Hindawi
Security and Communication Networks
Volume 2022, Article ID 4208066, 13 pages
https://doi.org/10.1155/2022/4208066

Research Article

A New Malware Detection Method Based on VMCADR in Cloud Environments

Luxin Zheng[1] **and Jian Zhang**[1,2,3]

[1]*School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China*
[2]*College of Cyber Science, Nankai University, Tianjin 300350, China*
[3]*Tianjin Key Laboratory of Network and Data Security Technology, Tianjin 300350, China*

Correspondence should be addressed to Jian Zhang; jeffersonzj@qq.com

Received 15 September 2021; Revised 21 December 2021; Accepted 8 March 2022; Published 27 March 2022

Academic Editor: Qi Jiang

With the cloud computing technology developing increasingly, malware and privacy protection have become two major challenges for cloud security. At present, the detection methods based on virtualization technology are mainly in-VM and out-of-VM approaches, both of which have high detection rates. However, a lot of relevant researches at present have focused on the accuracy of malware without considering the privacy protection of cloud tenants sufficiently. In this paper, we propose a new cloud-based malware detection method that can detect malware in cloud service platforms without compromising user privacy. In order to protect the privacy of cloud tenants, this method uses relevant virtualization technologies to obtain memory snapshots of cloud tenants. Because the memory snapshot is very large, and the semantics is of low level, it needs to be processed for feature dimensionality reduction. Therefore, we propose visualized memory change area dimensionality reduction (VMCADR) method. This method directly performs malware detection on binary memory snapshots without accessing user system information and files, thereby protecting user privacy. The following are the main steps of VMCADR method. First, we propose memory difference (MDIFF) algorithm to obtain the Memory Changed Area (MCA), which is changed by the test program. Then, in order to better detect the MCA files, we use visualization technology to process it. Next, we convert these MCA files into grayscale images and RGB images, respectively. And we resize the picture pixels uniformly, so that it can be classified using convolutional neural networks. Finally, we propose a Simplified Neural Network (SNN) to classify these images. After experiments, the RGB-dataset accuracy of malware detection is 99.39%.

1. Introduction

With the cloud computing technology developing increasingly, the malware detection is getting more and more challenging in cloud. In order to avoid detection in cloud, malware programs always change their code while propagating [1, 2].

However, many relevant researches at present have focused on the accuracy of malware detection without sufficiently considering the privacy protection of cloud tenants. Traditional antivirus methods based on signature scanning are challenged by malware polymorphism and antagonism [3]. Types of malware include viruses, worms, Trojan horses, backdoors, spyware, ransomware, bots, and rootkits [4]. In order to allow the malware to perform similar functions, attackers can share most of the code, so that the structure of the malware variants always has small differences [5].

There are some challenges with the existing malware detection methodology. First, many classification methods that extract asm functions by disassembling files have reached high accuracy. These methods usually require deploying professional disassembly software on the tenant's virtual machine, or submitting suspicious files to the server for disassembly. However, due to privacy protection, most cloud tenants may not allow antivirus service providers to access sensitive information in their files, which makes it impossible to extract asm functions. Second, most signatures are based on static analysis of executable file code, but

malware developers often evade static analysis by obfuscation specific modifications to the malware's code. Third, the antivirus software installed on the inspected machine is considered untrustworthy. The inspection mechanism is considered trusted only if the inspected instance does not know the inspection process and cannot interfere with the inspection process. Although some advanced antivirus solutions perform dynamic analysis by simulating the virtual environment and checking the behavior of files, the complex malware can still detect and evade the scanning process.

Researchers rarely share the datasets used in their work, and there is no available malware dataset that can be regarded as a reference dataset [6]. Researchers generally obtain live malware files from AV agents (such as VirusTotal, VirusShare, and VXheaven) to perform malware analysis and detection although there are some existing datasets, such as the big Challenge dataset released by Microsoft in 2015. This dataset is created based on static analysis. And it has the same advantages and limitations as the analysis technology.

In this paper, we propose a credible malware detection method. The advantages of the proposed approach are that we directly perform malware detection on binary memory snapshots without accessing user system information or files. Therefore, it can protect user privacy. The main contributions of this paper are described as follows:

(i) We propose a malware detection method based on memory images to protect user privacy. By using out-of-VM method, the memory is obtained from the VM running test program. Therefore, this method has high reliability. And we do not need to introspect the VM memory to high-level semantics or access user-system information or files. Besides, we directly perform malware detection on binary memory snapshots. Therefore, the method can protect the privacy of cloud tenants.

(ii) We propose an MDIFF algorithm to reduce the size of the feature data. The memory data is very large, but the effective feature data that we need to analyze only occupies a small part of the memory. The effective feature data is the changes to the memory caused by the behaviors of the test program. And we call the data MCA. Therefore, we use the MDIFF to obtain the MCA file.

(iii) We propose an improved neural network model (Simplified Neural Network (SNN)) to perform more accurate and efficient classification. Because the obtained MCA file is a binary file, in order to effectively classify it, we combine the visualization technology to convert it into a grayscale image and an RGB image. In order to improve the detection rate and detection efficiency, we modified the VGG16 model to obtain the optimized SNN model, which has a better detection rate.

The rest of this paper is arranged as follows. Section 2 introduces related work. The third part describes the method structure. Section 4 describes the implementation of the detection method. Section 5 introduces dataset creation and the experiment environment. Section 6 describes the evaluation of the experiment. Section 7 introduces the authors' conclusion.

2. Related Work

In this section, we review the related technologies and methods involved in malware detection in the cloud environment.

2.1. Cloud Computing and Virtualization. Cloud computing has become an indispensable part of IT infrastructure, including both resources and services (private or public) based on cloud computing. With the rapid development of processing and storage technology, computing resources have become cheaper than ever before. With more powerful functions and higher availability, these advancements have opened the door to new "cloud computing" technologies [7]. In cloud computing environment, resources (such as storage, memory, and data) are rented by users/consumers. Virtualization is the core technology within the cloud computing. Virtualization allows multiple operating systems to running together (isolated from each other) on the same physical server. The main software component that enables and monitors virtualization is the host hypervisor. The hypervisor acts as an additional layer between the physical and virtual domains. It manages the hardware resources of the system in order to efficiently allocate them among VMs.

Malware remains one of the main cyber-attack techniques against organizations. This reality has introduced a new business model in cybercrime and ransomware. Therefore, the attackers can encrypt the organization's critical data and files in the Storage Area Network (SAN) of the server and endpoints in this way. The attacker only provides the decryption key after the victim has paid the ransom. Currently, ransomware is a major cyber threat against individuals and organizations (especially against organizational VMs in the computing cloud). Nahmias et al. [7, 8] proposed a method using virtualization at the core of the cloud architecture to generate credible malware signatures based on the presence of malware processes in memory. By querying the hypervisor of the VM, they extracted the volatile memory dump of the VM. When the malware is running on the VM, these dumps (binary file) are obtained dynamically in a trusted way. These dumps include the behaviors of the malware in memory.

2.2. Convolutional Neural Networks. Nowadays, deep neural networks and deep learning have achieved outstanding performance in solving many important problems in computer vision [9], speech recognition, and natural language processing [10]. One of the main reasons why deep learning algorithms are becoming more and more popular is their ability to learn representations [11].

