= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}},\\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}},\\ \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}},\\ \frac{1}{F_1} &= \frac{1}{2}\times\left(\frac{1}{P}+\frac{1}{R}\right),\\ F_1 &= \frac{2\times\text{TP}}{2\times\text{TP}+\text{FP}+\text{FN}},\\ \text{FPR} &= \frac{\text{FP}}{\text{TN}+\text{FP}},\end{aligned}\tag{6}$$

where the true positive (TP) samples refer to the malware samples in which malicious code samples are correctly classified into their corresponding families. The false positive (FP) samples refer to the malware samples in which the samples are misclassified into the specified family. The true negative (TN) samples refer to the malware samples which do not belong to the specified family and are indeed classified into other families. The false negative (FN) samples refer to the malware samples of the specified family which are misclassified into other families.

4.3. *The Classification Performance of Three Kinds of Visualization Features.*

We first evaluated the classification performance based on different deep learning networks. Then, we extracted the visualization features and further evaluated the performance of three types of features based on different machine learning algorithms. The experiments in this section are conducted based on the malimg dataset.

4.3.1. *The Classification Performance of the Byte Sequence RGB Image Features.*

To evaluate the classification performance of different deep learning networks, we first extract malware byte-sequence visualization features based on nine deep learning networks (i.e., VGG16, VGG19, ResNet50, DenseNet, EfficientNet, InceptionResNet, InceptionV3, Xception, and NasNet). Then, we train the classifier based on the random forest algorithm. In the setting of each network, all the deep learning networks use the default configuration parameters in Keras. The classification performance is shown in Figure 13.

It can be seen that, except for the poor performance of the EfficientNet deep learning network, the other deep learning networks can extract the features of malicious code and achieve good classification performance. In the experiment, all networks remove the last fully connected layer, and the VGG16 and VGG19 networks also remove the pooling layer after the convolutional layer (if only the last

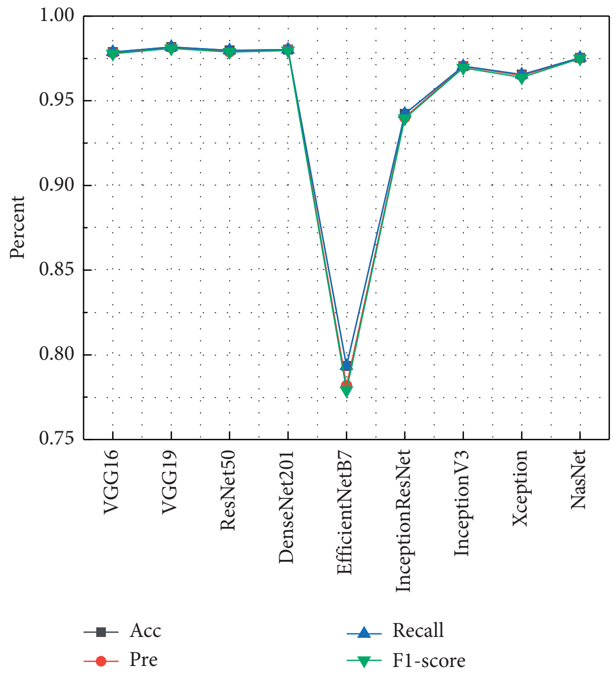


Figure 13: The performance of different deep learning networks trained based on byte-sequence features.

fully connected layer is removed, the performance of malware classification will be poor). However, this leads to a high dimension of image features extracted by the VGG16 and VGG19 networks (25088 and 100352, respectively). The experimental results of ResNet50 and DenseNet201 networks are almost the same. In this paper, the ResNet50 network is temporarily selected for extracting byte sequence level RGB image features.

We extract byte sequence level RGB image features based on the ResNet50 network. Then, we conduct experiments to evaluate the performance of different machine learning classifiers, which are trained based on random forest (RF), multilayer perceptron (MLP), k-nearest-neighbor (KNN), support vector machines (SVM), decision tree (DT), and Gaussian Naive Bayesian (NB) algorithms. Each sample has 2,048 features that are extracted by the ResNet50 network. In the setting of each classifier, the K value of the KNN classifier is set to 2, the SVM uses a linear kernel function, and the remaining classifiers use the default configuration parameters in SKlearn. The results are shown in Table 3.

From Table 3, we can see that the performance of the MLP classifier is relatively poor, which may be due to the default configuration. The classification performance could be improved by adjusting the parameters. KNN, RF, and SVM classifiers achieved relatively better classification performance, all achieving about 98% classification accuracy. In the experiment, the SVM classifier has the best classification performance, and the classification accuracy rate can reach 98.87%, but its training time is relatively long (i.e., about 60 s). However, the running time on the test set is about 4 seconds, and the efficiency is still relatively high, which proves that it is still suitable for large-scale malicious code classification.

