Hindawi
Security and Communication Networks
Volume 2023, Article ID 6390023, 15 pages
https://doi.org/10.1155/2023/6390023

Research Article

Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model

Zilin Zhao,[1] **Dawei Zhao**,[1] **Shumian Yang**,[1] **and Lijuan Xu**[1,2]

[1]*Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China*
[2]*School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China*

Correspondence should be addressed to Dawei Zhao; zhaodw@sdas.org and Shumian Yang; yangshm@sdas.org

Received 23 September 2022; Revised 1 February 2023; Accepted 28 March 2023; Published 17 April 2023

Academic Editor: Tom Chen

In recent years, malware has experienced explosive growth and has become one of the most severe security threats. However, feature engineering easily restricts the traditional machine learning methods-based malware classification and is hard to deal with massive malware. At the same time, the dynamic analysis methods have the problems of complex operation and high cost, which are not suitable for efficiently classifying large quantities of malware. Therefore, we propose a novel static malware detection method based on this study's AlexNet convolutional neural network (CNN). Unlike existing solutions, we convert all malware bytes into color images, propose an improved AlexNet architecture, and solve the unbalanced datasets with the data enhancement method. Extensive experiments are performed using the Microsoft malware dataset and the Google Code Jam (GCJ) dataset. The experimental results show that the accuracy of the Microsoft malware dataset reaches 99.99%, and the GCJ dataset reaches 99.38%. We also verify that our method can better extract the texture features of malware and improve the accuracy and detection efficiency.

1. Introduction

Malware software can infect computers or devices without the consent of users. Through these loopholes, criminals carry out various illegal and criminal acts, infringing on the country's and nation's legitimate rights and interests. According to the "China Network Security Report 2021," a total of 119 million virus samples were intercepted by Rising's "Cloud Security" system, with 259 million virus infections found, and the overall number of viruses declined relative to 2020. However, ransomware and mining viruses are still not to be underestimated. At the same time, Trojan viruses have become the most significant number, followed by worm viruses, accounting for more than 80% [1, 2]. Although large numbers of malicious samples can be processed using machine learning methods, it takes many human and material resources to identify and classify malware by feature engineering. We must face the challenge of reducing network security risk, rapid and accurate classification, and malware detection.

Traditional malware classification methods can be divided into two categories: static analysis (Gibert et al. [3], Seo et al. [4], Jeon and Moon [5], Shalaginov et al. [6], and Zhao et al. [7]) and dynamic analysis (Lin et al.[8], Sun et al. [9], Htun et al. [10], Bidoki et al. [11], and Kim et al. [12]). In general, static analysis methods do not require running malware binary samples and helpful information can be obtained directly through disassembly, such as functions, string lists, and hash values. With short consumption time, the static analysis methods have the advantages of simple operation and a high accuracy rate. However, static analysis methods can only analyze malware binary samples from the surface, easily affected by confusion techniques such as deformation. It is also difficult to detect and classify unknown malware. Dynamic analysis methods can run in a virtual environment and are not affected by obfuscation techniques. It can observe the dynamic changes of the malware binary samples in time and can identify new malware samples. However, it is a very time-consuming and complex operation.

Malware classification methods based on machine learning (Gibert et al.[13], Woźniak et al. [14], Li et al. [15], Ucci et al. [16], and Liu et al. [17]) must extract features artificially and then select the appropriate classification model. It is suitable for smaller datasets, relatively simple to operate, and can be run on low-end machines. With the increasing numbers and types of malware, machine learning methods have gradually exposed some weaknesses, such as being susceptible to feature engineering and difficulty dealing with large amounts of malware. To solve the above problems, malware classification methods based on deep learning (Amer and El-Sappagh [18], Zhao et al. [19], Kalash et al. [20], Wang and Wang [21], and Vasan et al. [22]) are widely used. It changes traditional machine learning that relies on hand-crafted features. These features must be based on expert knowledge and experience to construct a representation of malware behavior and then classify malware, which is time-consuming and may not be well generalized to new malware. However, deep learning solves the problems of the complicated construction of features and manual participation.

Recently, researchers have studied classification methods based on malware visualization (Vinayakumar et al. [23], Vasan et al. [24], Naeem et al. [25], and Zhao et al. [26]), malware classification based on images, and deep learning has become an effective solution. By generating images, we can observe malware more intuitively without requiring knowledge of the related fields. Most visualization methods require images to be uniformly sized and then fed into the neural network model. Nataraj et al. [27] applied the visualization method to malware for the first time. Firstly, read the unsigned integer vector of the binary file, and then every 8 bits were a unit and converted it into a decimal value between 0–255. Finally, the width is fixed and the height is set according to the file size to generate grayscale images. Although there is much research on malware visualization, there are still many problems to be solved, such as redundancy or loss of information, low classification accuracy, and difficulty detecting new malware variants.

The Markov model (Alipour and Ansari [28] and Yuan et al. [29]) has also made some achievements in image processing. According to previous studies, the malware method based on the Markov model has better detection and classification effects. Our solution uses the Markov model to generate the transition probability matrix. Ultimately, malware images of fixed size can be generated directly, reducing the loss of information. The approach based on a combination of deep learning and malware visualization avoids the high overhead of manual feature extraction, improving malware feature extraction capability and model classification performance.