Regarding the use of neural networks for supervisory tasks, every layer, except the last layer, performs a kind of representation learning. The representations learned from

these layers are forwarded to the last layer, which performs linear classification (such as softmax). In principle, after the hidden layer of the network learns to transform the input data into a good representation, the representation can be used as input to any other machine learning model. According to the definition, a good representation can reflect the posterior distribution of the basic explanatory factors of the input data [12].

Convolutional Neural Networks (CNN) [13, 14] rely on the basic knowledge of the topological structure of the input dimensions. And they calculate each low-level feature based on a subset of the given input. In the context of images, each feature is calculated from a fixed-size subimage. This is due to the assumption that the patterns in the input data may appear in different locations. Therefore, a pooling layer is usually used to aggregate all the values calculated from the same feature detector. CNN can be regarded as a feature extractor. It can sequentially scan the topology of the input data and obtain a topological representation. Essentially, the malware features are usually used as input data for antivirus classification programs. Therefore, malware features can be regarded as representations of malware executable files.

2.3. Visual Analysis Method. Visual analysis technology has made a major breakthrough in malware identification. Compared with traditional static analysis methods, it contains fewer features and higher detection accuracy [15]. The method using visualization technology not only inherits the advantages of traditional malware detection technology, but also can extract high-dimensional internal features from data samples. Nataraj et al. [16] first proposed a method to visualize malware as grayscale images. In this method, a given malware binary file is read as a vector of 8-bit unsigned integers and then organized into a 2D array, which can be visualized as a grayscale image in the range [0, 255] (0: black, 255: white). Ni et al. [17] proposed a malware classification method using SimHash and CNN. They converted the disassembled malware code into grayscale images based on SimHash, and they classified the malware family through CNN. Their method successfully utilizes the features provided by the opcode sequence and achieves a very high malware classification accuracy rate. Qiao et al. [15] proposed a unique multichannel visualization method for malware classification based on deep learning. The method enhances the applicability of malware detection. Their method uses both binary byte data and malware disassembly files and combines them effectively. Naeem et al. [18] designed an architecture to detect malware attacks on Industrial Internet of Things (IIoT). In order to analyze the malware in depth, a method based on color image visualization and deep convolutional neural network is proposed. The accuracy of this method on the IIoT dataset reached 98.47% on the Windows dataset.

2.4. Malware Detection in Cloud Environments. Malware detection methods are generally divided into static detection and dynamic detection. The dynamic analysis methods are to identify malware software by analyzing information such as application behavior in the running program. Compared with static analysis, dynamic analysis has higher accuracy, but high time overhead. Dynamic detection technology is currently mainly divided into in-VM and out-of-VM detection.

Willems et al. [19] proposed CWSandbox, a behavior-based dynamic malware analysis tool. It can execute malware samples in a simulated environment to monitor system calls and automatically generate detailed reports. The CWSandbox tool is widely used in dynamic analysis. Mosli et al. [20] used the Cuckoo sandbox to extract features such as registry activities, imported libraries, and API function calls. The linear support vector machine classifier using registered activity features achieves the highest accuracy rate of 96%. However, some malwares can easily detect the security sandbox environment to evade analysis. Sihwail et al. [21] collected datasets from the VirusTotal and Das Malwerk repositories. They used Cuckoo Sandbox and Volatility to generate two reports during each executable period. They extract the unique API call in each report as a feature and convert it into a binary vector. Finally, the result is improved by combining the two reports. The accuracy and FPR of the SVM classifier of this method are 98.5% and 1.7%, respectively.

The experimental environments of the above three methods are all carried out in a sandbox environment, and they all belong to in-VM dynamic detection. The low reliability of data is still the main challenge for in-VM detection, because the acquired data may have been tampered with by malware.

Huang and Stokes [22] proposed a new multitask, deep learning architecture for dynamic malware classification. They used data extracted from the dynamic analysis of malware and normal software to train their model. And they achieved 2.94% error rate of malware family classification. However, because dynamic malware analysis is time-consuming, effectively detecting malware attacks in suspicious files is challenging. Lin et al. [23] proposed a method based on virtual time control mechanics to overcome this challenge. This method uses an improved Xen hypervisor, in which a virtual clock source is generated according to a predefined speed ratio. It speeds up the sandbox system running on the modified hypervisor. The experimental results show that this method speeds up the running speed of the system timer and increases the record size by 41.54%. Nissim et al. [24] obtained memory snapshot of the VM running the malware program through VMM. Then, the method uses WinDbg to extract system calls. Finally, the authors use sequential mining method to analyze system calls. The authors evaluated their methods on ransomware and RATs. Their results show that the proposed method can effectively detect unknown malware. The F-measure value and FPR of the SVM classifier of this method are 100% and 1.4%, respectively.

Wang et al. [25] propose a method to detect kernel rootkits. When the executable program is running on the VM, this method captures a memory snapshot of the VM every 10 minutes, a total of 100 times. Then, they extract several types of data as features in the memory snapshot,

such as threads. The accuracy and FPR of this method for detecting unknown kernel rootkit attacks are 99.8% and 7.6%, respectively. The above four methods are out-of-VM dynamic detection. The reliability of the data they obtain is higher than that of in-VM dynamic detection. However, binary memory snapshots are unreadable data, but what we need is readable high-level semantic data. Therefore, this type of method requires semantic reconstruction of memory snapshots, which will increase time overhead.

In the above methods, most of them need to access the private information of user, such as API calls, system calls, processes, threads, and executable files. This will affect the privacy and security of user. From the perspective of privacy protection, the method we propose should perform malware detection without obtaining information about the user system.

Therefore, this paper only analyzes binary memory snapshots obtained from out-of-VM. In this way, this method can not only ensure the reliability of the acquired data, but also perform malware detection on the premise of protecting user data privacy.

3. Cloud-Based Malware Detection Method

3.1. The Architecture of the Detection Method. We will introduce the architecture of the detection method in this section. In order to eliminate the interference of other system factors on the experiment, we need to create a VM installing pure operating system without other applications as a restoring point. This can ensure that each program is running in the same environment when it is executed. Therefore, we create a pure VM at first. Then, we save the current operating state of the system, so that the virtual machine can be restored to the current system state at any time.

The steps required for one test program of this method are shown in the figure (see Figure 1). In step 1, we obtain the original memory dump file from the VM installing the operating system without other applications. Next, we use the drakvuf [26] tool to make the test program running in the VM automatically in step 2. In the step, we use python script to automatically obtain memory dump file of VM running test program within a certain period of time continuously. In the step of 45, we use the MDIFF algorithm to compare the acquired memory with the original memory dump file to get the MCA file. Immediately after that, we convert all of the MCA files into grayscale and RGB images of the same size in the step of 6. Finally, we propose a neural network model (SNN) to classify these images in the step of 78. The detection result will be compared with the current mainstream model, such as Inception-ResNet-v2 model. This method does not need to perform introspection on the user VM or obtain high-level semantic information of the current state of the user VM system. Therefore, this method protects the user privacy during the entire detection process.