Table 3: The performance of byte-sequence-feature-based classifiers.

Classifier	Training time (s)	Acc (%)	F1-score (%)	AUC
RF	2.64	97.97	97.87	0.9965
KNN	1.53	98.35	98.35	0.9890
MLP	157.90	90.36	88.91	0.9928
SVM	**61.90**	**98.87**	**98.84**	**0.9996**
DT	14.41	96.47	98.84	0.9559
NB	0.226	96.83	96.97	0.9813

The significance of the bold values given in the table is that, in the experiment, the SVM classifier achieved the best classification performance, and the classification accuracy rate can reach 98.87%.

4.3.2. The Classification Performance of the Location Information RGB Image Features. To evaluate the classification performance of different deep learning networks, we also extract malware location information visualization features based on nine different deep learning networks and then train the classifier based on the random forest algorithm. In the setting of each network, all the deep learning networks use the default configuration parameters in Keras. The classification performance is shown in Figure 14.

It can be seen that the classification performance of the classifiers trained based on the location information level RGB image features is worse than classifiers trained based on the byte sequence level RGB image features. However, the classification performance of the classifier trained based on the features extracted by the DenseNet201, InceptionV3, Xception, and NasNet networks could still reach 90%.

Inspired by the experiment in the previous section, we also evaluate the classification performance of the classifier trained based on these four types of features and the support vector machine (SVM) algorithm. Among them, the NasNet network extracts the features based on the linear kernel function, and the optimal solution cannot be trained. The performance of the other three classifiers is as follows: the accuracy of the DenseNet201 network features could reach 95.58%, the accuracy of the InceptionV3 feature could reach 94.67%, and the accuracy of the Xception features could reach 95.65%. It can be seen that the feature classification extracted by the Xception deep learning network has the best performance. Therefore, in this paper, the Xception network is temporarily selected for extracting location information level RGB image features.

We extract location information level RGB image features based on the Xception network. Then, we also conduct experiments to evaluate the performance of different machine learning classifiers. Each sample has 2,048 features that are extracted by the Xception network. In the setting of each classifier, the K value of the KNN classifier is set to 2, the SVM uses a linear kernel function, and the remaining classifiers use the default configuration parameters in SKlearn. The results are shown in Table 4.

From Table 4, we can see that the performance of the classifiers trained based on the location information level features is poor than the byte-sequence-feature-based classifiers. Among them, the NB classifier has the worst performance, and its accuracy is less than 70%. The classification performance of KNN, RF, and SVM classifiers is still relatively better than the others, all of which have achieved a classification accuracy of more than 90%. Among them, the SVM classifier has the best classification performance, and the classification accuracy can reach 95.65%. But in the experiment, the training time of the SVM classifier is still a bit long, but the running time on the test set is still about 4 seconds, and it still has practical application value.

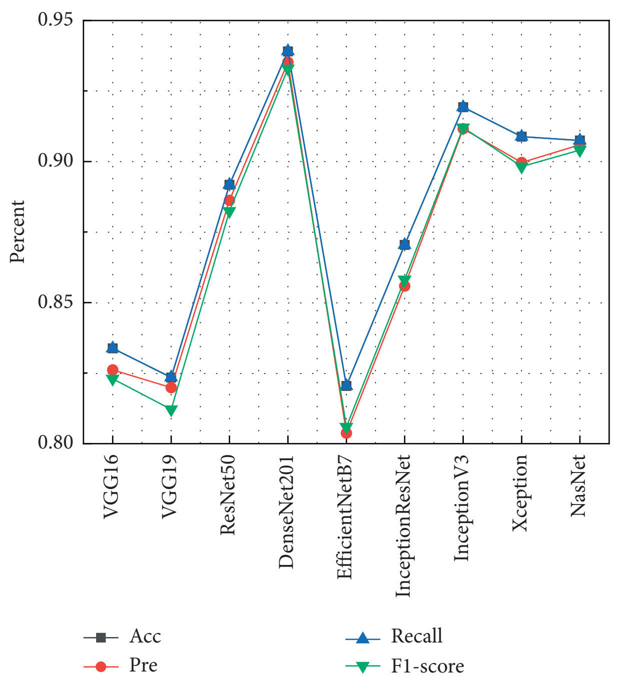


Figure 14: The performance of different deep learning networks trained based on location-information features.

