Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3301718, 19 pages
https://doi.org/10.1155/2022/3301718

Research Article

Malicious Code Classification Method Based on Deep Residual Network and Hybrid Attention Mechanism for Edge Security

Yanli Shao, Yang Lu, Dan Wei, Jinglong Fang, Feiwei Qin, and Bin Chen

Key Laboratory of Complex Systems Modeling and Simulation, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Correspondence should be addressed to Dan Wei; weiwd@hdu.edu.cn

Received 17 July 2021; Revised 8 June 2022; Accepted 17 June 2022; Published 7 July 2022

Academic Editor: Chakchai So-In

Edge computing is a feasible solution for effectively collecting and processing data in industrial Internet of Things (IIoT) systems, and edge security is an important guarantee for edge computing. Fast and accurate classification of malicious code in the whole lift cycle of edge computing is of great significance, which can effectively prevent malicious code from attacking wireless sensor networks and ensure the stable and secure transmission of data in smart devices. Considering that there is a large amount of code reuse in the same malicious code family, making their visual feature similar, many studies use visualization technology to assist malicious code classification. However, traditional malicious code visual classification schemes have the problems such as single image source, weak ability of deep-level feature extraction, and lack of attention to key image details. Therefore, an innovative malicious code visual classification method based on a deep residual network and hybrid attention mechanism for edge security is proposed in this study. Firstly, the malicious code visualization scheme integrates the bytecode file and assembly file of the malware and converts them into a four-channel RGBA image to fully represent malicious code feature information without increasing the computational complexity. Secondly, a hybrid attention mechanism is introduced into the deep residual network to construct an effective classification model, which extracts image texture features of malicious code from two dimensions of the channel and spatial to improve the classification performance. Finally, the experimental results on the BIG2015 and Malimg datasets show that the proposed scheme is feasible and effective and can be widely applied used in various malicious code classification issues, and the classification accuracy rate is relatively higher than the existing better-performing malicious code classification methods.

1. Introduction

In recent years, the fast expansion of the Internet of Things (IoT) has led to the industrial IoT (IIoT). Edge computing as a feasible solution for efficient collection and processing of data in IIoT has received great attention from academia, industry, and government departments and has been widely used in industries such as power, transportation, manufacturing, and smart cities. With the continuous deepening of the digital transformation process of the industry, the evolution of the edge computing network architecture will inevitably lead to an increasing number of security attacks on edge computing nodes, and edge security issues have become one of the obstacles restricting the development of the edge computing industry [1]. Nowadays, a large number of smart devices and sensors constitute a large wireless sensing network that can monitor, sense, and collect information from various monitored objects, while computing and storing these massive data. However, malware running on these ubiquitous sensors and smart devices can affect data security and cause other potential threats to data and IIoT devices [2]. Currently, the rapid increase in the types and quantity of malicious code not only brings property and economic losses but also gradually threatens national security [3]. For example, in May 2017, a computer ransomware called WannaCry spreads in more than 100 countries around the world. Many universities were infected and severely spreads to large public service areas such as airports, customs, and public safety networks [4]. In view of the weak security protection mechanism and limited computing resources of edge computing nodes, the detection and prevention of malicious code in the entire life cycle of edge computing is of great significance [5]. Malicious code classification is the key to preventing malicious code from running and improving information security and provides an important basis for malicious code detection, control, and removal.

Despite the continuous advancement of malicious code detection and classification technologies, malicious code has continuously evolved to generate new variants to avoid detection and quickly copied and spread, resulting in frequent security incidents in recent years. In most cases, the malicious code is generated or improved in an automated or semiautomated manner, and its core modules are reused during the generation process. Since the vast majority of new malicious codes are derived from the known malicious code mutations, there are generally less than 2% code differences between malicious codes of the same family [6]. This provides information security researchers with the basis for malicious code classification, that is, the detection and classification of different malicious code families can be achieved through visual feature similarity detection of malicious code core modules. The current mainstream malicious code detection and classification methods mainly include static analysis methods [7] and dynamic analysis methods [8]. The former refers to the analysis of malicious code without executing binary programs, which often fails to effectively solve the impact of packing and obfuscation technologies, while the latter refers to the use of program debugging tools to track and observe malicious code when it is executed and to verify the static analysis results according to the working process of the malware. This method is often inefficient and has a single execution path when dealing with large amounts of malicious code. Limited by computing power and resource consumption, traditional solutions perform poorly invariant similarity analysis of large-scale malicious code family samples. With the rapid development of deep learning technology and the increase in types and quantities of malicious code, researchers gradually began to convert the malicious code classification problem into an image classification problem. Malicious code visualization scheme based on deep learning has become a current research hotspot [9–11].

Malicious codes of the same family have similarities in visual features, but different families are different, which can be used as the basis for malicious code detection and classification. Thus, a malicious code visualization scheme transforms the problem of malicious code classification into an image classification problem and applies deep learning technology to solve it. Since different malicious code images reflect the differences in code data structure and information volume, the generation method of malicious code images is very important for malicious code classification. Currently, most of the existing malicious code visualization schemes only use bytecode files or assembly files and convert them into grayscale or RGB images for classification [10]. Some visualization schemes choose to calculate information entropy to enhance image information to further improve classification accuracy [11]. However, these methods have problems such as the single source of malicious code images and the large computational complexity of enhanced information, which increase the classification difficulty and reduce the classification accuracy to a certain extent. In addition, malicious code often exists in the local location of the program, manifesting as local image features. In malicious code visualization scheme, the commonly used convolutional neural network (CNN) pays more attention to the global image features and does not consider the detailed image features of the key regions. Therefore, it is necessary to introduce an attention mechanism to assign different weights to different regions in the image, so that the neural network can fully exploit and utilize the local detailed feature information of the malicious code image. In this way, the key image feature information is extracted through the attention mechanism, thereby improving the accuracy of subsequent malicious code detection and classification.

Based on the above analysis, this study proposes an innovative malicious code visualization classification method to further improve the classification accuracy and efficiency and then supports the detection and prevention of malicious code in edge computing. On the one hand, this scheme uses both the bytecode file and assembly file of the malware to visualize the malicious code as an RGBA image without additional calculation of code information entropy, which makes up for the defects of a single source of malicious code image information, insignificant image features, and excessive calculation. On the other hand, the hybrid attention mechanism is combined with the deep residual network to build a more accurate classification model. The deep residual network improves the classification accuracy while using shortcut connections to alleviate the gradient disappearance problem, accelerate model convergence, and improve the model's discriminative ability. Especially, each residual unit adopts a hybrid attention mechanism to extract more critical deep features from the two dimensions of channel and spatial to further improve the classification accuracy.

This study is organized as follows: Section 2 reviews the related work on malicious code classification. Section 3 introduces the core method, showing the detailed implementation of the proposed malicious code classification method from the malicious code visualization module and classification module, respectively. Section 4 presents the related experimental verification and performance analysis. The last section is the conclusion and future work.

2. Related Work

As mentioned above, edge security is an important guarantee for edge computing, wherein malicious code detection and prevention in the entire life cycle of edge computing is of great significance [1]. At present, malicious code visualization schemes have been developed on the basis of static analysis and dynamic analysis. Researchers have conducted extensive exploration and research on classification methods based on malicious code visualization. The key to improving the classification accuracy lies in how to extract reasonable and effective feature images to represent the program features of original malicious code as much as possible.

Conti et al. [12] pointed out that the malicious sample visualization method can help security analysts to quickly identify malicious code files. On this basis, Nataraj et al. [13] proposed a complete visual classification scheme for malicious samples, which mapped the malicious code to a grayscale image and extracted GIST features from it and finally implemented the malicious code classification through the K nearest neighbor (KNN) algorithm. They [14] also pointed out that the malicious code images of the same family have similar texture features, while the texture features of malicious code images of different families are quite different. Kornish et al. [15] found that appropriate improvements to images can improve the malware classification accuracy. Since then, malicious code visualization schemes have been enriched, the source of image information was no longer limited to bytecode files, and RGB images [16, 17] and RGBA images [18] were widely used. Wang et al. [16] divided the binary sequence of the malicious code file into RGB three-color channel values and converted the malicious sample into RGB images. Meanwhile, Sun et al. [17] used ASCII character information and PE structure information to convert malicious samples into RGB images and used VGG16 model to train and predict malicious code images. Chen et al. [18] used the bytecode file and local information entropy to convert the malware into RGBA images with larger information capacity, but this scheme increases the amount of calculation, and the image information source is single. These malicious code visualization schemes based on image features make up for the shortcomings of static analysis methods that are difficult to solve the problem of sample packing and confusion, as well as the long feature extraction time of dynamic analysis methods. Most of the aforementioned visualization schemes still follow Nataraj's grayscale scheme, using only bytecode files or assembly files, converting them to grayscale or RGB images for classification, or choosing to calculate information entropy to enhance image information to improve the classification accuracy. However, there are still the problems of single source of code images and high computational complexity. The feature information of malicious code images is not fully utilized, and the classification accuracy and efficiency still need to be improved.

As mentioned before, deep learning technology has power feature learning and expression ability, which makes it has outstanding advantages in extracting global features and contextual information of images. Currently, deep learning technology is widely used in various classification and prediction problems in different fields, such as hyperspectral image classification, IIoT security, and malicious code classification and detection [19–22]. Cheng et al. [9] explored an ensemble interpretable framework for automatic and efficient malicious code detection based on the knowledge graph of malware. Peng and Lu [23] proposed a discriminative extreme learning machine with supervised sparsity preserving (SPELM) model and verified the effectiveness of this model on four widely used image benchmark datasets. Pitolli et al. [24] proposed a novel approach for malware family identification based on an online clustering algorithm, which efficiently updates clusters as new samples are fed without rescanning the entire dataset. Cakir and Dogdu [25] used a shallow neural network based on the Word2Vec vector space model to represent the malicious code and finally applied the gradient search algorithm to classify the malicious code. Turnip et al. [26] proposed the eXtreme Gradient Boosting (XGBoost) to identify Android malware types. Liu et al. [27] combined graph neural networks with expert knowledge to realize smart contract vulnerability detection. Choi [28] proposed a malicious PowerShell detection method using GCN, which increased the detection rate of malicious PowerShell by approximately 8.2%. Wu et al. [29] proposed an attack-agnostic method based on cascaded self-supervised learning models [30] and achieved effective defense performance. With the development of the IIoT technology [31, 32], more and more users are beginning to use smart mobile terminal devices. Jaigirdar et al. [33] proposed the Prov-IoT model to maintain the data security of IoT devices. Zhou et al. [34] proposed a security defense system to protect the security of intelligent systems. However, the Android system is often attacked by malware due to its open source. Multimodal deep learning (MDL) performs well in complex scenes chosen to detect Android Malware by Kim et al. [35] and Vasu and Pari [36]. Ghouti and Imam [37] used principal component analysis (PCA) to extract the category and structural features of the malicious code and then used an optimized SVM to achieve malicious code classification. However, due to the structural characteristics of deep neural networks, such as focusing on global features and ignoring local details, some emerging research needs to be introduced to compensate for structural defects to comprehensively extract malicious code features and further improve the classification accuracy.