3.2. Motivation Underlying Detection Method. Many studies have focused on the accuracy of malware detection without fully considering the privacy protection of cloud tenants. They can obtain the application process, system calls, and a lot of private information currently used by the tenant through various methods. Their purpose is to obtain the state of the malware program, while it is running in the memory; besides, they can also obtain other sensitive information of the user. However, those methods may reveal user privacy; we need a method that does not affect user privacy to obtain similar features. The state of the malware program in the memory can be seen as its specific modification to the memory. Therefore, on the premise of protecting user privacy, we can use an algorithm to directly obtain the memory changed area caused by test program. We expect that the MDIFF algorithm can find out all the memory changes made by test programs. When malware programs or normal programs run in system memory, their memory allocation patterns might be different. We hope that the differences or changes obtained by MDIFF algorithm can represent these patterns.

4. Methods

4.1. Out-of-VM Memory Snapshot Acquisition. The virtualization platform we use is based on the Xen environment. Xen is an open source virtual machine monitor developed by the University of Cambridge. It has good isolation and can effectively isolate virtual machines. In the Xen environment, there are two main components. One is the virtual machine monitor (VMM) or called hypervisor. The hypervisor layer is between the hardware and the virtual machine. And it is the first layer that must be loaded into the hardware. After the hypervisor is loaded, the virtual machine can be deployed. The virtual machine is called "domain." Among these virtual machines, one of them plays a very important role, that is, domain0, which has high privileges. Domain0 is responsible for some specialized work. Since the hypervisor does not contain any drivers that talk to the hardware, and there is no interface to talk to the administrator, these drivers are provided by domain0. In domain0 virtual machine, administrators can use some Xen tools to create other virtual machines or acquire memory snapshot.

We used shell language to write a script that can automatically complete the entire process of taking a snapshot of the VM running test program. Once the malware sample starts running on the VM, the script will take a snapshot of the VM with a given number of snapshots at a given time interval. Because Xen has good isolation, the virtual machines are transparent to each other. Besides, the malware inside the virtual machine is completely ignorant of the snapshot process; therefore, it cannot interfere with it. However, during the snapshot process, the virtual machine needs to be paused.

4.2. Visualized Memory Change Area Dimensionality Reduction (VMCADR) Methods. In order to effectively detect malware programs, we need to obtain relevant features from memory. However, because the memory snapshot data is

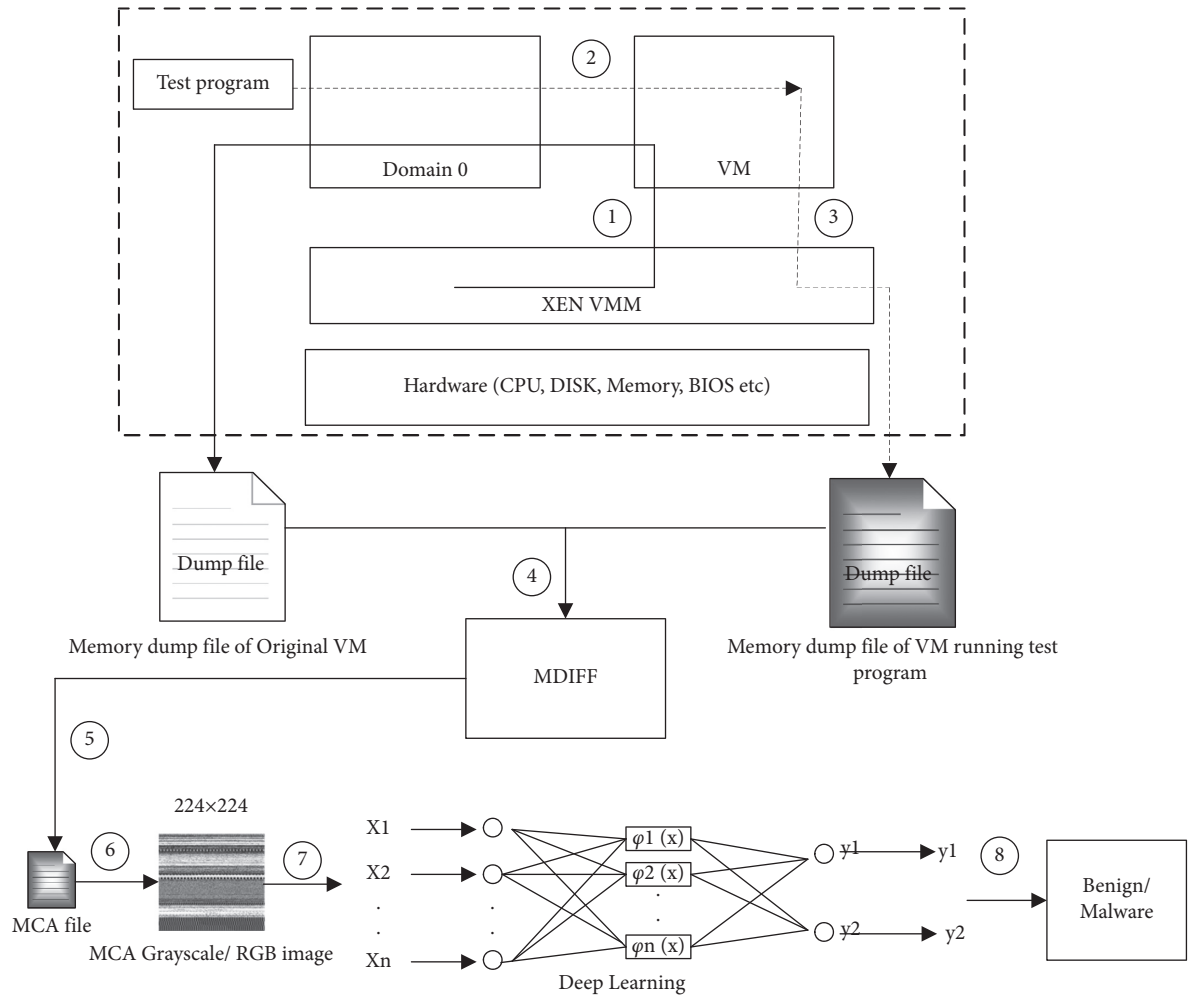

Figure 1: The overview of the detection method architecture.

very large, we need to obtain the effective features of the memory by using a series of techniques.

Jian et al. [27] research verifies the superiority of three-channel RGB images compared to grayscale images in malware detection. It compares the contributions of different channels and shows that data augmentation technology can use visualization technology to make a contribution to malware identification.

Therefore, we propose a visualized memory change area dimensionality reduction (VMCADR) method to achieve memory feature dimensionality reduction and transformation. The main technology of this method is mainly divided into three parts. They are described as follows.

4.2.1. MDIFF Algorithm. When the test program is loaded into the memory, it will make changes to the memory. After we obtain the memory snapshot of the program, the way how to find the memory change area of the program is key to our research. Since the memory snapshot is stored in a binary format, we need to compare the binary files to find out the change area accurately. It is obviously unreliable to update the difference file obtained through a simple binary comparison. Because the impact of the memory stack may not be considered. In order to obtain the memory change area of the program effectively, we propose an algorithm—memory different (MDIFF) algorithm.