Table 4: The performance of location-information-feature-based classifiers.

Classifier	Training time (s)	Acc (%)	F1-score (%)	AUC
RF	1.20	90.88	89.82	0.9843
KNN	1.61	91.91	91.78	0.9632
MLP	134.78	81.25	77.55	0.9835
SVM	**67.94**	**95.65**	**95.66**	**0.9989**
DT	10.19	86.28	86.35	0.9021
NB	0.10	69.70	68.12	0.9641

4.3.3. The Classification Performance of the Entropy Visualization Features. From the previous experiment, it can be seen that the classification performance of classifiers trained based on the features extracted by the ResNet50 network is all relatively good. Therefore, this paper directly extracts the entropy visualization features based on the ResNet50 network and conducts experiments to evaluate the performance of classifiers trained based on different machine learning algorithms. Each sample has 2,048 features that are extracted by the ResNet50 network. The setting of each classifier is the same as the previous experiment. The K value of the KNN classifier is set to 2, the SVM uses a linear kernel function, and the remaining classifiers use the default configuration parameters in SKlearn. The results are shown in Table 5.

Table 5: The performance of location-information-feature-based classifiers.

Classifier	Training time (s)	Acc (%)	F1-score (%)	AUC
RF	2.07	86.56	86.77	0.9935
KNN	1.43	86.00	85.94	0.9828
MLP	91.62	82.79	81.99	0.9917
SVM	**144.27**	**86.71**	**86.82**	**0.9957**
DT	15.03	84.42	84.35	0.9514
NB	0.21	85.57	85.82	0.9707

Table 6: The performance of combined-feature-based classifiers.

Classifier	Training time (s)	Acc (%)	F1-score (%)	AUC
RF	2.36	99.15	99.14	0.9997
KNN	7.25	99.10	99.09	0.9917
MLP	118.56	88.03	85.39	0.9902
SVM	**120.02**	**99.73**	**99.73**	**0.9999**
DT	26.55	98.13	98.11	0.9765
NB	0.64	98.68	98.68	0.9895

From Table 5, we can see that the performance of the classifiers trained based on the entropy visualization feature is poor than the previous two feature-based classifiers. The classification accuracy of each classifier is about 85%, among which the KNN, RF, and SVM classifiers still have a relatively better classification performance, and the classification accuracy could reach 86%. The classification performance of the SVM classifier is still the best, and the classification accuracy can reach 86.71%. Analyzing the experimental results, the reason for the decrease in classification accuracy is mainly due to the poor classification effect for similar families such as Allaple.L and Allaple.A. However, it has achieved a better classification performance for the C2LOP.P and C2LOP.gen!g and Swizzor.gen!j and Swizzor.gen!I families, which is somewhat different from the classification effect of the two types of features as mentioned above.

4.4. The Classification Performance of the Combined Features. Based on visualization technology, this paper extracts three types of features of malware: byte-sequence features, location-information features, and entropy visualization features. The length of the three types of features is 2,048. To evaluate the impact of the three types of features on the classification and whether the combination of the three types of features can help improve the experimental performance, this paper first trains different machine learning classifiers based on the combined features, with the same parameter settings as the previous experiment. The classification performance is shown in Table 6.

It can be seen from Table 6 that the classifier based on combined features performs very well. Except that the parameter settings of the MLP classifier may have problems, resulting in its poor classification performance, the other classifiers all achieved acceptable performance. Among them, the classification accuracy of the RF, KNN, and SVM classifiers have reached more than 99%, and the classification accuracy of the SVM classifier even could reach 99.73%. To further compare the classification performance of each type of feature and the combined feature, this paper conducts more experiments. The result is shown in Table 7.

In Table 7, RGB1 refers to byte-sequence features, RGB2 refers to location information level features, and Ent_gray refers to entropy grayscale image features. R1, R2, and Ent are the abbreviations for the three types of features, respectively. From the results, we can see that the classifiers trained based on three kinds of features all can achieve good malicious code classification performance, and the classification accuracy of the classifier trained based on byte sequence level features can reach 98.865%. Therefore, we first evaluated the performance of the classifiers based on the byte-sequence features combined with other two kinds of features separately and then evaluated the classification performance based on the combination of three types of features. From the experimental results, we can see that both the location information visualization feature and the entropy feature can improve the classification performance.

Moreover, the classification accuracy of the classifier trained based on the combination of three types of features has been further improved, and the classification accuracy can reach 99.732%. It can be seen that the three types of features all contribute to the final classification performance. To further evaluate the impact of different features on the classification performance, this paper compares the classification performance of different classifiers in each malware family, and the result is shown in Figure 15.