This study proposes a framework combining images with deep convolutional neural networks (CNNs) for malware classification, which can effectively and efficiently solve the problem of malware detection and variant recognition. First, this method uses the Markov model and z-score standardization method to visualize malware as images. Then, it uses colormap to mark images to generate color images. Finally, the malware colormap images are learnt and classified using an improved convolutional neural network model. We compare the performance of the proposed method on two benchmark datasets, and the experimental results show that the classification performance results of the method are significantly improved.

The main contributions of this study are summarized as follows:

(i) A novel malware image generation method is proposed, which can effectively retain the relevant information of the binary files and reduce the redundancy and loss of information, and the image generation does not require the execution or disassembly of any malware code.

(ii) We combine batch normalization and the improved CNN, which enhances the model's generalization ability and improves the accuracy of malware classification. The parameter tuning process is simplified, and the initialization requirements are reduced.

(iii) An improved CNN is introduced with fewer fully connected layers and lower output dimensions. The training time is significantly decreased, and the classification speed is improved.

(iv) This experiment is conducted on two unbalanced benchmark datasets to evaluate the model framework proposed in this article. The experimental results show that the proposed model framework achieves excellent classification performance. The data enhancement method used can effectively solve problems such as too few samples or poor quality to prevent overfitting.

The rest of the study is organized as follows: Section 2 presents related research on malware detection and classification. Section 3 details a novel framework for image-based malware classification with the AlexNet CNN model. The experimental results are presented in Section 4. Finally, Section 5 summarizes the study and future prospectives.

2. Related Work

Traditional malware classification methods were mainly based on static or dynamic analysis to obtain features, and then machine learning algorithms were used for classification. With the increasing types, quantity, and detection difficulty of malware, some drawbacks are gradually exposed. Researchers began to study the combination of visualization and deep learning of malware to improve the efficiency and accuracy of malware detection. Therefore, this study mainly introduces the following aspects of related technologies: static analysis methods, dynamic analysis methods, and analysis methods based on malware images and deep learning.

2.1. Malware Analysis Method Based on Static Analysis. The static analysis method does not need to execute malware and usually analyzes the frequency distribution of opcodes, byte sequences, etc. It has the characteristics of high accuracy and high speed. Santos et al. [30] used the n-gram method to detect unknown malware samples. Due to the large dataset, 1000 malware and 1000 benign software samples were selected separately for the experiment. Using the k-nearest neighbor algorithm, a final detection rate of 91.25% was obtained. Ahmadi et al. [31] proposed a malware classification method based on static learning, which classifies malware by extracting PE structure information. Features can be extracted directly without unpacking.

Naeem et al. [32] conducted experiments using three publicly available datasets of Windows systems, combining the extracted attribute features by capturing the local and global attributes of the gray-scale images. Finally, a machine learning approach was used to classify the malware samples. Liu et al. [33] proposed a method for malware visualization to classify malware binary files. This method was tested on three datasets, and the local features of malware samples were extracted through the bag of the visual words (BOVM) model. Compared with the global feature, the feature is more flexible, and the classification accuracy is significantly improved. However, the scheme is susceptible to obfuscation techniques and is computationally expensive. Naeem et al. [34] also visualized the malware binary file as an image, then used local and global information to extract features, and used four public malware datasets for experiments. Although the dimensionality of the two models has been reduced to reduce the computation time, the running time is still slightly longer compared to the other models. Li et al. [35] used static and statistical analysis to extract multidimensional features. Multidimensional text feature vectors were extracted by the n-gram and term frequency-inverse document frequency (TF-IDF); then, the best features were selected by a classifier. Finally, the best feature vectors were fused with other features for training using a machine learning framework.

2.2. Malware Analysis Method Based on Dynamic Analysis. Although the static analysis method consumes less time and is easy to operate, it is easily affected by obfuscation technology, and it is easy to judge malware in the form of deformation, packaging, and other ways as benign software. The dynamic analysis method is not affected by obfuscation technology and can trace the actual running path of malware, making up for the insufficiency of static analysis. Nair et al. [36] proposed a malware classification method based on dynamic analysis, which can compensate for the shortcomings of static analysis techniques. In the process of propagation, the structure of deformed malware will change, but its essential function is unchanged. Therefore, the corresponding features are obtained by dynamically tracking API calls issued by deformed malware.

Lim et al. [37] classified malware through network behavior. This scheme first clustered the extracted traffic characteristics through the K-means algorithm, compared the similarity of the sequences generated by clustering, and organized the malware behavior through the unified algorithm. Kim et al. [38] used the malware API call sequence and multiple sequence alignment (MSA) algorithm. By comparing the test API call sequence with the behavior sequence of malware samples, the malware samples were classified and detected by the dynamic analysis method. Xu et al. [39] detected malicious Windows software through the pretraining model, extracted the application programming interface (API) sequence of malware samples by combining natural language processing (NLP) with the dynamic analysis method, and then conducted experiments on two different datasets through the fine-tuning method.