Since using the attention module in the CNN can focus on key information and improve the representation ability of convolution [38], more and more researchers [39] introduce attention mechanism into the field of malicious code classification and detection. Yakura et al. [40] built an ACNN malicious code detection model by combining CNN and attention mechanism to reduce the workload of analysts. Wang et al. [41] proposed a Depthwise Efficient Attention Module (DEAM) and combined it with a DenseNet to propose a new malware detection and family classification model. However, these schemes did not conduct in-depth research on the classification of malicious code families; there is still a huge potential research space for the application research of attention mechanism in malicious code visualization-based classification schemes.

3. Malicious Code Visual Classification Method

3.1. Method Overview. In order to solve the abovementioned problems in existing malicious code classification methods, this study proposes a malicious code visualization classification method based on a deep residual network and hybrid attention mechanism to achieve the accurate and efficient classification of malicious code. The overall flowchart is shown in Figure 1, and the details are as follows:

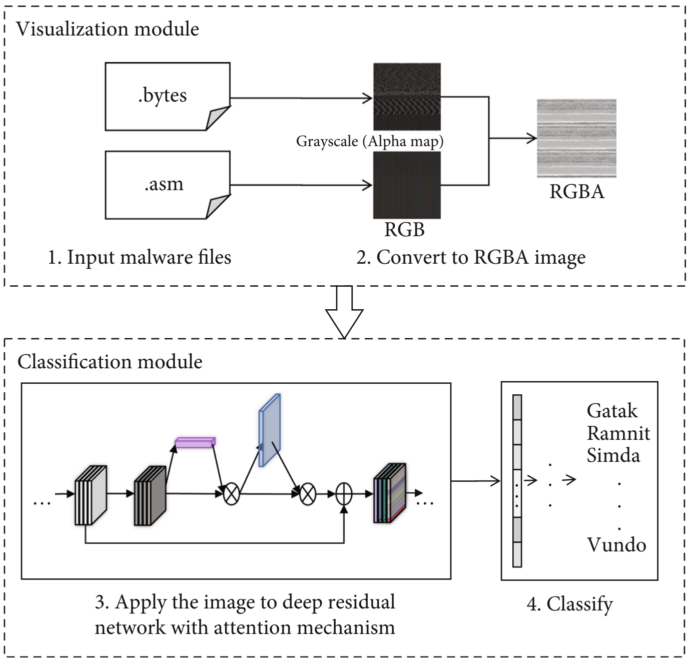


Figure 1: Method overview

(1) Malicious code visualization module: to compensate for the single source of malicious code image information, insignificant image features, and excessive computational complexity, a malicious code visualization scheme is proposed, which combines malware bytecode files and assembly files to form the RGBA images to enhance image information. This method converts the bytecode file into a grayscale image with a specified pixel size and also converts the assembly file into an RGB image of the same size to facilitate subsequent image fusion. Then, the value of the grayscale image as the transparency channel value is merged with the RGB image to form an RGBA image, so as to realize the visualization of malicious code while enhancing the effective information of the image without increasing the complexity of information calculation

(2) Malicious code classification module: in order to fully consider the key features of malicious code images and further improve the classification accuracy, a malicious code classification method combining a hybrid attention mechanism and a deep residual network is proposed. This method uses ResNet50 as the backbone network since the residual network can increase the accuracy by increasing the considerable network depth. The internal residual module uses shortcut connection to alleviate the problem of gradient disappearance caused by increasing depth of the network. Then, the channel attention module and the spatial attention module constitute a hybrid attention module, which is added to the residual unit of each convolution part of ResNet50 to improve the representation ability of the convolutional network. Combine the two to build a classification model, train the malicious code image dataset, and finally, realize the effective classification of malicious code

3.2. Malicious Code Visualization Module

3.2.1. Visual Problem Analysis. In an image, as the carrier of the malicious code file, each pixel contains a lot of code file information, and different malicious code images have different malicious code data structures and information amounts. For example, the images of malicious code of the Kelihos_ver1 family, the Vundo family, and the C2LOP.-gen!g family are shown in Figure 2. It can be seen that there are visual similarities between the malicious code images corresponding to the malicious code variants of the same family, while there are obvious visual differences between the malicious code images corresponding to different family variants. This difference in visual features shows that malicious code classification based on image similarity is feasible and effective. Thus, the generation method of malicious code images is very important for malicious code classification.

Generally, bytecode is a complied intermediate binary code that is independent of specific machine code and implementation platform. The assembly code is a low-level hardware-related assembly instruction compiled from the source code, which has poor cross-platform performance but relatively high execution performance. Bytecode and assembly code reflect different information about the code, but there is a close correlation between them. As a low-level language, assembly code has high scalability and lengthy code, so the assembly file of the same malicious code is longer and more informative than the bytecode file. Therefore, in order to comprehensively use image information to support the effective and accurate classification, in the malicious code visualization

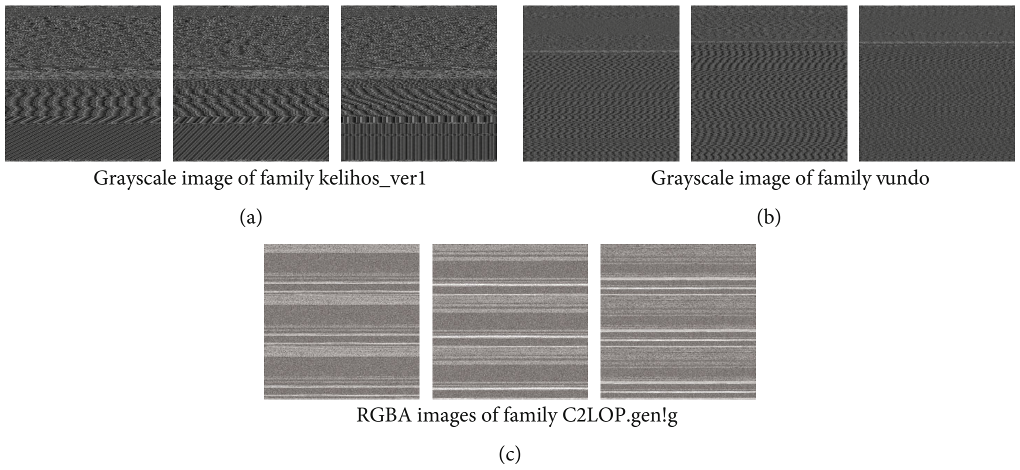

Grayscale image of family kelihos_ver1
(a)
Grayscale image of family vundo
(b)
RGBA images of family C2LOP.gen!g
(c)

Figure 2: Images of different families of malicious code.

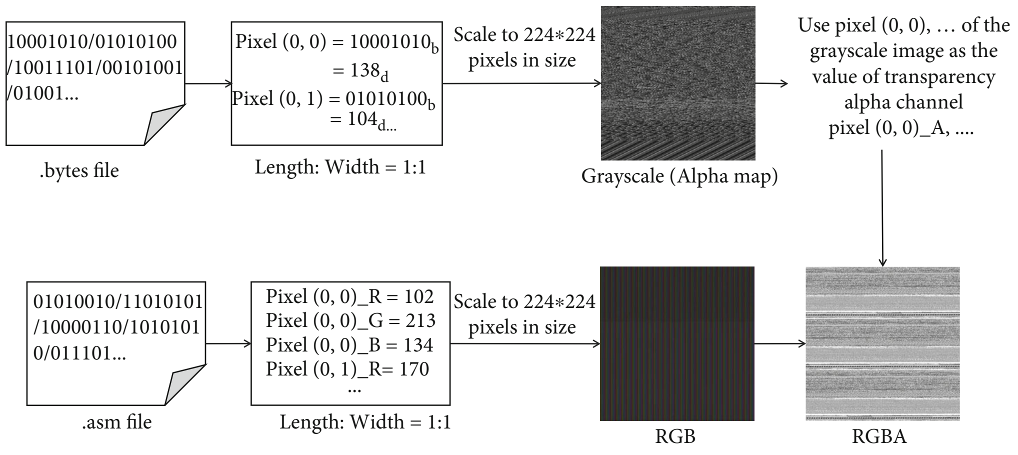


Figure 3: RGBA image generation flowchart.

module, we innovatively choose to use both the bytecode file and assembly file information to convert the malicious code into an RGBA image for subsequent classification. Compared with the grayscale image with only one channel, an RGBA image is an image with four channels by adding a transparency channel to an RGB image with three channels. Consequently, the effective information amount of RGBA images is 4 times that of grayscale images, which can provide more comprehensive and accurate image features for subsequent detection. Furthermore, RGBA images can not only carry more channel features but also can effectively fuse bytecode and assembly files.

3.2.2. RGBA Image Generation. Considering the differences in the amount of information between the assembly file and the bytecode file and the composition of the RGBA image, firstly, the assembly file and the bytecode file are converted to the same size ($224 * 224$ pixels) RGB image and grayscale image, respectively. Then, the grayscale image value is used as the transparency channel and merged with the RGB image generated by the assembly file to form an RGBA image. The RGBA image contains 4 channels, which are red, green, and blue color channels and transparency channel. Each channel has 8 bits and a total of 256 color levels. The malicious code file is read according to the binary data stream, and each 8-bit length ranges from 0 to 256, which exactly matches the length of each channel. In this case, the bytecode and assembly files are not added or deleted, and the bytecode and assembly features are similar in each local detail of the RGBA image after they are converted to images and fused. Thus, RGBA images can not only carry more channel features but also can effectively fuse bytecode and assembly files.