It can be seen from Figure 15 that among the three types of features, the classification performance of the byte-sequence-feature-based classifier is relatively better, but its classification performance on C2LOP.P and C2LOP.gen!g and Swizzor.gen!E and Swizzor.gen!I families is worse than the other two features. However, the classification performance of the combined features is best. Analyzing the misclassified samples, similar family samples still have similar characteristics, resulting in relatively low classification performance in the Swizzor.gen!I and C2LOP.P families. This issue needs further research in the future. Overall, this paper solves most of the sample classification confusion problems of similar families by combining features, and the application of combined features has better classification performance and practical value than single features.

4.5. The Classification Performance of Different Image Formats. The format of the image influences the texture characteristics of the image. In this section, we conduct experiments to evaluate the influence of image format. The malicious code classification method studied in this paper is developed on the traditional method of extracting image GIST and other texture features. It is difficult to process malicious code images of different formats based on the deep learning network. Therefore, we extract the GIST texture features of the malicious code images for classification to evaluate the impact of the changes in the image format on the malicious code classification performance. In the experiment, the format of malicious code images is set to 8 different formats (original, $32*32$, $64*64$, $128*128$,

Table 7: The performance of classifiers trained based on different visualization features.

Features	Acc (%)	Pre (%)	Recall (%)	F1-score (%)
RGB1	98.865	98.870	98.864	98.847
RGB2	95.653	95.755	95.655	95.662
Ent_gray	86.712	87.127	86.713	86.821
R1 + R2	99.539	99.560	99.539	99.537
R1 + Ent	99.615	99.634	99.614	99.609
R1 + R2 + Ent	**99.732**	**99.743**	**99.733**	**99.730**

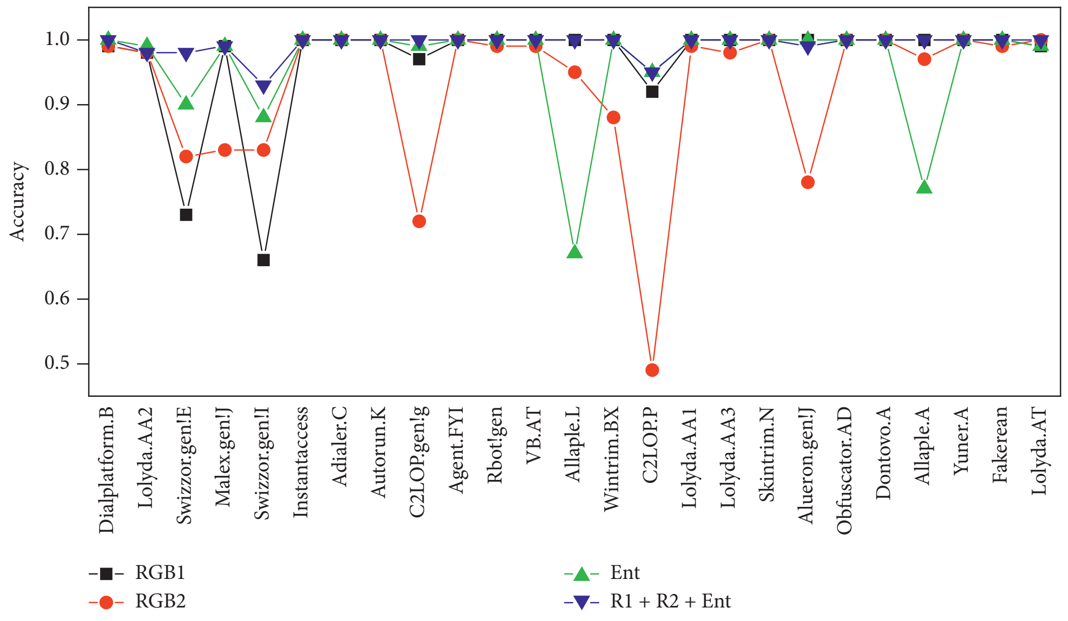


Figure 15: The performance of different visualization features on each family.