2.3. Analysis Method Based on Malware Image and Deep Learning. Recently, there have been an increasing number of studies combining visualization techniques with deep learning, with most research scholars converting malware into gray-scale images. Ni et al. [40] used SimHash and CNN to classify malware and generate SimHash-based grayscale images by extracting opcode sequences from malware. Further, CNNs were used to train images and identify malware families. Xue et al. [41] obtained grayscale images through static analysis and used the CNN model with the spatial pyramid pooling (SPP) layer for classification. Then, the API call sequence is analyzed by dynamic analysis methods using variable N-grams and machine learning. Finally, static analysis and dynamic analysis are connected through probability scoring. The experiments used 63 malware families, and the results show that the preprocessing and testing time is significantly reduced, while the method's accuracy is as high as 98.82%.

With the increasing research on visualization techniques, research scholars have found that a single low-order feature representation may not be conducive to discovering hidden features in malware families. Therefore, research into multichannel based malware image classification methods has begun. Pinhero et al. [42] compared three different malware images (grayscale, RGB, and Markov) and applied them with a Gabor filter to extract relevant features. The experiment uses 12971 benign samples and two malware datasets and designs four different sizes ($32 \times 32, 64 \times 64, 128 \times 128$, and 256×256) into 12 different CNN model architectures for training. The experimental results show that a 99.97 F-measure is finally produced. Yadav et al. [43] extracted the byte code of the dex file from left to right in the sequential order, every six digits as a group, forming three or 2 digits, which were then converted into decimal values. Finally, they were mapped into R, G, and B sequentially to form a color map. Training with EfficientNet-B4 CNN achieved 98.8% accuracy in separating malware from benign software images.

Most of the above deep learning-based visualization techniques generally convert malware binaries into gray-scale images. When using the depth neural network model to train the generated gray image, it is necessary to unify the size of the image. During the conversion process, it is easy to cause redundancy or loss of information. At the same time, the training models used by most methods have problems, such as a large number of parameters and a long training time, which affect the performance of classification detection.

3. The Proposed Method

We proposed a malware classification method based on image and deep learning, consisting of feature extraction, image generation, and CNN classification. The framework of our method is shown in Figure 1. Firstly, the malware is read in binary mode and traversed all the bytes. The information from the malware binary file is extracted, and the transition probability matrix is obtained. The value of the transition probability matrix is the transition probability of each byte to other bytes. Then, the transition probability matrix is standardized, and the color map is applied to the transfer probability matrix to visualize the malware as color images. Finally, the malware is classified through an improved CNN. Different evaluation indexes often have different dimensions. If not processed, the differences in numerical values may be significant, affecting the data analysis results. To eliminate the influence of dimension and value range differences between indicators, we must carry out standardization processing and scale them in proportion to make them fall into a specific area for us to conduct the comprehensive analysis.

3.1. Image Representation of Malware.

3.1. Image Representation of Malware. The grayscale image is also called the gray level image. In most current image-based malware classification methods, the width of the generated grayscale image is generally fixed, and the height is set according to the file size. Then, the grayscale image is directly cropped or rescaled to a uniform size and put into the model for training. However, such malware image generation is often associated with information missing or redundant problems, resulting in high error rates and low accuracy in the final classification. Therefore, we propose a simple and effective scheme to visualize the original malware binary file as color images to solve such a problem. No feature engineering is required, and the malware information can be effectively retained. The process of the proposed method is shown in Figure 2.

We read the malware file in the binary mode. In this way, each binary malware can be regarded as a byte stream $B = \{b_1, b_2, \ldots, b_N\}$, where n represents the number of bytes in the malware. Since the value of each byte ranges from 0 to 255, the byte stream is equivalent to a decimal integer stream $B' = \{\ldots, k, m, \ldots\}_n$, where $0 \leq k, m \leq 255$. Here, we call two adjacent bytes the byte pair and denote $f(k, m)$ as the number of byte pair (k, m) in B'. Then, each malware can be converted to a transition probability matrix, with a size of 256×256, denoted by the $P = \{P_{km}\}_{256\times 256}$. P_{km} refers to the value of the kth row and mth column of P, which is given by

$$P_{km} = \frac{f(k,m)}{\sum_{m=0}^{255} f(k,m)}. \tag{1}$$

Matrix P contains the byte information of the malware while ignoring the frequency difference between different columns. The z-score can be applied to numerical data, and the calculation is simple. It can convert data of different magnitudes to the same extent so that the data are comparable and used to generate z-score standardized images. Therefore, the normalized transition probability matrix can be obtained.

The data are normalized by the column to process the transition probability matrix, where the byte sequence $X = \{x_1, x_2, \ldots, x_n\}$ of a particular column; the normalized values y_{ij} in the ith row and jth column are shown in the following equation:

$$P'_{km} = \frac{P_{km} - \overline{X}}{\mathrm{std}(X)}, \tag{2}$$

where the mean value $\overline{X} = 1/256\sum_{m=0}^{255} P_{km}$ and the standard deviation $\mathrm{std}(X) = \sqrt{1/256\sum_{m=0}^{255}(P_{km} - \overline{X})^2}$.

The color map is a three-dimensional real number matrix. The color map represents a mapping (color mapping), which is not a continuous function type mapping but a matrix with three columns representing the color's R, G, and B components. The color map matrix can be generated manually or defined by calling the functions provided by MATLAB. By customizing the color arrangement order, after several sets of experiments, determine the appropriate color order. Finally, we apply color map to the generated standardized image and visualize it as malicious sample color images.