MDIFF is an excellent difference algorithm, which draws on a current mainstream update algorithm [28]. And we modify and optimize the algorithm. The basic flow of the MDIFF algorithm is shown in Algorithm 1. Next, we will introduce the main steps of the MDIFF algorithm.

MDIFF can be divided into three parts. First, we sort the contents of the old (original memory snapshot) by sorting technology to form a lexicographic order I. The sorting method we use is faster suffix sorting method with a complexity of nlogn and a space complexity of $O(n)$.

The second step is the core of the algorithm; we will find the longest matching len between old and new (memory snapshot of running test program) through dichotomy. Then, we get the MCA through len. We need to introduce several parameter variables in this step. The scan represents the character to be queried in new, pos represents the matched character in old and the lastscan = scan − lenb, lastpos = pos − lenb, lastoffset = scan − pos. Lastoffset is the offset between new and old. If the area in old can be found in

Input: original memory snapshot, memory snapshot of running test program
Output: MCA file
The old represents the original memory snapshot, the new represents memory snapshot of running test program;
Faster suffix sorting (old)
    Return lexicographic order I;
By using the lexicographic order I, find a position pos in old. The pos maximizes the k of new[scan, scan + k] = old[pos, pos + k]
Return len = k+1, offset = pos-scan.
While (scan < newsize)
  {
    If the length of old[scan, scan + lastoffset] and new[scan, scan + len] does not match with more than 8 bytes then
      Divides the forward-extension (lenf) of the former completely match area and the backward-extension (lenb) of the latter completely match area. The remaining part between the two completely match areas is used as MCA (i) ($i = 1, 2, \ldots, n$)
      Return lenf, lenb, MCA (i).
    else
      continue.
    Integrate all MCA (i) areas into one MCA file. And the MCA file obtained is the final output.
  }

Algorithm 1: The basic flow of the MDIFF algorithm.

new (area + lastoffset = area) in new, it is considered that the area in old and new is completely match area, such as area 2 and area 4 (see Figure 2). The scsc represents the starting position of the comparison between new and old. And the starting position in old is scsc + lastoffset. lenf stands for forward-extension, and lenb stands for backward-extension. By using the lexicographic order I, we find a position pos in old. The pos maximizes the k of $\text{new}[\text{scan}, \text{scan} + k] = \text{old}[\text{pos}, \text{pos} + k]$. And then, the $\text{len} = k + 1$, $\text{offset} = \text{pos} - \text{scan}$. If the length of $\text{old}[\text{scantoscan} + \text{lastoffset}]$ and $\text{new}[\text{scantoscan} + \text{len}]$ does not match with more than 8 bytes, the MDIFF algorithm will divide the forward-extension (lenf) of the former completely match area and the backward-extension (lenb) of the latter completely match area (see Figure 2). The remaining part between the two completely match areas is used as MCA (i). Finally, we obtain the MCA file as the integration of all MCA (i).

4.2.2. Image Conversion. Because the MCA file we obtained is an unreadable binary file, we combine it with visualization technology. In terms of image visualization technology, we used the visualization method of Pinhero [29]. First of all, we read the MCA file byte by byte and map the binary content of each byte to a decimal value in the range of 0 to 255. By doing this, each MCA file has been converted into a 1D vector of decimal numbers.

We need two consecutive decimal values from a 1D array to draw pixels. By using a random number generator [29], a pixel in the grayscale image is plotted through using a 2D color map. We use the decimal values as the row index and column index, respectively. The generated 1D pixel array is reshaped into a 2D matrix and visualized as a grayscale image. The process of converting an MCA file into a grayscale image is shown in the figure (see Figure 3). The pixels in an RGB image are specified by the number of red, green, and blue. In order to obtain the contribution of red, green, and blue in the pixels, three different colormaps of red, green, and blue are used. To draw the pixels in an RGB image, two consecutive values in a 1D vector array are used as row and column indexes [29]. The generated pixel array is reshaped into a 2D matrix and visualized as an RGB image ultimately. The process of converting an MCA file into an RGB image is shown in Figure 4.

The width of the generated image can be set manually. And the length will automatically change according to the size of the MCA file. In our experiment, the relationship between the width of the image and the size of the MCA file is shown in Table 1. The size of the MCA file determines the width of the image it generates. For example, when the size of a MCA file is 1 MB, the width of the image it generates is 1024.

4.2.3. Image Processing. The pixels of the acquired picture are different due to the size of the MCA file. The purpose of memory image processing is to resize the image while preserving the features of the original image as much as possible. In this way, the images can be classified using a convolutional neural network. We used image interpolation to reduce the size of the original image. The principle of the interpolation method is to calculate the value of the target point by using parameters such as the pixel values of several points around the target point. We used interpolation calculation methods to include bilinear interpolation, bicubic interpolation, and Lanczos interpolation [30].

The principle of bilinear interpolation is to use the interpolation algorithm to calculate the value of the target point according to the values of the nearest 4 points around the target point. The image quality obtained by the bilinear interpolation algorithm is the worst, but it has the fastest image processing speed.

The bicubic interpolation algorithm is improved on the basis of the bilinear interpolation algorithm, but the complexity is also increased. The interpolation algorithm not only uses the values of the four directly adjacent pixel points around the target point to participate in the interpolation

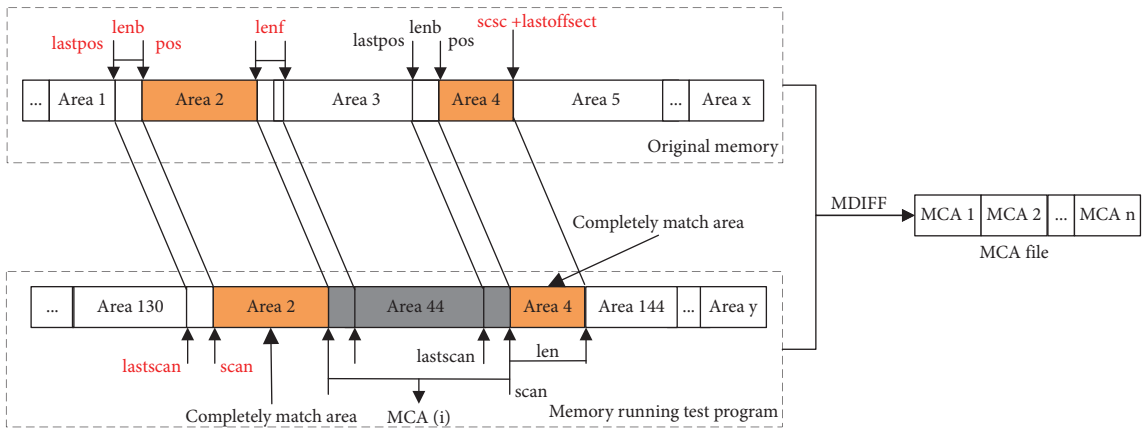

Figure 2: The core principle of MDIFF algorithm.