Based on the above analysis, suppose that the malware's .byte file is .byte = (..., 01101110, 10011100, 11010011, ...) = (..., 110, 156, 211, ...), and the .asm file is asm = (..., 01101100, 10011101, 11010010 ...) = (..., R:108, G:157, B:210, ...). The RGBA image generation flowchart and algorithm are shown in Figure 3 and Algorithm 1, and the specific steps are as follows:

(1) Read the malware's .bytes file by reading a binary data stream, and every 8 bits is converted to an unsigned integer vector. The value range of 8-bit unsigned integer is 0~255, which exactly corresponds to the pixel gray value 0~255. According to

length : width equal to 1 : 1, to generate a grayscale image

$$m_{\text{gray}} \in R^{1\times n\times n} = \begin{bmatrix} \cdots & 110 & 156 & 211 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \quad (1)$$

$n = \text{sqrt}(\text{file.length})$, and then, scale the original grayscale image to gray image $m_{\text{gray}} \in R^{1\times224\times224} = \text{Image.resize}((224, 224), \text{Image.ANTIALIAS})$

(2) Read the malware's .asm file by reading the binary data stream as well, each 8 bits corresponds to the R, G, and B values of a pixel ($R_k = \sum_{i=0}^{7} b_{i+16} \times 2^i$; $G_k = \sum_{i=0}^{7} b_{i+8} \times 2^i$; $B_k = \sum_{i=0}^{7} b_i \times 2^i$), according to length : width equal to 1 : 1 to generate RGB image

$$m_{rgb} \in R^{3\times n\times n} = \left\{ \begin{bmatrix} \cdots & R:108 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} \cdots & G:157 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} \cdots & B:210 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \right\}, \quad (2)$$

$n = \text{sqrt}(\text{file.length}/3)$ and then scale the original RGB image to RGB image $m_{rgb} \in R^{3\times224\times224} = \text{Image.resize}((224, 224), \text{Image.ANTIALIAS})$

(3) The gray value of the gray image is used as the transparency channel of the RGBA image, and it is merged with the RGB image generated by the .asm file to form an RGBA image $m_{rgba} \in R^{4\times224\times224}$

When the malicious code file is converted into an image, the zero-padded operation is performed instead of intercepting part of the file content, which ensures the source integrity of the malicious code image information to a certain extent. The parameter of the resize() function is set to Image.ANTIALIAS, which will perform high-quality compression on the image to ensure that the image quality will not be reduced when the image size changes. In this way, the RGBA image contains 4 channels, and the effective information contained is 4 times that of the grayscale image, which can provide more potential malicious code features and effectively support the subsequent malicious code classification.

3.3. Malicious Code Classification Module. After obtaining the malicious code image dataset through the above malicious code visualization module, the next step is to build an accurate malicious code classification model. In the malicious code classification module, we propose an innovative malicious code classification model based on the deep residual neural network - ResNet50 [42] and the attention module structure of Woo et al. [38], which combines a hybrid attention mechanism with the deep residual network to further improve classification accuracy. On the one hand, the deep neural network is used to improve classification accuracy by increasing the structural depth. Meanwhile, the residual structure can effectively avoid the problem of gradient disappearance through the shortcut connection. On the other hand, a hybrid attention mechanism is applied and injected into the residual network to effectively capture the key features of malicious code images and assign different learning weights, so that the model can learn the image features that need to be focused to further improve the classification accuracy. Moreover, the application of the attention mechanism adds less parameters and calculation amount, which can ensure the classification effect of the model without affecting the classification efficiency.

The overall network architecture of the classification model Mcs - ResNet is shown in Figure 4, containing 5 convolution parts (conv1~conv5). Among them, conv2_x, conv3_x, conv4_x, and conv5_x are formed by adding a hybrid attention module to the residual unit of the convolution part, to ensure the full integration of the hybrid attention module and the deep residual network to further enhance the mining of deep features. Moreover, the detailed parameter information of the model is shown in Table 1. Here, a 50-layer ResNet model with 3 layers of bottleneck blocks is chosen as the base network for malicious code classification. Therefore, the model complexity is about 3.8 billion FLOPs (floating-point operations) and so is the parameter size.

The convolutional layer implements the feature extraction and feature mapping, weight sharing, and local connection of the input image through the convolution filter in CNN. Generally, in the convolution process, the convolution filter often has multiple channels, and the filters of multiple channels usually perform feature extraction at the same time. For example, when the input image is $m_{i,j,k} (0 \le i \le W, 0 \le j \le H, 0 \le k \le K)$, that is, the image size is $W \times H$ and the channel is K, the convolution processing is shown in

$$m'_{i,j,k} = \sum_{l=0}^{K-1} \sum_{p=0}^{M-1} \sum_{q=0}^{M-1} m_{i+p,j+q,k+l} w_{p,q,l} + b_{i,j,k}, \quad (3)$$

where $b_{i,j,k}$ represents the bias of the neural network and $w_{p,q,l}$ represents the weight of K convolution filters with a size of $M \times M$.

```
Input: The bytecode file $file_{bytes}$ and assembly file $file_{asm}$ of the malicious code;
Output: The final training dataset RGBA images $m_{rgba} \in R^{4\times224\times224}$.
For each sample $file_{bytes}$:
        Calculate the width of the image $width_{bytes} = sqrt(file_{bytes}.length)$;
        Calculate the gray value corresponding to each pixel, $gray_n = \sum_{i=0}^{7} b \times 2^i$, form a grayscale image;
        Scale the original grayscale image to 224 * 224 pixel size gray image $m_{gray} \in R^{1\times224\times224}$ = Image.resize((224, 224), Image
.ANTIALIAS)..
End
For each sample $file_{asm}$:
        Calculate the width of the image $width_{asm} = sqrt(file_{asm}.length/3)$;
        Calculate the R, G, B value of each pixel, $R_k = \sum_{i=0}^{7} b_{i+16} \times 2^i$; $G_k = \sum_{i=0}^{7} b_{i+8} \times 2^i$; $B_k = \sum_{i=0}^{7} b_i \times 2^i$, form an RGB image;
        Scale the original RGB image to 224 * 224 pixel size RGB image $m_{rgb} \in R^{3\times224\times224}$ = Image.resize((224, 224), Image.
ANTIALIAS).
End
For each image $m_{gray}$ and $m_{rgb} \in R^{3\times224\times224}$:
        The gray value of $m_{gray}$ is used as the $A$ value of the RGBA image;
        Merged with $m_{rgb}$ to form an RGBA image $m_{rgba} \in R^{4\times224\times224}$.
End
```

Algorithm 1:RGBA image generation algorithm.

As shown in the lower part of Figure 4, the hybrid attention module is composed of a channel attention module and a spatial attention module to simultaneously obtain the channel feature weights and spatial feature weights of the malicious code image, thereby enhancing the obtained important features. After that, the enhanced features and the original input image features are connected through the shortcut connection structure to obtain the final output features. The channel attention module and the spatial attention module emphasize the special regions of the malicious code image to enhance the accuracy of malicious code image classification. The following describes the residual module and hybrid attention module and their combination in detail.

3.3.1. Residual Module. The deep learning model is usually composed of multiple layers, and its deep structure has powerful learning capabilities and efficient feature expression capabilities to automatically learn features from a large amount of data. It is widely used in image recognition, speech recognition, and other fields, and has become an important part of computer vision technology. The network depth of a deep learning model determines whether it can extract deeper features, but as the network depth continues to deepen, it will cause network degradation and gradient disappearance problems. The residual network proposes a shortcut connection technique to solve the above problems. The input is transferred across layers and added to the result of the convolution, and the identity mapping is added, as shown in Figure 5(a). When the network input is x, the learned feature is $F(x) + x$, that is, the unit input and output are directly added, and then activated by the ReLU activation function. This network structure does not add additional parameters, which facilitates the subsequent network optimization and greatly improves the training efficiency. Based on these characteristics of the residual network, the attention module is injected into the residual network to construct a residual attention network to simultaneously utilize the advantages of both, as shown in Figure 5(b).

Here, the proposed malicious code classification model Mcs-ResNet uses the ResNet50 residual network as the backbone network. ResNet50 is a deep residual network formed by adding a shortcut connection mechanism on the basis of the VGG19 network. The network structure of the traditional CNN model is directly stacked, which is equivalent to multiplication calculation. In this ResNet model, the network structure is connected through a shortcut connection, and the calculation is changed from multiplication to addition. The feature calculation under this structure will be more stable, so the original feature information in the malicious code image and the key feature information processed by the attention module will flow to the next layer more stably, and the malicious code image classification will be more efficient.

Based on the above analysis, the expression of the RGBA image $m \in R^{4\times224\times224}$ processed by the residual module is as follows:

$$m' = m + f(F_S(F_C(m))), \tag{4}$$

where f represents operations such as feature mapping, activation, and attention weighting; F_S is the spatial attention weight; and F_C is the channel attention weight. The specific calculation of F_S and F_C will be described in the next section. At this time, the features of the RGBA image are not compressed, so that the channel and spatial features can be learned more fully after adding the attention module. This ensures that more critical deep features in the two dimensions flow more stably to the next layer.

3.3.2. Hybrid Attention Module. As mentioned earlier, the use of attention mechanism in CNNs can focus on key

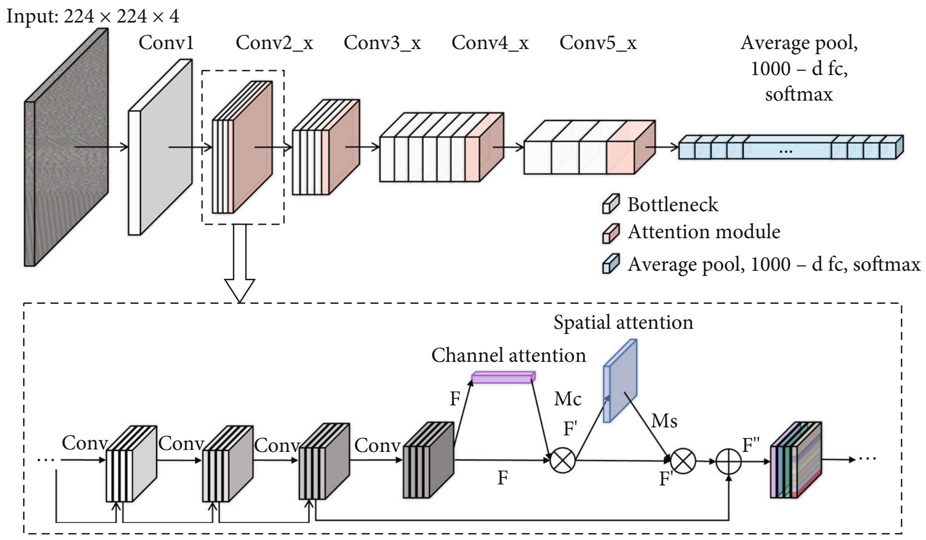


Figure 4: The network architecture of Mcs-ResNet.