3.2. Data Augmentation. In the process of deep learning, when we train the model, if the number of samples is too small or the quality of examples is not good, it is prone to overfitting. In general, the more the number of samples, the better the effect of the trained model. By increasing the number of samples or improving the quality, the problem of sample imbalance can be solved, and the dependence of the model on some characteristic attributes can be reduced to improve the model's generalization ability. However, if the number of our samples is too small or the quality is not good, it can be processed by data enhancement technology, which improves the robustness of the model.

Data enhancement is a technology to expand the number of samples. The existing data becomes rich and diverse by increasing the number of instances. Data enhancement techniques can be divided into two categories: offline data enhancement and online data enhancement. The offline data enhancement method is suitable for the case of small datasets and directly processes the datasets. When the dataset is large, the offline data enhancement method will consume much space, so this study uses the online data enhancement method. Before each epoch, the original data image will be transformed. Each method contains random factors, so the data used for model training differs each time. That is to say, how many epochs have been experienced and how many times the data have expanded.

In general, we entirely use the limited data through geometric and color transformations of the original image. The data enhancement methods used in our experiments are shown in Table 1:

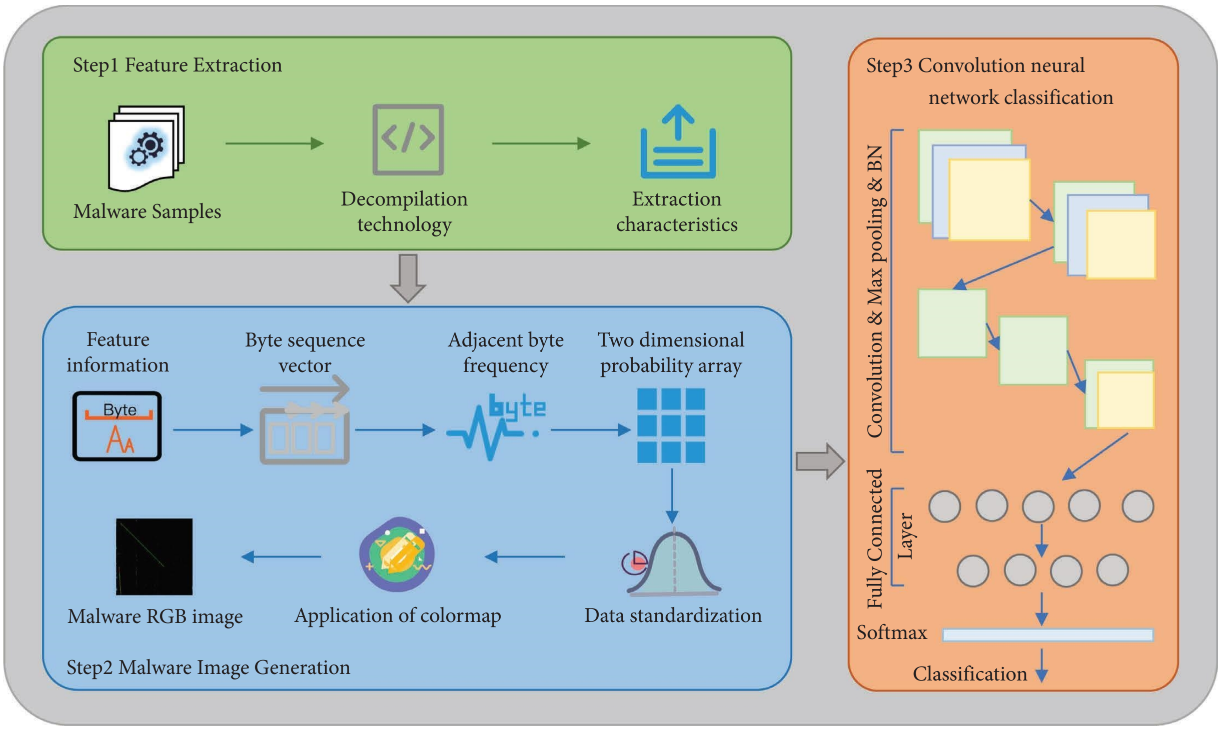


Figure 1: The proposed program framework.