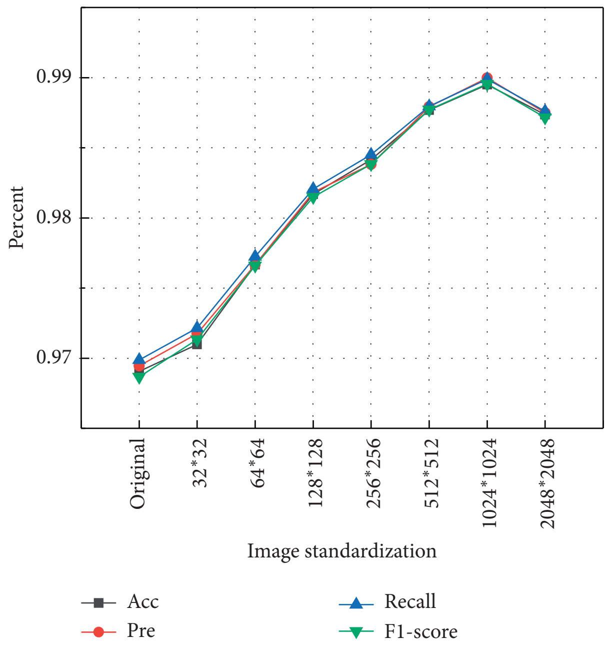


Figure 16: The performance of different image formats.

256 ∗ 256, 512 ∗ 512, 1, 024 ∗ 1, 024, and 2, 048 ∗ 2, 048). The experimental results are shown in Figure 16.

It can be seen from Figure 16 that the uniform and standardized formats of malicious code images can improve the accuracy of malicious code detection. The reason may be that the sizes of malicious codes in the same family are different, resulting in different sizes of converted images. In this way, there are some differences in the extracted image features, which may cause some confusion in the classification of malicious codes in different families. After the image format is unified, the difference in characteristics of malicious codes of the same family is reduced, and the classification performance is improved. At the same time, as shown in the figure, the classification performance is best when the malicious code image format is set to 1, 024 ∗ 1, 024. However, the image format is modified to a larger size, and the more time it takes to extract image features. When the image format is set to 256 ∗ 256, the feature extraction time of the entire dataset is about 1-2 minutes, and when the image is set to 1, 024 ∗ 1, 024, the experimental time for extracting GIST texture features is more than half an hour. Although the classification accuracy has been improved, the efficiency is too low and needs to be improved in practical applications. Therefore, in our method, we resize all the malware images into a uniform size, which has improved the classification performance.

4.6. Compared with Other Malware Classification Methods. The classification technology based on the malware visualization features is a hot spot in recent research, and there have been many significant research results. To evaluate the proposed method, we compare the method with four famous methods proposed in related research in recent years (GIST + KNN [3], LBP + KNN [25], GIST + DSIFT + KNN [6], and VGG16 (fine-tune) [28]) on the malimg dataset. The experimental results are shown in Table 8. It can be seen that the method proposed in this paper has the best classification performance.

Naeem et al. [6] proposed a new method (GIST + DSIFT) to extract more complex features and improved classification performance. Besides, Rezende et al. [28] proposed a method (VGG16 (fine-tune)) to extract image features based on deep learning and transfer learning algorithms, achieving a better performance. These methods partially alleviate the problem of malicious code confusion in similar families. However, it can be seen that this paper has achieved the best classification accuracy based on three types of visualization methods and almost solved the problem of malware confusion of similar families in the malimg dataset. However, for the efficiency, due to the large number of features proposed in this paper, the running time of the proposed method is a bit longer. The training time of the experiment is about 120 seconds and the running time on the test set is about 7 seconds, but it is still applicable to the actual environment and can be used to detect large-scale malicious code samples. To sum up, the method proposed in this paper has achieved a good classification performance on the malimg dataset, showing its superiority to the other methods.

All the previous experiments were performed on the malimg dataset. To further evaluate the performance of the proposed method, we also conducted experiments to compare our method with the other four methods on the big 2015 dataset. The experimental results are shown in Table 9. It can be seen from Table 9 that the method proposed in this paper still shows the best classification performance. The training time on the Big2015 dataset is about 195.85 seconds, and the running time is about 7 seconds. It is still applicable to the actual environment.

In the experiment, VGG16 (fine-tune) has a better classification performance than the other three methods, which proves that the deep learning algorithm can extract better features in different datasets. The three types of features proposed in this paper describe the malware samples more detailed and achieve better classification performance. We have conducted further research on samples that were misclassified by this method. The proposed method only has a classification accuracy of 88% for Simda family samples, while the classification accuracy of other families is almost 100%. It could be found that the proposed method misclassifies some malware samples of the Simda family into Vundo and Obfuscator.ACY families. The misclassified samples have similar visual features with the samples of the two families, which leads to the misclassification. In future research, more in-depth research will be conducted on this issue.

Table 8: Proposed method compared with other malware classification methods on the malimg dataset.