Table 1: Network architecture parameter information.

Layer name	Output size	Model
conv1	112×112	7×7, 64, stride 2
		3×3 max pool, stride 2
conv2_x	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3 \oplus$ **Attention**
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4 \oplus$ **Attention**
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6 \oplus$ **Attention**
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 \oplus$ **Attention**
	1×1	Average pool, 1000-d fc, softmax
FLOPs		3.8×10^9

information and improve the convolutional representation ability. Therefore, an attention module is added after the residual module to focus on key information and weaken the useless information. The one-dimensional channel attention feature matrix and the two-dimensional spatial attention feature matrix are derived in turn, and then, the generated attention feature matrix is multiplied with the original input feature matrix to form the output feature matrix, which enables the classification model to focus on key areas with higher correlation with malicious behaviors for more accurate classification.

(1) Channel Attention Module. Compared with grayscale images or RGB images, RGBA images contain richer information and more channels. Using the CNN with channel attention for classification can assign different weights to each channel, thereby effectively improving the classification accuracy of malicious code. In the CNN, the two-dimensional malicious image will generate an image feature matrix (H, W, C) after the convolution kernel operation, where H, W represent the image height and width, and C represents the image feature channel. Introducing the channel attention mechanism into the malicious code classification model can effectively strengthen the model's extraction of global texture features of malicious code images. The channel attention module can pay attention to the importance of different feature channels of the input image. By modeling the importance of each feature channel, assign different weights to the channel features, and strengthen or suppress different channels according to the degree of correlation with malicious behavior.

The operation process of the channel attention module is shown in Figure 6, and the specific steps are as follows: firstly, the output feature matrix of the previous layer of convolution is used as the intermediate input feature. Then, the intermediate feature matrix obtains two-channel descriptions in the form of $1 \times 1 \times C$ through average pooling and maximum pooling based on spatial dimensions to compress the spatial dimensions of the input feature matrix and gather spatial information. The feature information is extracted from different angles, the importance of each feature channel is modeled, and the channel features are assigned weights, thereby effectively utilizing the special interaction relationship between the channels of the intermediate feature matrix obtained after convolution. Afterwards, through the adjustment of the shared network multilayer perceptron, the output vector dimension should match the number of channels of intermediate feature matrix, and the adjusted vector elements are added together and activated by the

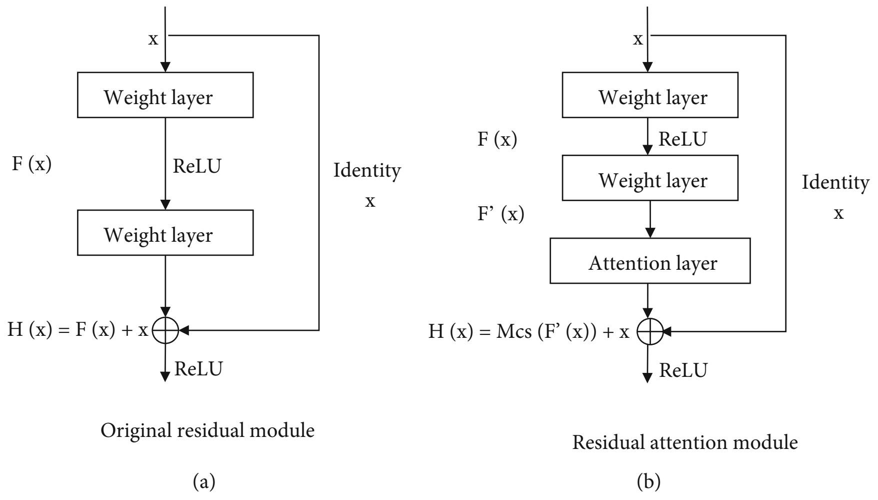


Figure 5: Residual module.

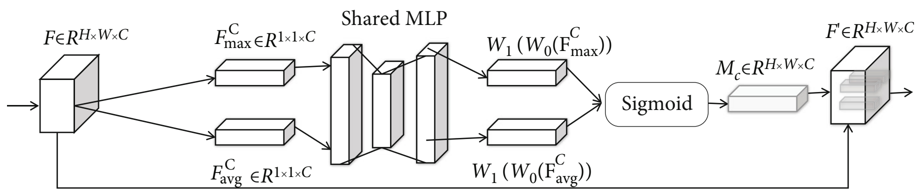

$F \in R^{H\times W\times C}$
$F^C_{max} \in R^{1\times1\times C}$
Shared MLP
$W_1(W_0(F^C_{max}))$
$M_c \in R^{H\times W\times C}$
$F' \in R^{H\times W\times C}$
Sigmoid
$F^C_{avg} \in R^{1\times1\times C}$
$W_1(W_0(F^C_{avg}))$

Figure 6: The operation process of the channel attention module.

Sigmoid function, realizing the enhancement or suppression of different channels as needed. The expression of the channel attention module is shown in formula (5).

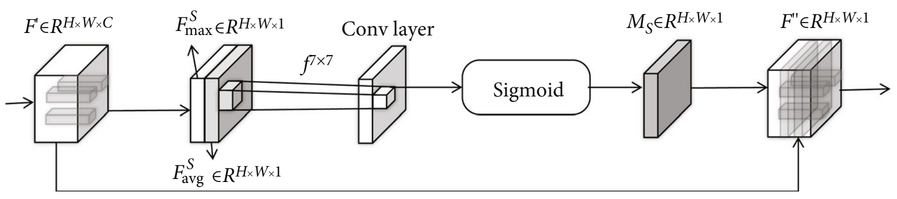
$$M_C(F) = \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F)) + \mathrm{MLP}(\mathrm{MaxPool}(F))), \\ = \sigma\left(W_1\left(W_0\left(F^C_{\mathrm{avg}}\right)\right) + W_1\left(W_0\left(F^C_{\mathrm{max}}\right)\right)\right), \tag{5}$$

where σ denotes the Sigmoid function, $W_0 \in R^{(C/r)\times C}$ and $W_1 \in R^{C\times(C/r)}$, r is the compression ratio. Note that W_0 and W_1 are weights of the multilayer perceptron (MLP), shared by the input features and the ReLU activation function of W_0. F^C_{avg} and F^C_{max} represent the spatial matrix generated by the average pooling and the maximum pooling.

Finally, the channel attention module output matrix and the input intermediate feature matrix are weighted and summed channel by channel to complete the channel attention calculation of the output feature matrix. On the basis of the residual module, combined with the channel attention module, it can retain more global texture information in the input malicious code image, greatly improving the malicious feature representation ability.

(2) Spatial Attention Module. Since most new malicious codes are derived from existing malicious code mutations, their core modules are repeatedly rewritten to generate new malicious code. Hence, the key to malicious code variant detection is how to extract the core module feature information and how to assign different weights to different regions in the image to focus on key feature information to improve the detection and classification accuracy of malicious code variants. The spatial attention module focuses on the importance of different feature spatial locations, generates spatial attention weights for the output feature map, and enhances the spatial location features with higher correlation with malicious behavior according to the feature weights.

The operation process of the spatial attention module is shown in Figure 7, and the specific steps are as follows: firstly, take the feature matrix processed by the channel attention as the intermediate input feature, and perform average pooling and maximum pooling, respectively, based on channel dimensions to obtain two spatial description matrices in the form of $H \times W \times 1$. This will not only consider the contribution of local malicious code image space but also can capture the contribution of global space. Next, the two spatial description matrices are merged into a feature matrix, and a two-dimensional spatial attention map is generated through the convolutional layer to better fit the spatial complexity correlation. Thus, adding a spatial attention module to the classification model can improve the learning ability in key regions with higher correlation with malicious behavior and complements the channel attention, thereby further improving the classification accuracy. Finally, the spatial attention map can be generated after the activation of the Sigmoid function. The spatial attention module is shown in formula (6), where σ represents the



$F \in R^{H\times W\times C}$
$F^S_{max} \in R^{H\times W\times 1}$
Conv layer
$f^{7\times7}$
Sigmoid
$M_S \in R^{H\times W\times 1}$
$F'' \in R^{H\times W\times 1}$
$F^S_{avg} \in R^{H\times W\times 1}$

Figure 7: The operation process of the spatial attention module.

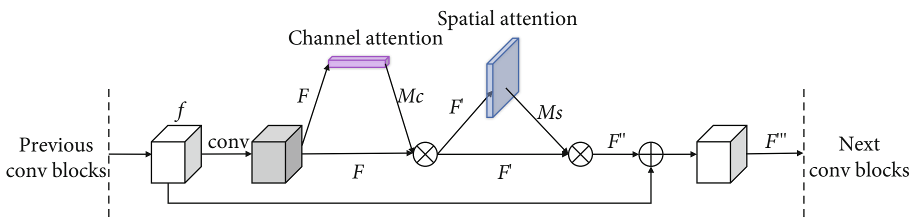


Figure 8: Local structure of the classification model.

sigmoid function, $f^{7\times7}$ represents the convolution operation, and the size of the convolution kernel is 7×7. F^S_{avg} and F^S_{max} also represent the matrices generated by the average pooling and maximum pooling.

$$M_S(F) = \sigma\left(f^{7\times7}(\mathrm{AvgPool}(F)\,;\mathrm{MaxPool}(F))\right), \\ = \sigma\left(f^{7\times7}\left(F^S_{avg}\,;F^S_{max}\right)\right). \tag{6}$$

(3) Hybrid Attention Mechanism. The classification model proposed in this study extracts malicious code features by fusing channel attention and spatial attention. The channel attention module focuses on the global feature information between each channel, and the spatial attention module focuses on the local feature information within the channel. The combination of the two forms a hybrid attention mechanism, which supports the learning of key features and further improves classification accuracy. Woo et al. [38] proved that the channel attention module and the spatial attention module can be arranged in parallel or sequentially, but the sequential arrangement has better performance, and the model performance with the channel attention module priority is slightly better than the spatial attention module priority. The reason is that the channel attention focuses on “what” is critical and meaningful in an input image, and the spatial attention focuses on “where” is an informative part, which is complementary to the channel attention. Therefore, the priority order of the channel attention module is used in the proposed classification model.