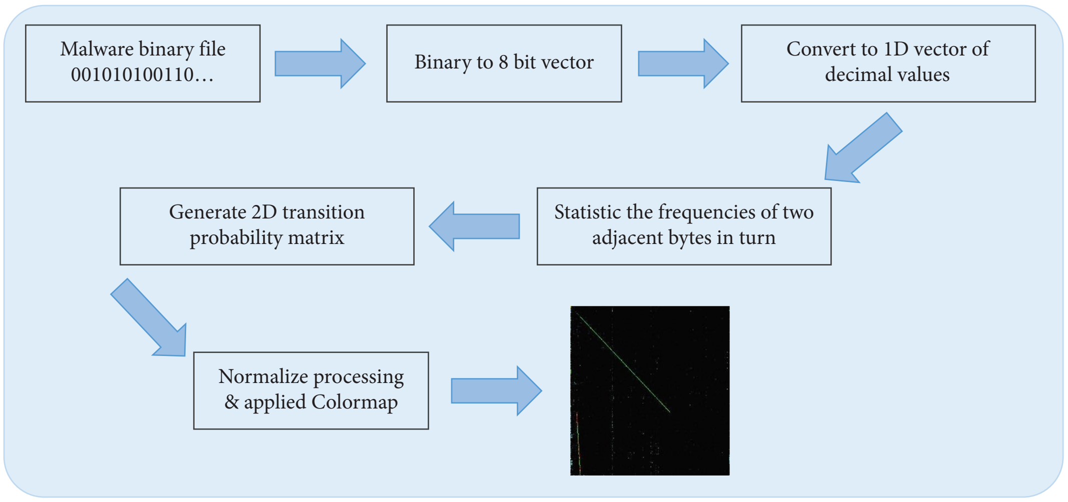

Malware binary file 001010100110…
Binary to 8 bit vector
Convert to 1D vector of decimal values
Statistic the frequencies of two adjacent bytes in turn
Generate 2D transition probability matrix
Normalize processing & applied Colormap

Figure 2: Malware image generation process.

Table 1: The setting of the data enhancement method.

Transform methods	Parameter setting
Width shift	0.0
Height shift	0.0
Fill mode	None
Rotation range	0.0
Zoom_range	0.0
Image transformation	Contrast, saturation
Data standardization	Normalize

3.3. Batch Normalization. With the deepening of the neural network, in the training process, the change of the parameters of the previous layer constantly influences changes in the input distribution of the next layer. As the input of each layer is no longer independent and identically distributed, the upper network needs to adapt to these distribution changes constantly, and training becomes increasingly complex. The convergence rate will gradually become slow, resulting in the internal covariate shift problem.

In the neural network, each layer of network M can perform two operations of the linear transformation $Y_M = W_M \cdot X + b_M$ and nonlinear transformation $F_M = G_M(Y_M)$, where W_M represents the weight of the M layer in the neural network, b_M represents the offset vector of each neuron on the M layer, and G_M represents the activation function of the M layer. In the process of reverse propagation, according to the gradient descent method to update the W_M and b_M of each layer, the distribution of Y_M will change and the distribution of F_M will also change. However F_M is used as input to the next layer $(M+1)$, which makes the neurons in the next layer also need to constantly adapt to such changes, which reduces the speed of convergence of the whole network. It becomes more severe as the number of network layers deepens.

To reduce the internal covariate shift problem in neural networks, we introduced the batch normalization algorithm in model training, which was proposed by Ioffe and Szegedy [44] in 2015. Through this method, each dimension in the feature map corresponding to each batch of data is standardized in the output of each layer to accelerate the network's convergence and improve its accuracy. This method normalizes the output at the network's each layer for each dimension of the feature map corresponding to each batch of data, accelerating the convergence of the network and improving the accuracy. By calculating the mean and variance of all data for the same channel in each batch, the normalization method requires the results to be subtracted from the mean and divided by the standard deviation before the linear calculation is fed into the activation function. The input of each layer of the neural network is guaranteed to conform to the standardized normal distribution with the mean μ of 0 and the variance σ^2 of 1 to reduce the difference of samples. The standardized normal distribution formula is shown in equation (3). Backpropagation learning is obtained by batch normalization to ensure the ability of nonlinear expression, scale parameters, and shift parameters. Two parameters are added to each neuron. The scale and shift change operations were performed on $\hat{x}$ that satisfies the normalized normal distribution after the transformation, as shown in equation (4).

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left((x+\mu)^2/2\sigma^2\right)} \longrightarrow P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(x^2/2\right)}, \tag{3}$$

$$BN_{\text{scale,shift}}(x) = \text{scale} \cdot \hat{x} + \text{shift}. \tag{4}$$

By using batch normalization, the parameter adjustment process is simplified and the requirement for initialization is reduced. The larger the learning rate, the larger is the step size of the parameter update, which is prone to oscillation and nonconvergence. The batch normalization layer is generally placed between the convolutional layer and the ReLU layer. Using this algorithm will make the network not affected by the parameter values, and the network training is more stable. Even if the network uses a higher learning rate, it will not cause the problem of disappearance and explosion of the gradient. Using the batch normalization algorithm, we can remove dropout and L2 regularization parameter selection problems and make each layer's normalized mean and variance different, increasing the model's robustness and resisting overfitting. The ability to generalize the network is improved, while the constant adjustment of parameters can be reduced. Through standardization and linear transformation, batch normalization makes data maintain the identical distribution in the training process while achieving decoupling between the various layers in the network. The network of each layer can learn independently, which is conducive to accelerating the training and convergence speed of the model. Each layer of the network can learn independently, which is conducive to accelerating the training and convergence of the model.

3.4. Classification of Malware Images. In recent years, CNNs have developed rapidly and have achieved significant breakthroughs in computer vision, speech recognition, face recognition, and natural language processing. CNNs are a deep learning method that Yann LeCun first proposed. This study designs a CNN architecture based on the AlexNet network, as shown in Figure 3. The network was first proposed by Krizhevsky et al. [45] in 2012. The AlexNet deep convolution network structure was first applied to large-scale image datasets in the ImageNet 2012 competition. The graphics processing unit (GPU) was introduced to speed up model training. The ReLU activation function and local response normalization (LRN) were used.