Methods	Acc (%)	Recall (%)	F_1(%)	RT (s)
GIST + KNN [3]	97.28	97.28	96.61	0.423
LBP + KNN [25]	97.87	97.88	97.86	0.052
GIST + DSIFT + RF [6]	98.53	98.53	98.49	0.026
VGG (fine-tune) [28]	98.84	98.84	98.82	59.50
Our method	**99.73**	**99.73**	**99.73**	**7.625**

RT means running time.

Table 9: Proposed method compared with other malware classification methods on the Big2015 dataset.

Methods	Acc (%)	Recall (%)	F_1 (%)	RT (s)
GIST + KNN [3]	94.82	94.81	94.75	0.621
LBP + KNN [25]	91.87	91.88	91.87	0.045
GIST + DSIFT + RF [6]	95.09	95.10	95.06	0.004
VGG16 (fine-tune) [28]	97.92	97.93	97.92	62.59
Our method	**99.54**	**99.53**	**99.53**	**7.91**

RT means running time.

Moreover, to further evaluate the classification performance of the proposed method, we deeply compared the classification performance with the method proposed by Cui et al. [4] and Rezende et al. [28] on the malimg dataset for each family. The result is shown in Figure 17. The two methods proposed by Cui et al. [4] and Rezende et al. [28] used the raw bytes of malware and deep learning algorithms to achieve malicious code classification. Cui et al. [4] pointed out that training the classifier directly based on the deep learning algorithm and the malimg dataset is not a well solution, because the samples of some families in the malimg dataset is not enough for training, resulting in low classification accuracy. To improve the performance of the classifier, they fine-tuned the malware samples in the dataset. and Rezende et al. [28] trained the malware classifier based on the transfer learning algorithm and also achieved good classification performance.

The misclassification problem of similar families has always been a relatively difficult problem to solve in related research. In the malimg dataset, several methods have not been able to solve the misclassification problem of Swizzor.gen!E family and Swizzor.gen!I family, C2LOP.P family, and C2LOP.gen!g family virtually. As shown in Figure 17, the classification performance of the method proposed by Cui et al. [4] on the four families is not good. The VGG16 (fine-tune) method [28] achieved a good classification performance on the C2LOP.P and C2LOP.gen!g families, but the classification accuracy of the samples in the Swizzor.gen!E family and Swizzor.gen!I family still could not reach 80%. It still fails to solve the problem of sample confusion between these two families.

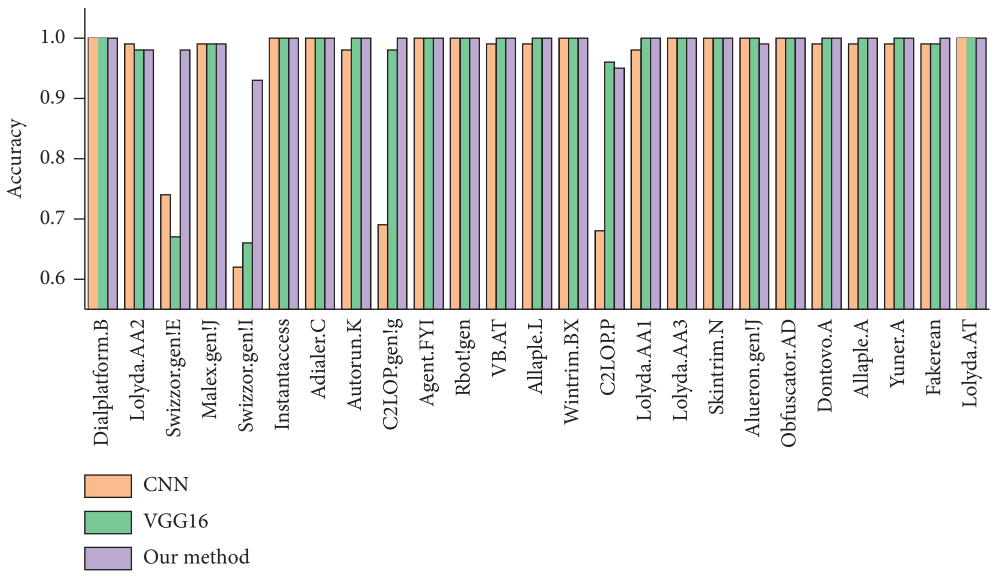


Figure 17: Comparison with other methods on each malware family.

However, the proposed method has achieved good classification performance in all these four families, and the classification accuracy of each family could reach 90%. The experimental results further prove that the proposed method has a better classification performance than other methods, especially for some similar family samples, which can achieve higher classification accuracy.