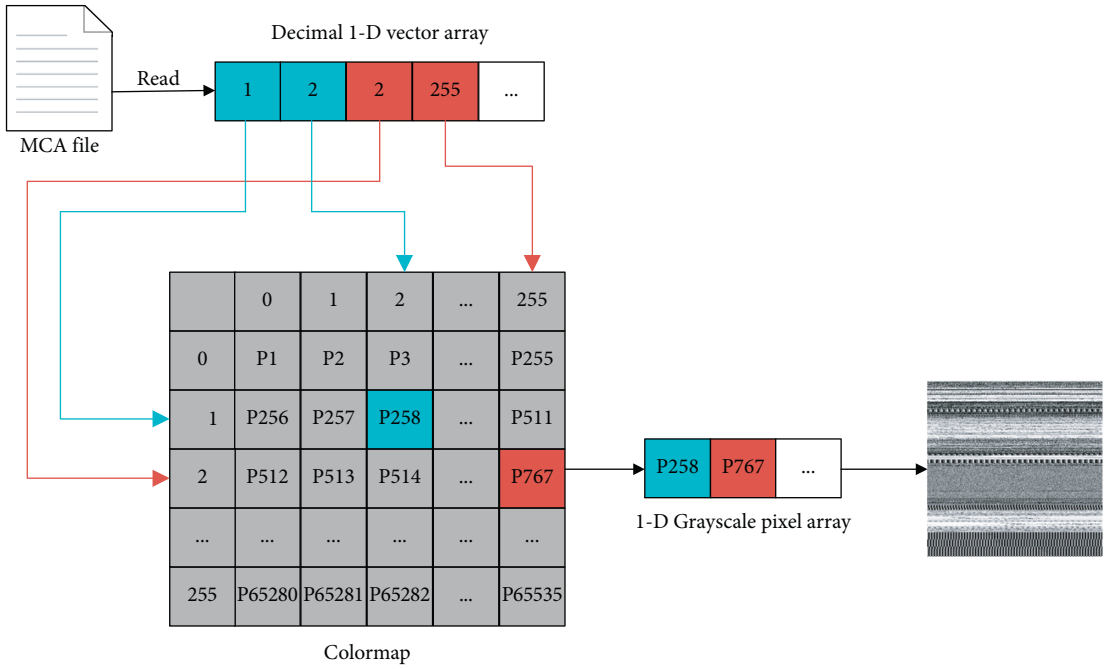

Figure 3: Principles of grayscale image generation.

calculation, but also uses the rate of change of the values of these four points to perform the interpolation calculation. The interpolation algorithm uses the values of 16 pixels near the target point for cubic interpolation calculation, and the cubic polynomial $s(x)$ is also used in the calculation (see (1)). The interpolation algorithm processes images faster than bilinear interpolation. And the image quality obtained is higher than bilinear interpolation, but lower than the Lanczos interpolation algorithm.

The one-dimensional Lanczos interpolation algorithm mainly selects four points from the left and right of the sampling point for interpolation calculation and calculates the weights of these eight points through a high-order interpolation function. The two-dimensional Lanczos interpolation algorithm performs interpolation calculations on the adjacent eight points in the x-axis and y-axis directions, respectively. The principle of Lanczos interpolation algorithm is as follows.

For a one-dimensional vector, the input point set is X, the window size is $2a$, and the weight of each point in the window is calculated. The weight calculation formula is as follows:

$$L(x) = \begin{cases} 1 & \text{if } x = 0 \\ \dfrac{\text{asin}(\pi x)\sin(\pi x/a)}{\pi^2 x^2} & \text{if } 0 < |x| < a \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

After the weight of each point in the window is obtained, the weighted summation of the points in the window s_i can

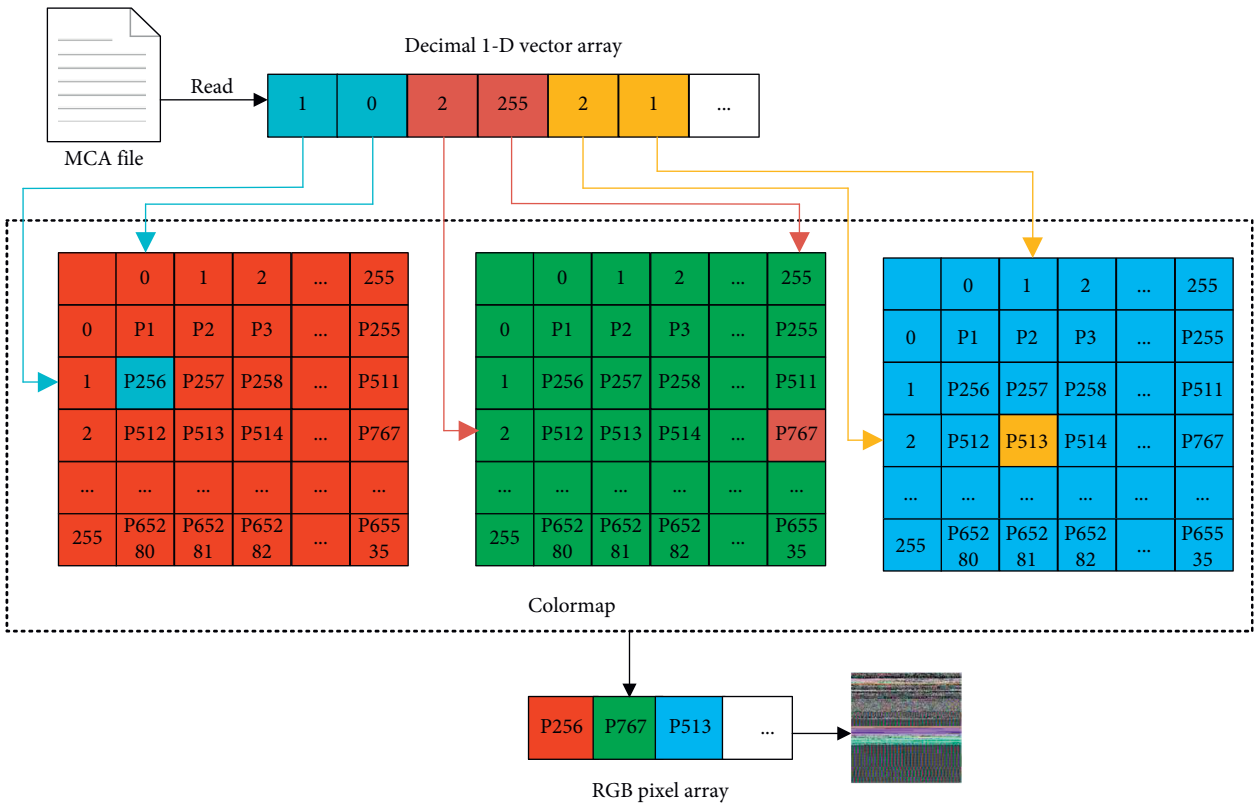

Figure 4: Principles of RGB image generation.

Table 1: The relationship between the width of the image and the size of the MCA file.