We put 256×256 malware images into the model for training. The CNN designed in this study has five convolution layers and three pooling layers, similar to the AlexNet network architecture. The difference is that the PReLU activation function is used in each convolutional layer, and the use of local response normalization is eliminated. We set the number of convolution kernels in the model to half the original number, and the proposed model architecture consists of two fully connected layers. The last full connection layer is directly connected to the output layer and the category classification by the softmax function. A new deep CNN model is constructed using the PyTorch library to write the model code and add a batch normalization layer to the model.

The network model is trained in the training phase using a cross-entropy loss function, essentially a log-likelihood function. The learning model parameters of the Adam optimizer are used to train the network model. The deep CNN architecture for malware image classification has enhanced model generalization ability, lowered output dimensions, and reduced model parameters compared with the original AlexNet architecture. Compared with other traditional neural networks, effective features can be extracted with a smaller amount of parameters. At the same time, much time can be saved and space consumption is also reduced.

4. Experiments and Analysis

4.1. Dataset and Experimental Settings. This experiment is conducted on the two datasets, the Microsoft dataset and

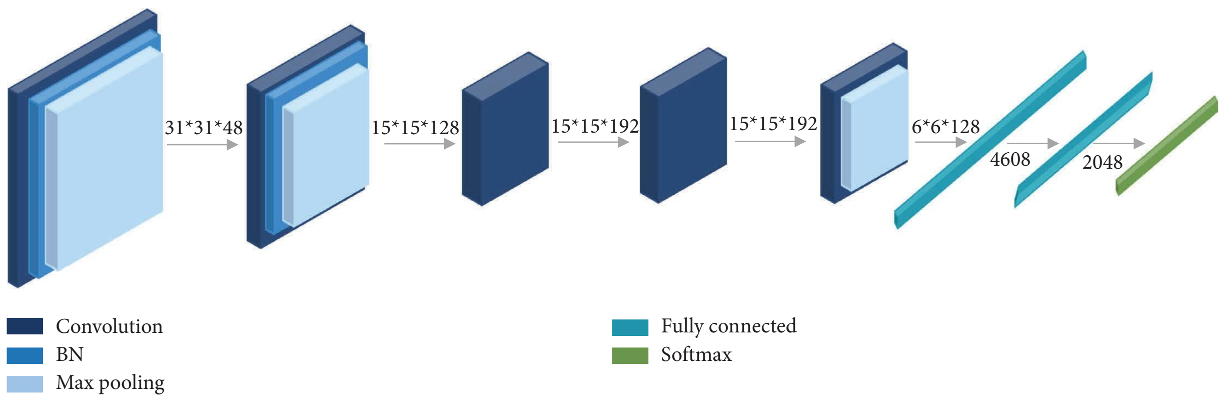


Figure 3: Proposed neural network architecture.

Google Code Jam (GCJ) dataset. For the Microsoft Malware Classification Challenge (BIG 2015) dataset, we use 10,868 labeled malware samples from 9 families. Generally, the assembly code of the malware binary file is obtained by decompilation, while the Microsoft dataset has been processed in advance and can be used directly. The malware distribution is shown in Table 2. GCJ is hosted by Google and is an international programming competition. Participants in the competition can use any programming language but are required to solve algorithmic problems within a specified time frame. We collected programming languages from ten years of GCJ projects from 2010 to 2019 and analyzed 1600 authors, with the number of codes per year shown in Figure 4.

The experimental program was written in python, using Keras 2.4.3 and PyTorch 1.4.0 as the backend. Hardware environment includes processor: R7-5800H; 8 cores 16 threads; core graphics card: core AMD Radeon^{TM} Graphics; discrete graphics card: GTX1650; and video memory: 4G. We divide the dataset into training and test sets, in which the training set accounts for 90% and the test set accounts for 10%. The generated malware sample images are 256×256 and then input into the designed network model for training. According to different malware samples, different batch sizes, training cycles, and learning rates are set. In this experiment, the batch size is set to 128, the training period is set to 50, and the learning rate is set to $1e^{-4}$.

Table 2: Number of samples for each malware family in the Microsoft dataset.

ClassID	Families	Samples
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_Ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_Ver1	398
8	Obfuscator.ACY	1228
9	Gatak	1013
		10868

4.2. Evaluation Indicators.

4.2. Evaluation Indicators. The experiment uses accuracy, F-measure, precision, and recall as evaluation metrics. Accuracy is the total percentage of samples predicted to be correct, precision is the proportion of samples predicted to be positive to those actually positive, recall is the proportion of samples actually positive to those predicted to be positive, and F-measure is the harmonic average of accuracy and recall. If the malware samples are successfully classified as malware, they are called true positive (TP); if misclassified as benign software, it is called false negative (FN). If the benign software samples are successfully classified as harmless software, they are called true negative (TN); if misclassified into malware, it is called false positive (FP). The formula is shown as follows:

$$\begin{aligned}\text{accuary}\,(A) &= \frac{TP+TN}{TP+TN+FP+FN},\\ \text{precision}\,(P) &= \frac{TP}{TP+FP},\\ \text{recall}\,(R) &= \frac{TP}{TP+FN},\\ F-\text{measure}\,(F) &= \frac{P\times R}{P+R}\times 2.\end{aligned} \tag{5}$$