3.3.3. Classification Model Structure. Based on the above analysis, the classification model structure that combines the residual module and hybrid attention mechanism is shown in Figure 8. Firstly, perform a convolution operation on the features generated in the previous layer to generate the input feature F. F passes through the channel attention module to obtain the importance of each feature channel, so that the model pays more attention to the channel related to malicious behavior with high weight and suppresses the channel with low correlation, so as to obtain the channel attention feature Mc. The corresponding matrix elements are multiplied by F and Mc to extract the features from the spatial dimension, improve the classification model’s ability to extract local texture features, and obtain the new feature F'. Then, the F' is used as the input feature of the spatial attention module to obtain the spatial attention feature Ms. Ms and F' are multiplied by the corresponding matrix elements to obtain the mixed feature F''. Finally, F'' is added to the features generated in the previous layer to generate feature F''' as the input of the next module.

The whole attention calculation process is shown in formulas (7)–(9). This process strengthens the feature information between channels in the global features of the malicious code and the local location information within the channels, thereby improving the classification performance.

$$F' = M_C(F) \otimes F, \tag{7}$$

$$F'' = M_S\left(F'\right) \otimes F', \tag{8}$$

$$F''' = F'' \oplus f. \tag{9}$$

In order to fully learn the image characteristics of malicious code and improve the performance of the attention module, a hybrid attention module is added after each residual unit instead of just adding it once. Therefore, when the next module performs the deep convolution operation, the features learned by the attention module in the previous module will be retained to continue learning. Moreover, although channel attention and spatial attention are arranged sequentially, they are also connected by identity mapping, which can prevent information of different dimensions from interfering with each other.

Table 2: Malware dataset.

Dataset	Family	Number of samples	Family	Number of samples
BIG2015	Gatak	1013	Ramnit	1541
	Kelihos_ver1	398	Simda	42
	Kelihos_ver3	2942	Tracur	751
	Lollipop	2478	Vundo	475
	Obfuscator.ACY	1288		
Malimg	Adialer.C	125	Lolyda.AA2	184
	Agent.FYI	116	Lolyda.AA3	123
	Allaple.A	2949	Lolyda.AT	159
	Allaple.L	1591	Malex.gen!J	136
	Alueron.gen!J	198	Obfuscator.AD	142
	Autorun.K	106	Rbot!gen	158
	C2LOP.gen!g	200	Skintrim.N	80
	C2LOP.P	146	Swizzor.gen!E	128
	Dialplatform.B	177	Swizzor.gen!I	132
	Dontovo.A	162	VB.AT	408
	Fakerean	381	Wintrim.BX	97
	Instantaccess	431	Yuner.A	800
	Lolyda.AA1	213		

Table 3: Experimental equipment environment.

Hardware	Description	Software	Description
GPU	GTX1060 6GB	CUDA	10.0
CPU	Intel i7-7700	cuDNN	7.6.5
RAM	16 GB	Language	Python

4. Experiment and Performance Analysis

4.1. Experimental Preparation

4.1.1. Experimental Dataset. The experimental dataset used in this study are the BIG2015 dataset (https://www.kaggle.com/c/malware-classification/data) and the Malimg dataset (https://www.kaggle.com/keerthicheepurupalli/malimg-dataset9010). The BIG2015 dataset is a 500 G malware file dataset released by Microsoft on Kaggle during its malware classification challenge in 2015, which includes assembly files and bytecode files of more than 20,000 malware samples. In addition to providing services in Kaggle competitions, the BIG2015 dataset has become a standard benchmark for studying malware behavior modeling. So far, it has been cited by more than 50 research papers. Therefore, this dataset is used here to verify the performance of the proposed malicious code classification model. In order to facilitate the performance verification, the labeled training dataset which consists of 10868 malware samples from 9 families is selected as the experimental dataset, as shown in the upper part of Table 2, and divided into a training dataset and test dataset according to the ratio of 8 : 2.

The Malimg dataset is released by the Advanced Visualization Research Project of the Visual Research Laboratory under the University of California-Santa Barbara. They first proposed a malicious code visualization method for malicious code detection and classification. In 2011, this team constructed the Malimg dataset and published the code visualization method to promote software security research. This dataset contains a total of 9342 samples from 25 family categories, as shown in the lower part of Table 2. Furthermore, the Malimg dataset is composed of grayscale images converted from malware bytecode files.

4.1.2. Experimental Settings. The experimental environment is shown in Table 3.

The stochastic gradient descent (SGD) algorithm with momentum can effectively suppress the oscillation of SGD and accelerate the convergence speed. The data distribution of the model in this paper is relatively uniform and can be well adapted to the SGD algorithm for model optimization. Therefore, the experiment uses the SGD algorithm with momentum optimization to update the model parameters to improve the computational efficiency, and the momentum is set to 0.9. A total of 2000 epochs are trained, and the training batch samples are 16. The dynamic attenuation learning rate is used, and the initial learning rate is set to 0.01, and the classification function is softmax.

Since the classification of malicious code families is a multiclassification problem, in order to facilitate comparison with other models and better measure the classification performance, the arithmetic average of the accuracy of various malicious code families is taken as the standard for performance evaluation. Here, TP is defined as the number of malicious samples classified as malware, TN is defined as the number of benign samples classified as benign, FP is the number of benign samples classified as malware, and FN is the number of malicious samples classified as benign. Thus, accuracy (Acc) is defined as follows:

$$\mathrm{Acc} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{TP}_i + \mathrm{TN}_i}{\mathrm{TP}_i + \mathrm{FN}_i + \mathrm{TN}_i + \mathrm{FP}_i}. \quad (10)$$

4.2. Ablation Experiment and Analysis. Two sets of ablation experiments are conducted to verify the feasibility and effectiveness of the proposed malicious code visualization and classification module, which includes visualization scheme validity verification and hybrid attention module performance analysis. The comparative experiments all use the classification accuracy (Acc) as the evaluation index to facilitate comparison.

4.2.1. Visualization Scheme Validity Verification. The first experiment applied the malicious code images obtained from different visualization schemes to the classic classification models and the proposed classification model - Mcs-ResNet for comparative analysis. The classic classification models include VGG16, VGG19, and ResNet50 pretraining models for feature extraction. The KNN model is used as the classifier, where k is 5. This group of experiments uses the bytecode files and assembly files provided by the

Table 4: Classification effect of different visualization schemes.

No.	Image	VGG16+KNN5	VGG19+KNN5	ResNet50+KNN5	Mcs-ResNet
1	Byte-gray	88.05	87.33	89.17	89.03
2	Byte-RGB	88.59	88.41	89.98	91.83
3	ASM-gray	89.98	90.77	92.07	89.12
4	ASM-RGB	91.53	90.19	94.34	90.04
5	Byte-gray (224 ∗ 224)	93.64	93.84	93.01	96.72
6	Byte-RGB (224 ∗ 224)	92.54	92.52	91.79	92.98
7	ASM-gray (224 ∗ 224)	93.95	93.49	93.86	96.98
8	ASM-RGB (224 ∗ 224)	94.26	94.41	94.70	96.70
9	RGBA (224 ∗ 224)	**94.69**	**95.23**	**95.12**	**97.21**

BIG2015 dataset for verification. In addition to using the proposed visualization scheme to convert it into RGBA images, only bytecode files or assembly files are converted into grayscale and RGB images, and the image size is changed to show the classification effect based on different visualization schemes.

The experimental results are shown in Table 4, wherein the grayscale image and RGB image generated from the bytecode file are marked as Byte-gray and Byte-RGB, respectively, and the grayscale image and RGB image generated from the assembly file are marked as ASM-gray and ASM-RGB, respectively, and the RGBA images are RGB images that contain transparency information, and 224 ∗ 224 represents the image size. And according to the experimental results, the following conclusions can be drawn:

(1) According to the experimental results of nos. 1-4, the classification effect of converting bytecode files into grayscale or RGB images is almost the same, while the classification effect of converting assembly files into RGB images is better than that of grayscale images. For example, in the classification model ResNet50+KNN5, the accuracy of the bytecode file converted into the two types of images is 89.17% and 89.98%, respectively, with a difference of only 0.81%, while the accuracy of RGB image converted from the assembly file is 2.27% higher than that of grayscale image. The reason is that the assembly file size of the same malware code is much larger than that of the bytecode file. The grayscale image can effectively represent the bytecode file but not the assembly file. Therefore, the classification effect of the bytecode file converted into two kinds of images is similar, and the classification effect of the assembly file converted into RGB image is better.

(2) From the comparison of nos. 5-8 and nos. 1-4, it can be seen that the classification effect of prescaling the image to a uniform size (224 ∗ 224 pixels) with high quality is significantly better than that of directly inputting the original pixel size image. In the classification model composed of VGG19 and KNN5, the accuracy on the original images are 87.33%, 88.41%, 90.77%, and 90.19%, while the accuracy on the corresponding 224 ∗ 224 pixel images improved by 6.51%, 4.11%, 2.72%, and 4.22%, respectively. Meanwhile, in the Mcs-ResNet model, it also improves 7.69%, 1.15%, 7.86%, and 6.66%, respectively, all of which are significantly improved. The reason is that when the image is scaled in the preprocessing, the parameter of the resize() function is set to *Image.ANTIALIAS*. This is an operation for high-quality image compression, and the original image features can be preserved to the greatest extent. But the original image is directly compressed to 224 ∗ 224 pixel size when the data is loaded. This is low-quality processing, resulting in the loss of a large amount of effective information in the original image and poor classification effect. In addition, the performance of the Mcs-ResNet model is better than other models, i.e., in the no. 7 and no. 8 experiments; its accuracy is 3.49% and 2.29% higher than that of the VGG19+KNN5 model, respectively.