MCA file size (KB)	Image width
Between 0 and 10	32
Between 11 and 30	64
Between 31 and 60	128
Between 61 and 100	256
Between 101 and 200	384
Between 201 and 1000	512
Between 1001 and 1500	1024
Greater than 1500	2048

calculate the interpolation value $s(x)$ at the window, as shown in the following formula:

$$S(x) = \sum_{i=x-a+1}^{x+a} s_i L(x-i). \tag{2}$$

From one-dimensional interpolation to two-dimensional interpolation, the weight calculation formula is shown in the following formula:

$$L(x, y) = L(x)L(y). \tag{3}$$

When interpolating the image, we set $a = 4$, and the window size is 8Œ8, which means that 64 pixels are used to represent the target pixel. The interpolation formula is shown in the following formula:

$$S(x) = \sum_{i=x-a+1}^{x+4} \sum_{j=y-a+1}^{y+4} s_{ij} L(x-i)(y-j). \tag{4}$$

In our experiments, we perform image processing on two datasets with 3 interpolation algorithms. The specific results are shown in Table 2. The time overhead of these three interpolation algorithms is not very different in our experiments. The Lanczos interpolation algorithm is the slowest to process images, but the image quality is the best. And the time overhead of the least expensive bilinear interpolation algorithm and the most expensive Lanczos interpolation algorithm is less than 2 minutes. Therefore, in order to preserve the relevant features of the original image as much as possible to achieve better detection results, this paper uses the Lanczos interpolation algorithm to resize the image.

4.3. Deep Learning. In order to confirm whether the MCA file effectively contains the performance characteristics of the malware in the memory, we need to find a specific method to classify the obtained grayscale image and RGB image data sets, respectively. In recent years, due to the elimination of a lot of feature engineering work, there have been more and more malware classification methods based on malware images and deep learning [31]. The current mainstream neural network models (such as VGG16, VGG19, and Inception-ResNet-v2) have high detection rates

Table 2: Interpolation algorithm time cost comparison.

	Bilinear interpolation (second)	bicubic interpolation (second)	Lanczos interpolation (second)
Grayscale	400.2181	442.9408	478.7009
RGB	381.6803	392.2938	417.8827

on the ImageNet dataset [32], which consists of 14 million labeled images.

We need to choose a suitable mainstream deep learning model to classify the dataset. The purpose of this step is to confirm whether the dataset is valid in the malicious detection process. We found that most models have a high detection rate for our dataset. In order to reduce the time overhead, we choose the VGG16 network model with a small number of layers as the basis. And we make adaptive improvements to it. In order to highlight the advantages of our proposed model, we choose the V2 model with the highest detection rate as the comparison model.

4.4. Simplified Neural Network (SNN). We have designed a deep learning model (SNN) that is suitable for memory image classification. The model is improved based on the vgg16 model. The specific architecture of SNN is shown in Figure 5.

The size of the convolution kernel in the SNN network structure is 3Œ3, and the step size is 1. And we use multiple continuous 3Œ3 small convolution kernels instead of large convolution kernels. The dashed box in the figure is a convolution block. The pooling layer uses the maximum pooling with a window size of 2Œ2 and a step size of 2. Because the memory image does not have a clear object or target, the depth of the network model is too deep to easily cause overfitting, and we reduce the depth of the network to improve the accuracy of detection. And it can also reduce the time overhead. In addition, we use global maximum pooling instead of the fully connected layer using the activation function. There are some benefits of using global maximum pooling. Global maximum pooling is to take the global maximum value of the feature map as the output. It does not take the maximum value in the form of a window but takes the maximum value in the unit of the feature map. It can reduce the parameters, avoid overfitting, and improve the generalization ability of the network model.

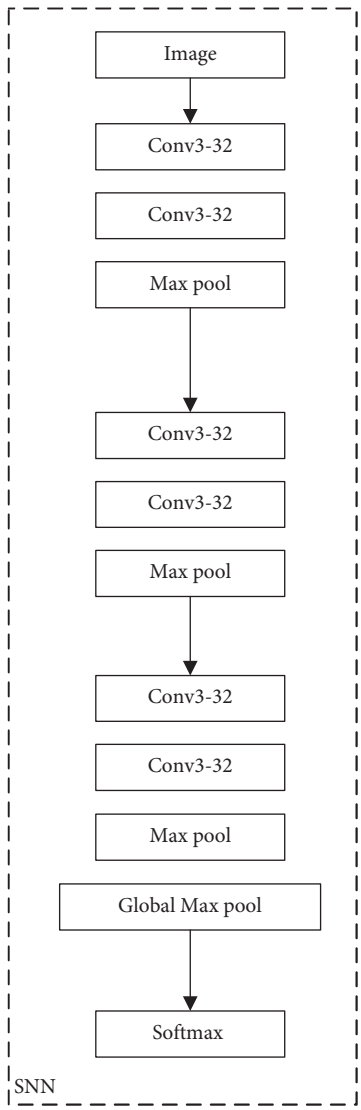

Figure 5: SNN architecture.

5. Dataset Creation and Experiment

5.1. Dataset Creation. One challenge of malware research is the lack of reference malware datasets [33]. Currently, some malware datasets are created based on static or dynamic analysis. However, the dataset is subject to the same limitations of analytical techniques. Datasets containing weak and irrelevant features may negatively affect the training of machine learning classifiers. In addition, it makes the method easily manipulated by malware evasion techniques.

In order to verify the effectiveness and feasibility of the detection method, we used a large number of real malware samples. In order to ensure the real-time and diversity of the training samples, we directly connected the customer's virtual machine to the Internet. We have collected 220 normal executable software including browsers, video players, compression software, music players, Office software such as Word, novel accessing software, and other common software. When a normal program is running, we will perform related operations on it, such as opening a word document, playing music, and playing video. In addition, we also collected 440 malware executable files samples from VirusShare. It includes viruses, Trojan, worms, adware, backdoor software, and spyware. The sample set is shown in Table 3.

We create a VM installing pure operating system without other applications as a restore point. Then, each of the above samples will run in the restored VM. While the program is running, the time intervals at which we obtain memory snapshots in order are 10 seconds, 20 seconds, 30 seconds, 1 minute, and 1 minute. Each sample program can obtain 5 memory snapshots after running in the memory. It represents the manifestation of a program at a specific time in the

Table 3: Executable program dataset.

Type	Instances
Malware	440
Normal	220

Table 4: Image dataset.

Type	Instances grayscale	Instances RGB
Malware	2200	2200
Normal	1100	1100

memory. Therefore, we create a dataset containing 3300 memory snapshots, which includes 1,100 memory snapshots of running normal programs and 2,200 memory snapshots of running malware programs.

Due to the large memory data, we use the MDIFF algorithm to reduce the dimensionality of the memory data and obtain a smaller MCA file. Therefore, we got 3300 MCA file datasets, which contains 1100 MCA files of normal programs and 2200 MCA files of malware programs. However, the MCA file is a low-level binary file; therefore, we use visualization technology to process it. Through visualization technology, we convert MCA files into grayscale images and RGB images, respectively. Then, this paper adjusts the converted picture pixels to $224 * 224$. Finally, we obtained 3300-grayscale-image dataset and 3300-RGB-image dataset. It contains 2,200 malware MCA files images and 1,100 normal MCA file images, respectively, as shown in Table 4.

In this paper, the grayscale image dataset and RGB dataset are divided into training set, validation set, and test set according to 8 : 1 : 1.