5. Limitation and Discussion

In this paper, a ten-fold cross-validation experiment is performed on the experimental dataset. From the experimental results, the proposed method can achieve better malware classification performance. There are two main reasons why our method can achieve better classification performance. The first reason is that we use different deep networks. After a large number of experiments, these new types of deep learning networks can achieve better performance based on these three types of features. The second reason is that we have proposed new malicious code features. Based on the related research, we adopted a variety of malicious code data representation methods and extracted a variety of features, including RGB and entropy features. These features can more accurately describe the malicious code to help achieve a better classification performance. For the entropy features, the entropy visualization method proposed in this paper could improve the final classification performance. However, the classification performance of the entropy feature classifier still needs to be improved. In future work, it is necessary to apply more methods to achieve higher classification performance. Besides, the proposed method extracts the features mainly based on the raw byte of malicious code, and it still has some limitations. In future research, the proposed method can be used combined with different kinds of methods to solve various problems encountered in practical applications.

The combination of these three types of features effectively improves the performance of malicious code classification, and the selection of different deep learning networks further improves the classification performance. However, how to quickly find the relatively optimal deep learning network according to the characteristics of the dataset is indeed worthy of in-depth study. In future work, we will conduct further research on the settings of the deep network and image format to propose new malicious code classification methods.

6. Conclusions and Future Work

This paper proposed a new malware classification method based on multiple visualization features and the transfer algorithm. First, the raw byte of malware is converted into byte sequence RGB image, location information RGB image, and entropy grayscale image, respectively. Then, three types of features are extracted based on different deep learning networks. Finally, the three types of features are combined to train a classifier. We implement the method and evaluate it on two custom datasets with a total of more than 20,000 samples provided by the Malware Research Lab and Microsoft Research. The experimental results show that the proposed method can effectively facilitate us in distinguishing malware variants of similar families, and the values of classification accuracy on the two datasets all could reach 99.5%, showing its superiority over other methods. Moreover, this paper also proves that we could solve the misclassification problem of malware samples from similar families based on visual features. The visual characteristics of malware are of great significance to further improve the performance of malware classification in the future.

In future work, we would further explore the entropy feature extraction method and propose better malicious code classification methods. Besides, with the rapid development of deep learning algorithms and natural language processing algorithms, we would conduct in-depth research on how to improve the classification performance based on new algorithms.

Data Availability

The malware samples used to support the finding of this study are available at DOI: 10.1145/2016904.2016908 and BIG 2015 (available at https://ww.kaggle.com/c/malware-classification). These prior studies (and datasets) are cited at relevant places within the text as references [3, 43].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors are very thankful to Zhi-hao Yan, Zhi-jie Xie, Xuan-zhen Guo, Hang Zhou, Rui-peng Wang, and Yuan-chao Chen for their help in the preparation of experiment and paper review. This research was funded by the Laboratory of Network Security, College of Electronic Engineering, National University of Defense Technology, Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, Hefei 230037, China, and the Natural Science Foundation of Anhui Provincial (grant number 1908085QF291).

References

[1] SYMANTEC, "Internet security threat report," May 2020, https://www.broadcom.com/support/security-center/publications/threat-report.

[2] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[3] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pp. 1–7, Pittsburgh, PA, USA, July 2011.

[4] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-g. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[5] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pp. 21–30, Chicago, IL, USA, October 2011.

[6] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, and M. S. Javed, "Identification of malicious code variants based on image visualization," *Computers & Electrical Engineering*, vol. 76, pp. 225–237, 2019.

[7] Z. Tang, P. Wang, and J. Wang, "Convprotonet: deep prototype induction towards better class representation for few-shot malware classification," *Applied Sciences*, vol. 10, no. 8, p. 2847, 2020.

[8] M. Wojnowicz, G. Chisholm, M. Wolff, and X. Zhao, "Wavelet decomposition of software entropy reveals symptoms of malicious code," *Journal of Innovation in Digital Ecosystems*, vol. 3, no. 2, pp. 130–140, 2016.

[9] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M.-S. Choi, "Entropy analysis to classify unknown packing algorithms for malware detection," *International Journal of Information Security*, vol. 16, no. 3, pp. 227–248, 2017.

[10] L. Liu, X. He, L. Liu, L. Qing, Y. Fang, and J. Liu, "Capturing the symptoms of malicious code in electronic documents by file's entropy signal combined with machine learning," *Applied Soft Computing*, vol. 82, Article ID 105598, 2019.

[11] G. Canfora, F. Mercaldo, and C. A. Visaggio, "An hmm and structural entropy based detector for android malware: an empirical study," *Computers & Security*, vol. 61, pp. 1–18, 2016.