(3) It can be seen from no.8 and no. 9 that the classification effect based on RGBA images is better than those based on other images. All four models achieved the best classification accuracy on RGBA images. Especially, the classification accuracy of the proposed model is 97.21%, which is 2.95%, 2.8%, and 2.51% higher than the previous three models based on RGB images (no.8). It is also 2.52%, 1.98% and 2.09% higher than the previous three models based RGBA images (no.9). The information source of RGBA image is composed of bytecode files and assembly files. Therefore, the information source is richer, and the amount of information contained is larger than the grayscale image and RGB image, which can better describe the features of malicious code images.

Based on the above analysis, the proposed visualization scheme is feasible and effective and shows good classification performance in different classifiers. Thus, when fusing bytecode files and assembly files, it is a reasonable choice to use bytecode files as the data source of the transparency channel and the assembly files as the R, G, and B channel data sources. This operation can deeply exploit and utilize the

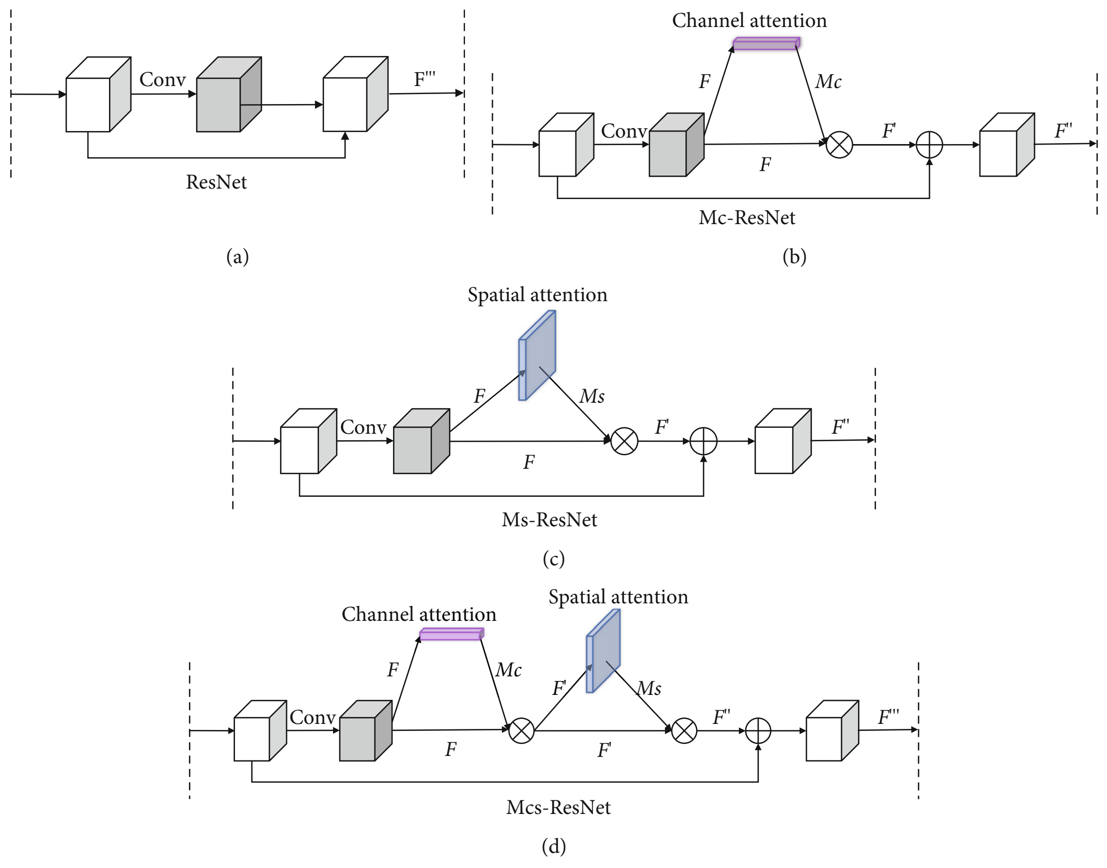


Figure 9: Four models with different attention module structures.

feature information of malicious code images and effectively support the accurate classification of malicious codes.

4.2.2. Hybrid Attention Module Performance Analysis. In order to verify the classification performance of different attention mechanisms on malicious code image classification, four sets of comparative experiments are conducted here, including the ResNet model without an embedded attention module; the ResNet model with only the channel attention module embedded (Mc-ResNet); the ResNet model with only the spatial attention module embedded (Ms-ResNet); and the ResNet model with the hybrid attention module (Mcs-ResNet), as shown in Figure 9.

Except for the embedded attention module, the other model parameters are consistent with those shown in Section 4.1.2. The experimental results are shown in Table 5. It can be seen that:

(1) The Acc value of the ResNet model without the attention module is significantly lower than the other three residual neural network models embedded with the attention module. The ResNet model has the lowest accuracy rate on the Byte-Gray dataset of 78.32%, and the average accuracy rate is 87.39%, which is the lowest among the four models, and 3.7%, 4.87%, and 6.01% lower than the other three models, respectively. It also works best on RGBA images in each dataset, with an accuracy rate of 95.14%, and it is still 0.64%, 1.95%, and 2.07% lower than other models.

(2) The average Acc value of the Mc-ResNet model and the Ms-ResNet model is 3.7% and 4.87% higher than that of the ResNet model. The experiments show

Table 5: Classification results under different attention modules.

No.	Image	ResNet	Mc-ResNet	Ms-ResNet	Mcs-ResNet
1	Byte-gray	78.32	85.96	88.51	**89.03**
2	Byte-RGB	80.44	87.15	88.98	**91.83**
3	ASM-gray	79.38	85.08	87.42	**89.12**
4	ASM-RGB	85.49	88.11	87.97	**90.04**
5	Byte-gray (224 ∗ 224)	94.27	95.78	96.58	**96.72**
6	Byte-RGB (224 ∗ 224)	86.88	89.64	92.21	**92.98**
7	ASM-gray (224∗224)	95.05	96.43	96.38	**96.98**
8	ASM-RGB (224 ∗ 224)	91.54	95.88	95.19	**96.70**
9	RGBA (224 ∗ 224)	95.14	95.78	97.09	**97.21**
Average		87.39	91.09	92.26	93.40

Table 6: Running time (s) under different attention modules.

Dataset	Model	Training time	Prediction time
BIG2015	ResNet	15.1398	10.2272
	Mc-ResNet	19.3584	12.0378
	Ms-ResNet	18.5872	11.3385
	Mcs-ResNet	21.7059	13.5375
Malimg	ResNet	16.4576	3.9875
	Mc-ResNet	20.5867	4.7621
	Ms-ResNet	19.8541	4.5753
	Mcs-ResNet	22.3302	5.7758

that the introduction of the attention mechanism is helpful to the extraction of key image features and can effectively improve classification accuracy. Similarly, the above two models also work best on RGBA images.

(3) The average Acc value of the proposed Mcs-ResNet embedded with the hybrid attention module is 6.01% higher than the ResNet model. Moreover, it is 2.31% and 1.14% higher than the Mc-ResNet model and Ms-ResNet model which are embedded with a single attention module. In general, Mcs-ResNet, which embeds both the channel attention module and spatial attention module, achieves the best classification performance even on different visualization schemes. Especially on RGBA images, its classification accuracy is still the best with an accuracy rate of 97.21%. As shown in the 9th experiment, when using RGBA images, the model after embedding hybrid attention improves the classification accuracy by 2.07% compared to the model with only residual network. Moreover, the classification accuracy is also improved compared with the model embeds channel or spatial attention alone. This further verifies the effectiveness of the proposed malicious code visualization scheme.

The running times of the above four models are shown in Table 6. It can be seen that with the introduction of channel attention and spatial attention mechanisms, the training time and prediction time of the model are longer. But given the improvement in accuracy, the increase in prediction time is small and acceptable within the expected range. In addition, the training time on the two datasets are relatively close, 21.7059 seconds and 22.3302 seconds, while the prediction time are 13.5375 seconds and 5.7758 seconds, respectively, and the residual network model parameter scale is close to 3.8 billion FLOPs. Therefore, the experimental results show that the proposed model can achieve better prediction results within a reasonable running time.

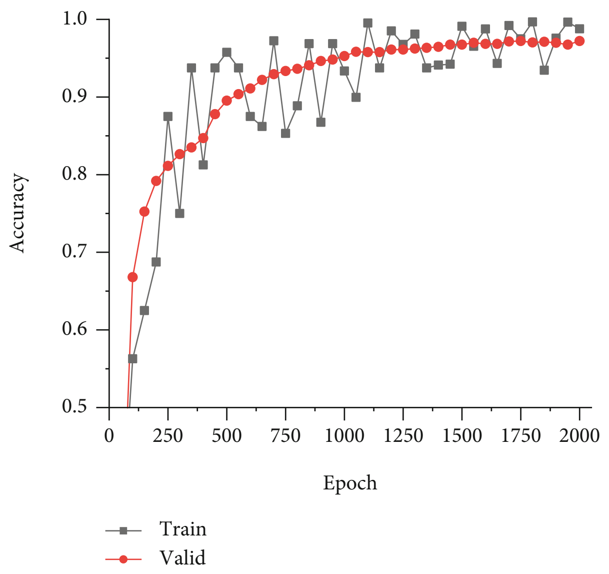
Based on the above analysis, the attention mechanism can improve the classification effect, and different attention mechanisms have different effects on the model. Especially, the introduction of the hybrid attention mechanism in the deep residual network can effectively improve the classification accuracy. The reason is that a single attention mechanism is not enough to fully characterize key features. If channel attention is ignored, the ability to extract global texture features will suffer. If the spatial attention is ignored, it will have an impact on local texture feature learning, thus ignoring the local texture information. The hybrid attention mechanism can learn different weights from the two dimensions of channel and spatial, extract the deep texture feature of malicious code images from the whole and local perspective, and strengthen the model's ability to extract key features. Overall, the fusion of channel and spatial attention mechanism enables the deep features of malicious code images to be fully represented, enabling the classification model to have better classification accuracy for different malicious families.