5.2. Experimental Environment. The method is deployed on the hardware environment of GeForce Titan XP GPU (video memory: 12 GB), Intel Xeon E5-2600 CPU, 64G RAM, and 4TB HDD. We use Xen 4.9.1 to build a virtualized environment. The Ubuntu 16.04 64-bit operating system acts as a secure virtual machine (Domain0), and the Windows 7 Professional 64-bit operating system, 2G RAM, and 200G hard disk acts as a VM. We use the Xen tools to directly dump the memory of the VM.

5.3. Experimental Design. Our experimental design aims to evaluate the classification effect of our proposed classification algorithm on the dataset we generated. In order to evaluate the effect of the MDIFF algorithm, we use the mainstream deep learning model (Inception-ResNet-v2) to classify the grayscale image and RGB image converted from the memory modification file obtained by the MDIFF algorithm.

In order to improve the accuracy and efficiency of detection, we tested the above-mentioned datasets separately for our proposed SNN model. Then, we compare them with the results of Inception-ResNet-v2 model.

6. Evaluation

6.1. Evaluation Metrics. In our evaluation, we evaluate our proposed image dataset and our framework's detection capabilities. In addition to using the metric Accuracy (see (5)) to judge the classification performance of the classifier, the TPR (see (6)) and FPR (see (7)) measure the proportion of correctly identified positives and measure the proportion of negatives incorrectly classified as positives of the total number of negatives, respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \tag{5}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{6}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \tag{7}$$

In the experiment of this paper, the malware memory image is a positive sample, and the normal memory image is a negative sample. The confusion matrix is shown in Table 5.

6.2. Results. In order to prove the feasibility of the MDIFF method and SNN model proposed in this paper, the mainstream deep learning model is tested on the processed image dataset. We chose Inception-ResNet-v2 convolutional neural network models to compare them with the SNN model proposed in this paper. The reason why we chose the Inception-ResNet-v2 model for comparison experiments is worth noting. The Inception-ResNet-v2 model is a pre-trained model provided in Keras. This model is one of the best performing models in the field of image classification in recent years.

First of all, we set up a comparative experiment of two neural network models. When using a convolutional neural network for classification, the training time cost of the model has a greater impact on the overall time cost. Therefore, we compared the training costs of several models. Before comparing the time cost, we must first introduce a hyper-parameter (batch_size) that affects the training time. For the same model, the larger the batch_size setting, the shorter the training time. But it has higher requirements for the graphics card. Therefore, when setting the parameters, we set the largest batch_size much as possible. The specific values of this parameter for different models are shown in Table 6. It can be seen from the table that the batch_size of the Inception-ResNet-v2 model is set to 30. This is because the model has many parameters, which can be set to a maximum of 4 in our experimental environment. The time in Table 6 is the average time for training an epoch. It can be seen that the more complex the network model structure, the longer the time consumed.

We use the TensorBoard tool to directly visualize the changes in the accuracy of the model training process (see Figure 6). The vertical axis of the graph is the accuracy, and the horizontal axis is the number of batches passed. The effects of the two models will stabilize after being trained for 16 epochs. And the Inception-ResNet-v2 model will have a

Table 5: Confusion matrix.

	Predicted as malware	Predicted as normal
Malware	TP	FN
Normal	FP	TN

Table 6: Different model hyperparameter settings and training time.

	batch_size	Training time (s/epoch)
Inception-ResNet-v2 grayscale	30	39.8
Inception-ResNet-v2 RGB	30	35.3
SNN grayscale	30	9.6
SNN RGB	30	9.4

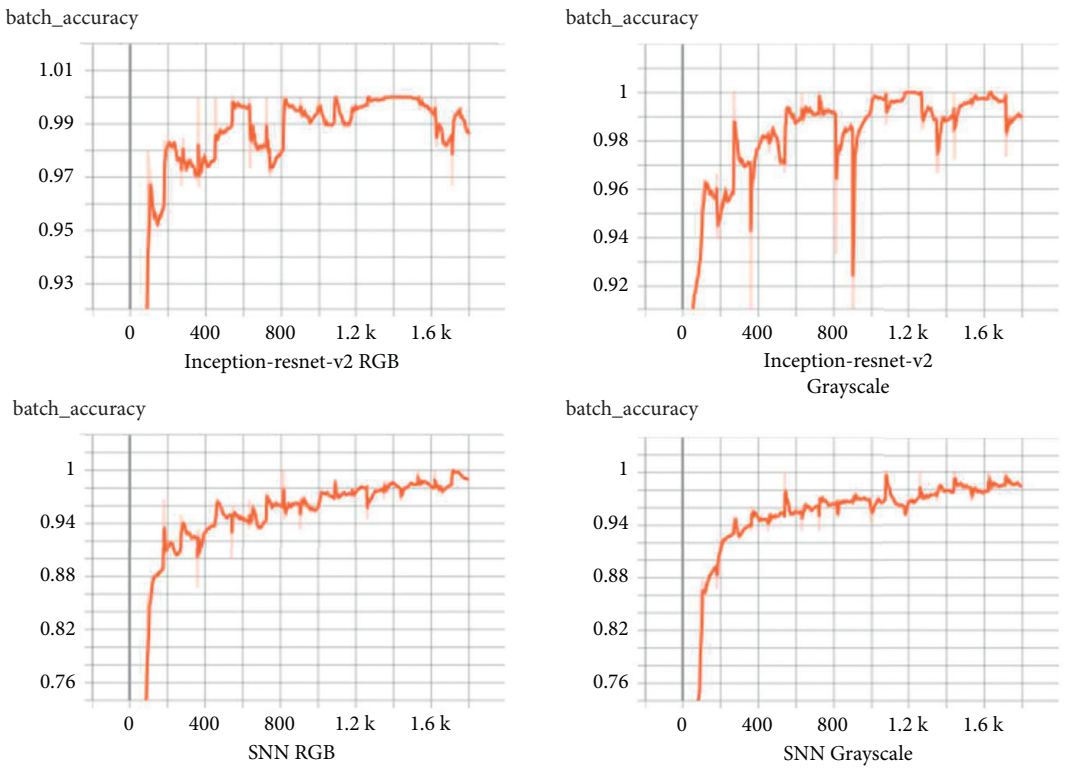

Figure 6: The accuracy rate change graph of the model training process.

certain degree of overfitting as the training epoch increases. The overall time cost and accuracy of the SNN model are better than those of the Inception-ResNet-v2 model. And the performance of RGB is better than that of grayscale images.

The model proposed in this paper has achieved 97.63% and 99.96% accuracy on the grayscale image and RGB verification set, respectively, which is better than the 94.35% and 98.96% of Inception-ResNet-v2.

It can be seen from the above that the feature set extracted by the MDIFF algorithm has outstanding performance in malware detection. The structure of the Inception-ResNet-v2 model is more complicated than that of the classification model proposed in this paper, but the classification effect is worse. The main reason for this result is that the memory image has no clear objects or targets. In the learning process of complex network structure, it is possible to learn the unique characteristics of a certain picture rather than the common characteristics of such samples. Because of the huge network parameter, the Inception-ResNet-v2 model is easy to overfit. In summary, the MDIFF algorithm proposed in this paper has outstanding performance in extracting memory features. The model SNN performs better than traditional neural networks in classification. In addition, the detection effect of RGB images is better than the effect of grayscale images.