4.3. Experimental Results

4.3.1. The Influence of Image Generation Methods on Classification Accuracy. This experiment compares standardized images, Markov images, and the method proposed in this study to generate images. The variation of the accuracy with epochs for the training set and the test set is shown in Figure 5. The abscissa represents the training period, which we set to 50, and the ordinate represents the training set's or test set's accuracy. We find that with the increase of epochs, the accuracy keeps increasing, and when the training reaches a certain level, it gradually stabilizes. By observation, the proposed method for generating malware images outperforms the other two methods. When tending to stabilize, the accuracy fluctuates between 99% and 100%. Figure 6 shows the change of the loss value of the training set and the

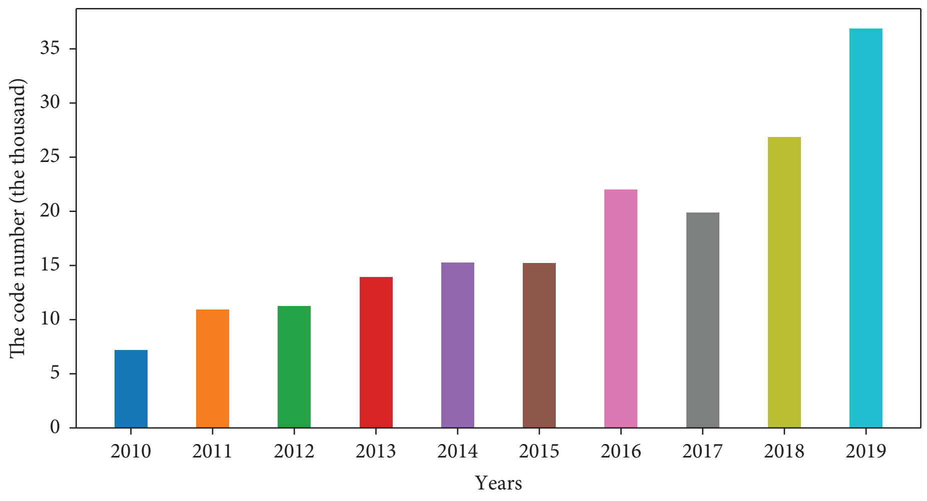


Figure 4: Details of the GCJ dataset used in the experiment.

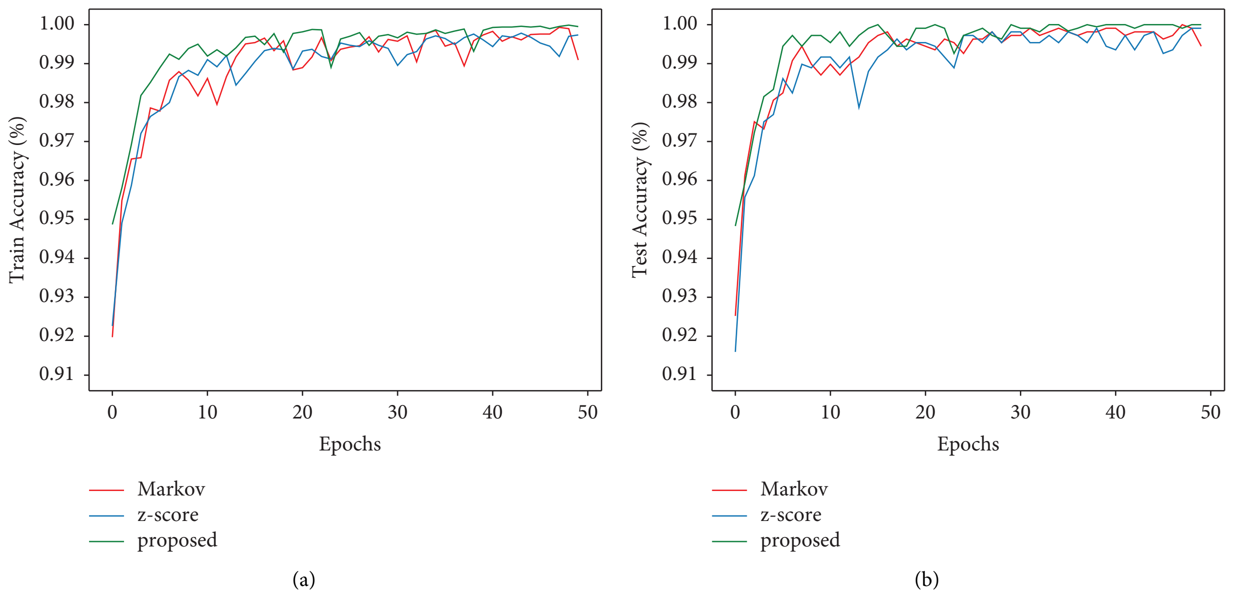
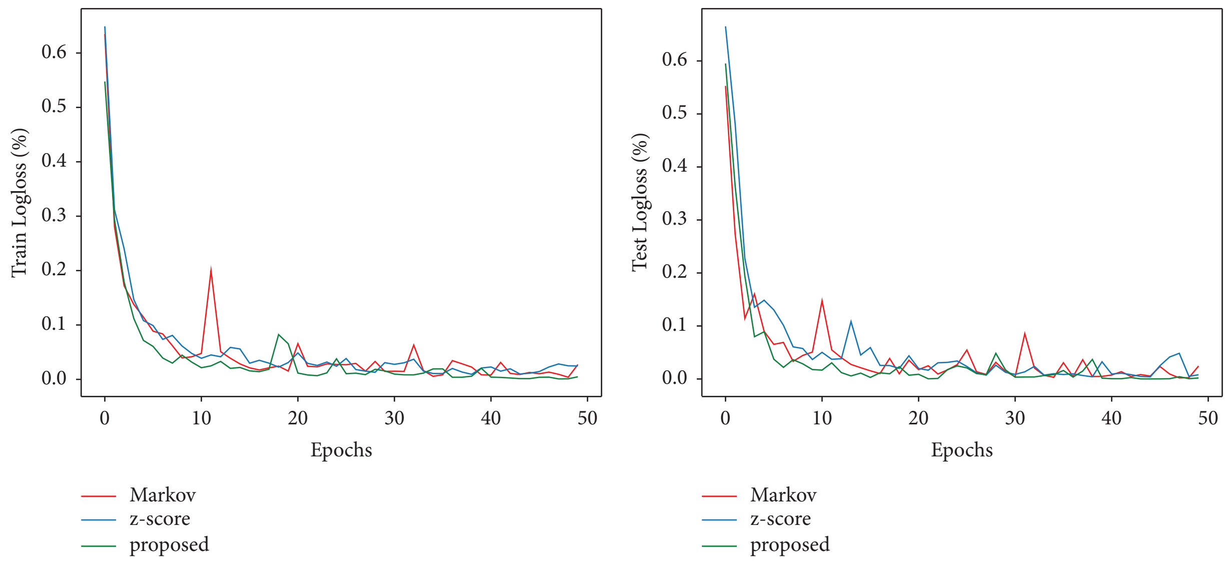