4.3. Overall Performance Experiment and Analysis. In this section, two groups of experiments are conducted to analyze the overall performance of the model: (1) performance analysis on different datasets: verify the general applicability of the proposed classification model on different experimental datasets; (2) performance comparison analysis: compare the proposed scheme with other malicious code classification schemes to verify the superiority of the proposed scheme.

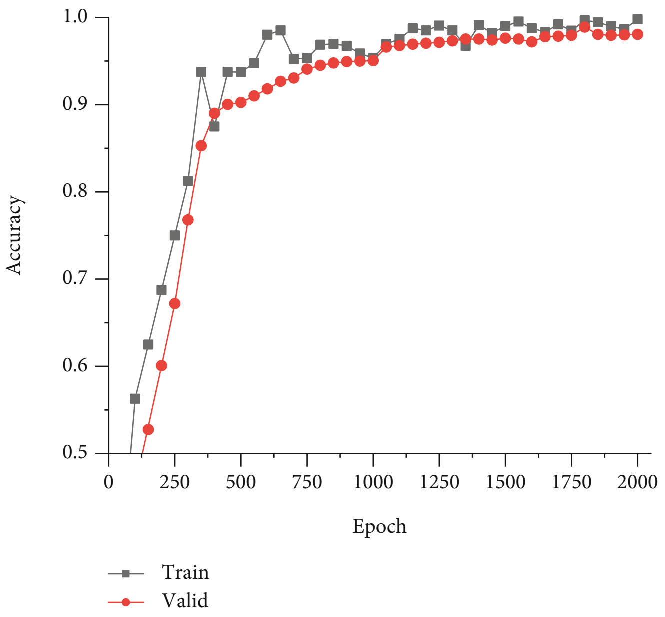
4.3.1. Performance Analysis on Different Datasets. In order to verify the general applicability of the proposed classification model, the BIG2015 dataset and the Malimg dataset are selected for comparative experiments, and the experimental results are shown in Figure 10 and Table 7. The BIG2015 dataset provides both bytecode files and assembly files that can be directly used in the proposed classification model, while the Malimg dataset provides the image format of the malicious code after gray-scale processing. Since the malicious code file is truncated and other operations that cause information loss, reverse processing can be performed to completely restore the image file to a bytecode file. And then use the IDA PRO disassembly tool (https://www.hex-rays.com/products/ida/) to analyze the bytecode files to obtain the corresponding assembly file and finally use the Mcs-ResNet model to classify malicious code.

It can be concluded from the experimental results that when the epoch is 250, the validation accuracy of the BIG2015 dataset is 81.13%, and the validation accuracy of the Malimg dataset is 67.19%, so in the initial stage of training, the optimization of the training effect on the BIG2015 dataset is slightly faster. When the epoch is 500, the validation accuracy of these two datasets reaches 89.53% and 90.25%, respectively; now, the prediction effect on the Malimg dataset is slightly better. And both datasets can reach stability at 1250 epochs, and the accuracy is basically close to the maximum. Finally, it achieved an average classification accuracy of 97.21% on the BIG2015 dataset and 98.06% on the Malimg dataset.

In addition to the accuracy rate, precision, recall, $F1$-score, and confusion matrix are also used to evaluate the



(a) BIG2015 dataset



(b) Malimg dataset

FIGURE 10: The performance of Mcs-ResNet on different datasets.

model performance; the formula and experimental results are shown below (Table 8).

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \tag{11}$$

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}, \tag{12}$$

$$F1\_\text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{13}$$

It can be seen that the proposed model performs well under different evaluation metrics. The number of malicious code families Skintrim.N and Swizzor.gen!E in Malimg dataset is only 80 and 128, and the classification effect on these two families is unstable and is largely affected by the data imbalance. Data imbalance is not the focus of this paper and will be studied in the follow-up work. In summary, the proposed classification model shows good generalization performance and is not limited to a specific dataset, which can achieve better classification results on different datasets while ensuring the classification efficiency (Figure 11).

4.3.2. Performance Comparison of Malicious Code Classification Schemes. The last set of experiments compares the proposed Mcs-ResNet model with several models that currently perform well in malicious code classification to verify its classification performance. The experimental results on two datasets are shown in Table 9. Among them, Nataraj et al. [13] convert the bytecode file into a grayscale image, extract the GIST features, and use the KNN algorithm for classification; Wang et al. [16] convert the bytecode file into an RGB image and use VGGNet model for classification; Cui et al. [43] convert the malicious code into grayscale images and use CNN for classification; Cakir and Dogdu [25] use the assembly file of malicious code, extract features based on Word2Vec, and then, use Gradient Boosting Machine (GBM) for classification; Ma et al. [39] use both bytecode files and assembly files of malicious codes and classify them based on SVM.

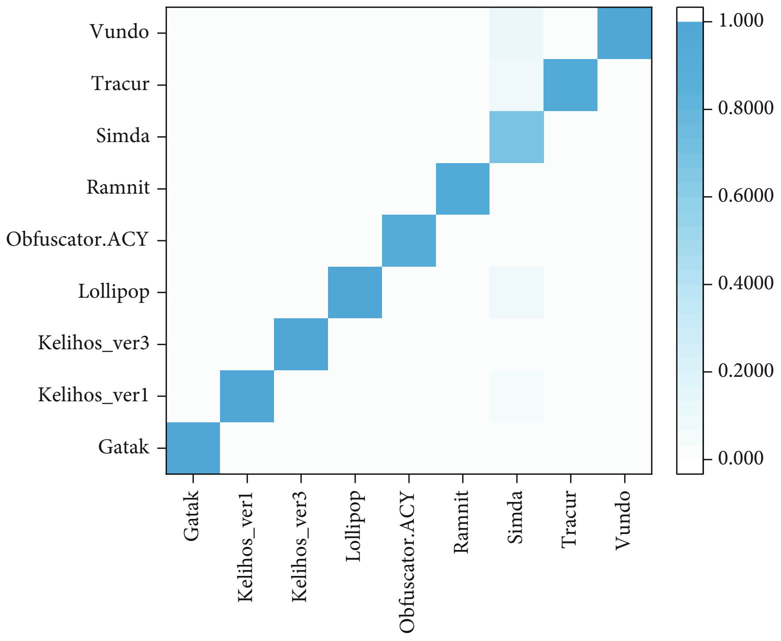
It can be seen from Table 9 that the proposed Mcs-ResNet model reaches 97.21% and 98.06% classification accuracy on the two datasets, respectively. Compared with other methods that only use .bytes files, such as Nataraj et al. [13] and Cui et al. [43], the classification accuracy rate

TABLE 7: Accuracy in different training epochs.

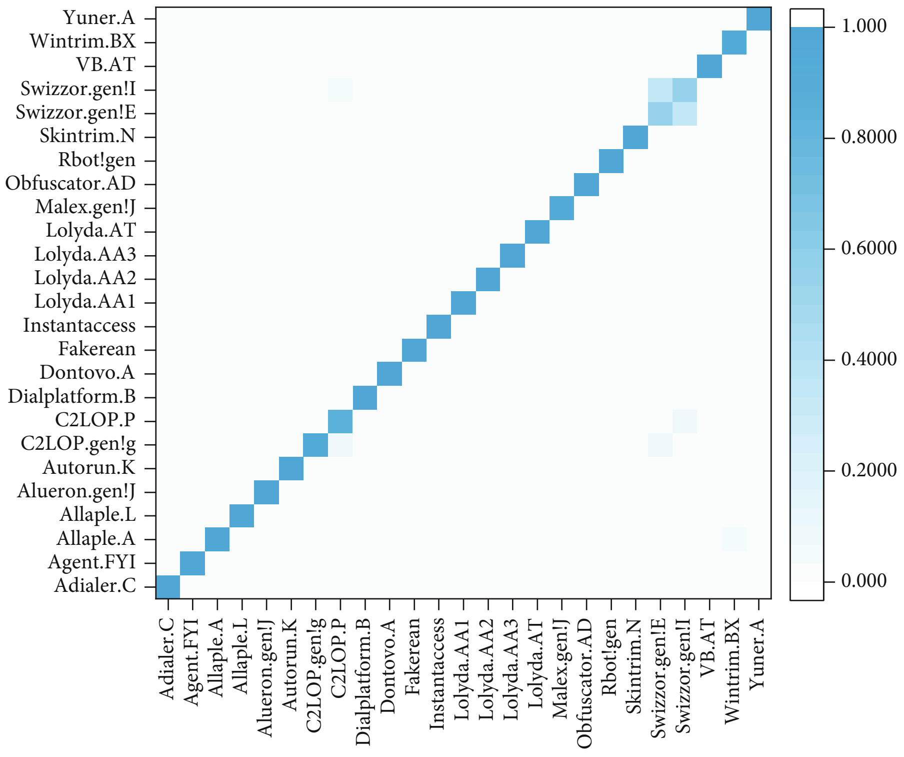
Epoch	BIG2015		Malimg	
	Train	Valid	Train	Valid
250	87.5	81.13	75	67.19
500	95.75	89.53	93.75	90.25
750	85.32	93.35	95.32	94.07
1000	93.35	95.28	95.35	95.04
1250	96.77	96.11	99.07	97.14
1500	99.11	96.75	99.01	97.63
1750	97.49	97.21	98.49	97.95
2000	98.79	97.21	99.79	98.06

TABLE 8: The result of precision, recall, and $F1$-score.

	Precision	Recall	$F1$-score
BIG2015	96.55	96.24	96.39
Malimg	97.11	96.93	97.02



(a) BIG2015



(b) Malimg

Figure 11: Confusion matrix of different datasets.

is increased by 1~3%. Compared with methods that only use .asm files, such as Cakir nad Dogdu [25], the classification accuracy rate is improved by 1.07%. Therefore, the experimental results of using both .byte files and .asm files are better than using only one of them, indicating that more file types can provide more information and further improve the subsequent classification accuracy. And compared with the Ma et al. [39] method that uses both .bytes files and .asm files, the classification accuracy rate is also improved by 1.12%. Ma et al. only use the global attention mechanism to extract weights of each assembly statement, without considering the key channels and regions of intermediate feature maps of the classification model. Therefore, the hybrid attention module composed of channel attention and spatial attention outperforms the global attention mechanism used by Ma et al. Overall, the proposed method has

Table 9: Comparative experimental results.