After the experiment, the specific data of the results are shown in Table 7. In summary, the SNN model architecture proposed in this paper has a higher detection rate than the Inception-ResNet-v2 model in terms of detection rate. And

Table 7: TPR and FPR of different classifiers on different feature type.

	TPR	FPR
Inception-ResNet-v2 grayscale	0.9484	0.1938
Inception-ResNet-v2 RGB	0.9787	0.0568
SNN grayscale	0.9606	0.1266
SNN RGB	0.9939	0.0259

the RGB picture has a higher detection rate than the grayscale picture.

7. Conclusion

In this paper, in order to achieve credible detection, we propose a credible malware detection method based on the VMCADR method that does not affect user privacy. In this work, when the malware process is active, we use the Xen tools to obtain a continuous memory snapshot of the VM running test program after a specific time interval. The whole process is credible, because we obtain the VM memory dump from outside of the VM running test program. And it is not interfered by malware running in the VM.

Because the acquired memory data is very large, we use VMCADR to process it. First, we propose an algorithm MDIFF to reduce the dimensionality of memory feature data. And the obtained data MCA file has the features that play an important role in subsequent malware detection. Because the obtained MCA file is a low-level binary file, we use visualization technology to process it. After converting the MCA file into grayscale and RGB images, we proposed our own neural network SNN model. The SNN model has the advantages of higher detection rate and lower time cost than the traditional Inception-ResNet-v2 model in detection.

7.1. Limitations. While VMCADR has an excellent detection rate on malware detection, it still has some limitations. First, the memory snapshot we get from the VM is 2 GB in our experiments. However, most of the VM memory on the cloud is larger than 2 GB. When taking a VM memory snapshot, we need to suspend the VM about 5 seconds, which may affect users. In addition, obtaining MCA files and visualization processing also require additional time overhead.

7.2. Future Work. The testing of multiple programs will be carried out in our future work. Besides, we will consider more reliable features into malware detection methods, such as hardware features of hardware counters. Because the data features generated by the underlying hardware have high reliability, it may be possible to further improve the accuracy of detection.

Data Availability

Some or all data, models, or code that support the findings of this study are available from the author Luxin Zheng upon reasonable request via e-mail.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank all of the team members and those who have helped this work. This work is supported by the National Key R&D Program of China (2016YFB0800805) and the Major Projects of Science and Technology Service Industry in Tianjin (16ZXFWGX00140).

References

[1] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid," *Nature*, vol. 171, no. 4356, pp. 737-738, 1953.

[2] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: a survey," *Journal of Information Security*, vol. 5, no. 2, pp. 56–64, 2014.

[3] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey," *Information Security Technical Report*, vol. 14, no. 1, pp. 16–29, 2009.

[4] Y. Ye, T. Li, D. Adjeroh, and S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017.

[5] L. Nataraj, S. Karthikeyan, and B. S. Manjunath, "SATTVA: spArsiTy inspired classificaTionof malware VAriants," in *Proceedings of the 3rd ACM Workshop on InformationHiding and Multimedia Security*, pp. 135–140, ACM, Portland Oregon, USA, 17 June 2015.

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

[7] D. Nahmias, A. Cohen, N. Nissim, and Y. Elovici, "Trustsign: trusted malware signature generation in private clouds using deep feature transfer learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Budapest, Hungary, 14 July 2019.

[8] D. Nahmias, A. Cohen, N. Nissim, and Y. Elovici, "Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments," *Neural Networks*, vol. 124, pp. 243–257, 2020.

[9] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Ng, "Manning Convolutional-recursive deep learning for 3D object classification," *Advances in Neural Information Processing Systems*, vol. 25, pp. 656–664, 2012.

[10] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, ACM, Helsinki, Finland, 5 July 2008.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning"*, The MIT Press, Cambridge, MA, 2016.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[13] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[14] H. Yang, C. Yuan, B. Li et al., "Asymmetric 3D convolutional neural networks for action recognition," *Pattern Recognition*, vol. 85, pp. 1–12, 2018.

[15] Y. Qiao, Q. Jiang, Z. Jiang, and L. Gu, "A multi-channel visualization method for malware classification based on deep learning," in *Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 757–762, IEEE, Rotorua, New Zealand, 5 August 2019.

[16] 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, ACM, Pittsburgh Pennsylvania, USA, 20 July 2011.

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

[18] H. Naeem, F. Ullah, M. R. Naeem et al., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, Article ID 102154, 2020.

[19] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security and Privacy Magazine*, vol. 5, no. 2, pp. 32–39, 2007.

[20] R. Mosli, R. Li, B. Yuan, and Y. Pan, "Automated malware detection using artifacts in forensic memory images," in *Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, IEEE, Waltham, MA, USA, 10 May 2016.

[21] R. Sihwail, K. Omar, K. Zainol Ariffin, and S. Al Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," *Applied Sciences*, vol. 9, no. 18, p. 3680, 2019.

[22] W. Huang and J. W. Stokes, "MtNet: a multi-task neural network for dynamic malware classification," in *Proceedings of the International conference on detection of intrusions and malware, and vulnerability assessment*, pp. 399–418, Springer, San Sebastián, Spain, 7 July 2016.

[23] C.-H. Lin, H.-K. Pao, and J.-W. Liao, "Efficient dynamic malware analysis using virtual time control mechanics," *Computers & Security*, vol. 73, pp. 359–373, 2018.

[24] N. Nissim, Y. Lapidot, A. Cohen, and Y. Elovici, "Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining," *Knowledge-Based Systems*, vol. 153, pp. 147–175, 2018.

[25] X. Wang, J. Zhang, J. Zhang, A. Zhang, and J. Ren, "TKRD: trusted kernel rootkit detection for cybersecurity of VMs based on machine learning and memory forensic analysis," *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 2650–2667, 2019.

[26] "DRAKVUF eb/ol," https://drakvuf.com/.

[27] Y. Jian, H. Kuang, C. Ren, Z. Ma, and H. Wang, "A novel framework for image-based malware detection with a deep neural network," *Computers & Security*, vol. 109, Article ID 102400, 2021.

[28] P. Colin, "Naive differences of executable code," 2003, http://www.daemonology.net/bsdiff/.

[29] A. Pinhero, M. L. Anupama, P. Vinod et al., "Malware detection employed by visualization and deep neural network," *Computers & Security*, vol. 105, Article ID 102247, 2021.

[30] S. Fadnavis, "Image interpolation techniques in digital image processing: an overview," *International Journal of Engineering Research in Africa*, vol. 4, no. 10, pp. 70–73, 2014.

[31] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on Markov images and deep learning," *Computers & Security*, vol. 92, Article ID 101740, 2020.

[32] L. Fei-Fei, J. Deng, and K. Li, "ImageNet: constructing a large-scale image database," *Journal of Vision*, vol. 9, no. 8, p. 1037, 2010.

[33] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: static, dynamic, hybrid and memory analysis," *International Journal of Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, p. 1662, 2018.