[12] S. S. W. Piyanuntcharatsr, S. Adulkasem, and C. Chantrapornchai, "On the comparison of malware detection methods using data mining with two feature sets," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 293–318, 2015.

[13] Z. Feng, S. Xiong, D. Cao et al., "A hybrid framework for malware detection," in *Proceedings of the 2015 ACM International Workshop on International Workshop on Security and Privacy Analytics*, pp. 19–26, San Antonio, TX, USA, March 2015.

[14] E. Raff and C. Nicholas, "An alternative to ncd for large sequences, lempel-ziv jaccard distance," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1007–1015, Halifax, Canada, August 2017.

[15] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1357–1365, Chicago, IL, USA, August 2013.

[16] J. Upchurch and X. Zhou, "Variant: a malware similarity testing framework," in *Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 31–39, IEEE, Fajardo, PR, USA, October 2015.

[17] N. Kawaguchi and K. Omote, "Malware function classification using apis in initial behavior," in *Proceedings of the 2015 10th Asia Joint Conference on Information Security*, pp. 138–144, IEEE, Kaohsiung City, Taiwan, May 2015.

[18] P. Vadrevu and R. Perdisci, "Maxs: scaling malware execution with sequential multi-hypothesis testing," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 771–782, Xi'an, China, May-June 2016.

[19] H. Kim, J. Kim, Y. Kim, I. Kim, K. J. Kim, and H. Kim, "Improvement of malware detection and classification using api call sequence alignment and visualization," *Cluster Computing*, vol. 22, no. 1, pp. 921–929, 2019.

[20] Y. Dai, H. Li, Y. Qian, R. Yang, and M. Zheng, "Smash: a malware detection method based on multi-feature ensemble learning," *IEEE Access*, vol. 7, pp. 112 588–112 597, 2019.

[21] C.-T. Lin, N.-J. Wang, H. Xiao, and C. Eckert, "Feature selection and extraction for malware classification," *Journal of Information Science and Engineering*, vol. 31, no. 3, pp. 965–992, 2015.

[22] K. Kosmidis and C. Kalloniatis, "Machine learning and images for malware detection and clxassification," in *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, pp. 1–6, Larissa Greece, September 2017.

[23] H. Naeem, B. Guo, F. Ullah, and M. R. Naeem, "A cross-platform malware variant classification based on image representation," *KSII Transactions on Internet & Information Systems*, vol. 13, no. 7, 2019.

[24] B. Xiaofang, C. Li, H. Weihua, and W. Qu, “Malware variant detection using similarity search over content fingerprint,” in *Proceedings of the the 26th Chinese Control and Decision Conference (2014 CCDC)*, pp. 5334–5339, IEEE, Changsha, China, May-June 2014.

[25] H. Hashemi and A. Hamzeh, “Visual malware detection using local malicious pattern,” *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 1–14, 2019.

[26] J. Zhang, Z. Qin, H. Yin, L. Ou, S. Xiao, and Y. Hu, “Malware variant detection using opcode image recognition with small training sets,” in *Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, IEEE, Waikoloa, HI, USA, August 2016.

[27] K. Han, B. Kang, and E. G. Im, “Malware analysis using visualized image matrices,” *The Scientific World Journal*, vol. 2014, p. 15, Article ID 132713, 2014.

[28] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. de Geus, “Malicious software classification using vgg16 deep neural network’s bottleneck features,” in *Information Technology-New Generations*, pp. 51–59, Springer, Berlin, Germany, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[30] F. Chollet, “Xception: deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, Honolulu, HI, USA, July 2017.

[31] C. Szegedy, W. Liu, Y. Jia et al., “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Boston, MA, USA, June 2015.

[32] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” 2015, https://arxiv.org/abs/1502.03167.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, Las Vegas, NV, USA, June-July 2016.

[34] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” 2016, https://arxiv.org/abs/1602.07261.

[35] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, https://arxiv.org/abs/1409.1556.

[36] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[37] S. Hochreiter, “Untersuchungen zu dynamischen neuronalen netzen,” *Diploma, Technische Universität München*, vol. 91, no. 1, 1991.

[38] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, “Aggregating local image descriptors into compact codes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2011.

[39] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, Sardinia, Italy, May 2010.

[40] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, Santiago, Chile, December 2015.

[41] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353–5360, Boston, MA, USA, June 2015.

[42] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, June 2010.

[43] Microsoft, “Microsoft malware classification challenge (big 2015),” June 2020, https://www.kaggle.com/c/malware-classification.