Method	Files used	Dataset	Accuracy
Nataraj et al. [13]	.bytes	Malimg	97.18
Wang et al. [16]	.bytes	Malimg	96.16
Cui et al. [43]	.bytes	Malimg	94.50
Cakir and Dogdu [25]	.asm	BIG2015	96.14
Ma et al. [39]	.bytes+.asm	BIG2015	96.09
Mcs-ResNet (ours)	.bytes+.asm	BIG2015	**97.21**
Mcs-ResNet (ours)	.bytes+.asm	Malimg	**98.06**

certain advantages in classification accuracy on different datasets over other malicious code classification methods. The reason is that this method uses both bytecode files and assembly files to form RGBA images to obtain more malicious code information, and the proposed hybrid attention mechanism pays more attention to the extraction of key regions and local features, which further improves the classification accuracy.

5. Conclusion and Future Work

Edge security is an important guarantee for edge computing, and it is of great significance to classify malicious code quickly and accurately in the entire life cycle of edge computing. Therefore, a malicious code visualization classification method based on a deep residual network and hybrid attention mechanism for edge security is proposed to effectively support the detection and accurate classification of malicious code. The main contributions are as follows:

(1) A visualization scheme that converts malicious code into RGBA images is proposed to improve the deep feature representation ability of malicious code images. This scheme effectively integrates the bytecode file and assembly file of the malware, deeply exploits and utilizes the image feature information, and solves the problem of a single source of code images in other visualization solutions without adding additional computational complexity

(2) A classification model - Mcs-ResNet that combines a hybrid attention mechanism and deep residual network is proposed to accurately classify malicious code. Due to its powerful feature extraction capability and shortcut connection architecture, the deep residual network improves classification accuracy while alleviating the problem of model degradation and gradient disappearance. The hybrid attention mechanism including channel and spatial attention can effectively extract the key feature information of malicious code images. The combination of the two can further improve the classification accuracy and effectiveness

The experimental results on the BIG2015 and Malimg datasets demonstrate the feasibility and effectiveness of the proposed visualization scheme and classification model. Compared with the existing malicious code classification methods, the proposed model performs better in classification accuracy and generalization performance. Future work will start with the serialization of malicious code. Consider associating the bytecode file of the malware with the assembly file and extracting the features of the sequence information in the vertical direction and the associated information in the horizontal direction to fully utilize the malicious code information. How to better combine the attention mechanism with malicious code classification is also the focus of the future work.

Data Availability

The datasets used in the experimental part include the BIG2015 dataset and the Malimg dataset, from the following websites: https://www.kaggle.com/c/malware-classification/data and https://www.kaggle.com/keerthicheepurupalli/malimg-dataset9010.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors appreciate the support from the Zhejiang Provincial Natural Science Foundation of China (LY20F020015 and LY21F020015), Zhejiang Province Key R&D Project (2021C02012), the National Science Foundation of China (61902345, 61972121, 61902099, 61702517, and 61802101), the Defense Industrial Technology Development Program (no. JCKY2019415C001), and the Open Project Program of the State Key Lab of CAD&CG (grant no. 2109), Zhejiang University.

References

[1] X. Luo, Q. Qin, X. Gong, and M. Xue, "A survey of adversarial attacks on wireless communications," *Champions*, vol. 437, pp. 83–91, 2022.

[2] Z. Guo, Y. Lu, H. Tian, J. Zuo, and H. Lu, "A security evaluation model for multi-source heterogeneous systems based on IOT and edge computing," *Cluster Computing*, vol. 24, 2021.

[3] Y. Wang, G. Yang, T. Li, L. Zhang, and X. Yu, "Optimal mixed block withholding attacks based on reinforcement learning," *International Journal of Intelligent Systems*, vol. 35, no. 12, pp. 2032–2048, 2020.

[4] Wikipedia org, "Wikipedia's official website," 2022, https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

[5] S. Shen, K. Zhang, Y. Zhou, and S. Ci, "Security in edge-assisted Internet of Things: challenges and solutions," *Science China Information Sciences*, vol. 63, no. 12, article 220302, 2020.

[6] S. Greengard, "Cybersecurity gets smart," *Communications of the ACM*, vol. 59, no. 5, pp. 29–31, 2016.

[7] P. Seshagiri, A. Vazhayil, and P. Sriram, "AMA: static code analysis of web page for the detection of malicious scripts," *Procedia Computer Science*, vol. 93, pp. 768–773, 2016.

[8] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Washington, DC, USA, 2016.

[9] J. Cheng, J. Zheng, and X. Yu, "An ensemble framework for interpretable malicious code detection," *International Journal of Intelligent Systems*, vol. 36, 2020.

[10] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of cnn architectures (imcec)," *Computers & Security*, vol. 92, article 101748, 2020.

[11] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.

[12] G. J. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," in *International Workshop on Visualization for Computer Security*, J. R. Goodall, G. Conti, and K. L. Ma, Eds., vol. 5210 of Lecture Notes in Computer Science, pp. 1–17, Springer, Berlin, Heidelberg, 2008.

[13] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, pp. 1–7, Pittsburgh, Pennsylvania, USA, 2011.

[14] L. Nataraj and B. S. Manjunath, "SPAM: signal processing to analyze malware [applications corner]," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 105–117, 2016.

[15] D. Kornish, J. Geary, V. Sansing, S. Ezekiel, L. Pearlstein, and L. Njilla, "Malware classification using deep convolutional neural networks," in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–6, Washington, DC, USA, 2018.

[16] B. Wang, H. H. Cai, and Y. Su, "Classification of malicious code variants based on VGGNet," *Journal of Computer Applications*, vol. 40, no. 1, pp. 162–167, 2020.

[17] B. Sun, P. Zhang, M. Y. Cheng, X. T. Li, and Q. Li, "Malware detection method based on enhanced code images," *Journal of Tsinghua University (Science and Technology)*, vol. 60, no. 5, pp. 386–392, 2020.

[18] T. Chen, B. Xiang, L. V. Mingqi, B. Chen, and X. Jiang, "Android malware detection method based on byte-code image and deep learning," *Telecommunications Science*, vol. 35, pp. 9–17, 2019.

[19] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1923–1938, 2018.

[20] H. Gao, W. Huang, and Y. Duan, "The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, pp. 1–23, 2021.

[21] Y. Yin, Z. Cao, Y. Xu, H. Gao, and Z. Mai, "QoS prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1136–1145, 2020.

[22] F. Qin, N. Gao, Y. Peng, Z. Wu, S. Shen, and A. Grudtsin, "Fine-grained leukocyte classification with deep residual learning for microscopic images," *Computer Methods and Programs in Biomedicine*, vol. 162, pp. 243–252, 2018.

[23] Y. Peng and B. L. Lu, "Discriminative extreme learning machine with supervised sparsity preserving for image classification," *Neurocomputing*, vol. 261, pp. 242–252, 2017.

[24] G. Pitolli, G. Laurenza, L. Aniello, L. Querzoni, and R. Baldoni, "MalFamAware: automatic family identification and malware classification through online clustering," *International Journal of Information Security*, vol. 20, no. 3, pp. 371–386, 2021.

[25] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in *Proceedings of the ACMSE 2018 Conference*, pp. 1–5, Richmond, Kentucky, 2018.

[26] T. N. Turnip, A. Situmorang, A. Lumbantobing, J. Marpaung, and S. I. Situmeang, "Android malware classification based on permission categories using extreme gradient boosting," in *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology*, pp. 190–194, Malang, Indonesia, 2020.

[27] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, "Combining graph neural networks with expert knowledge for smart contract vulnerability detection," in *IEEE Transactions on Knowledge and Data Engineering*, IEEE Xplore, 2021.

[28] S. Choi, "Malicious powershell detection using graph convolution network," *Applied Sciences*, vol. 11, no. 14, p. 6429, 2021.

[29] H. Wu, X. Li, A. T. Liu, Z. Wu, and H. Y. Lee, "Adversarial defense for automatic speaker verification by cascaded self-supervised learning models," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6718–6722, Toronto, ON, Canada, 2021.

[30] D. Hong, L. Gao, J. Yao, N. Yokoya, and B. Zhang, "Endmember-Guided Unmixing Network (EGU-Net): A General Deep Learning Framework for Self-Supervised Hyperspectral Unmixing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 1–14, 2021.

[31] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 146–158, 2021.

[32] H. Gao, X. Qin, R. J. D. Barroso, W. Hussain, Y. Xu, and Y. Yin, "Collaborative learning-based industrial IoT API recommendation for software-defined devices: the implicit knowledge discovery perspective," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 1, pp. 66–76, 2022.

[33] F. T. Jaigirdar, C. Rudolph, and C. Bain, "Prov-IoT: a security-aware IoT provenance model," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1360–1367, Guangzhou, China, 2020.

[34] C. Zhou, Y. Yu, S. Yang, and H. Xu, "Intelligent immunity based security defense system for multi-access edge computing network," *China Communications*, vol. 18, no. 1, pp. 100–107, 2021.

[35] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[36] B. Vasu and N. Pari, "Combining multimodal DNN and SigPid technique for detecting malicious android apps," in *2019 11th International Conference on Advanced Computing (ICoAC)*, pp. 289–294, Chennai, India, 2019.

[37] L. Ghouti and M. Imam, “Malware classification using compact image features and multiclass support vector machines,” *IET Information Security*, vol. 14, no. 4, pp. 419–429, 2020.

[38] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, “Cbam: convolutional block attention module,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, Munich, Germany, 2018.

[39] X. Ma, S. Guo, H. Li et al., “How to make attention mechanisms more practical in malware classification,” *IEEE Access*, vol. 7, pp. 155270–155280, 2019.

[40] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, “Neural malware analysis with attention mechanism,” *Computers & Security*, vol. 87, article 101592, 2019.

[41] C. Wang, Z. Zhao, F. Wang, and Q. Li, “A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet,” *Security and Communication Networks*, vol. 2021, Article ID 6658842, 16 pages, 2021.

[42] 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, 2016.

[43] 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.