Figure 5: The changing trend of accuracy with training epochs on the Microsoft dataset: (a) train accuracy and (b) test accuracy.

test set with epochs, where the abscissa represents the training period, which is still set to 50. The ordinate represents the loss value of the training set or the test set. As the number of iterations increases, the loss value converges rapidly, gradually decreases, and stabilizes. It can be seen from the observation that the loss effect of the method proposed in this study is better than the other two methods, and the convergence speed is also faster. By comparing the accuracy and loss values of the datasets, it is found that the effect of the test set is better than that of the training set because of the use of regularization and data augmentation methods.

The confusion matrix can be seen as an $n \times n$ table. The actual malware category is located in each row, and the malware prediction category is found in each column, which can measure the effectiveness of our model. The confusion matrix of the malware sample is shown in Figures 7, and Figures 7(a)–7(c) represent the confusion matrix of the Markov images, standardized images, and images generated by the proposed method, respectively. Through comparison, it is found that the method of generating images presented in this study has a good performance in each category and can achieve a performance of 94.23% even in a few categories.

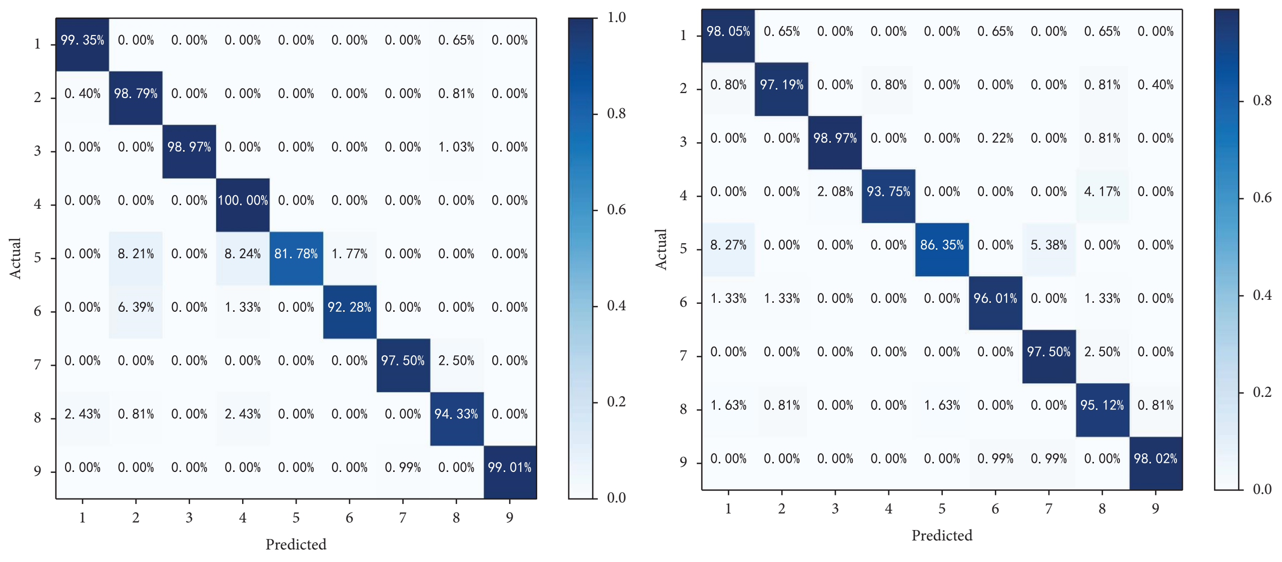
4.3.2. Performance Comparison with Unimproved Model and Data Augmentation. In this experiment, the CNN model designed in this study is compared with the CNN model before improvement and the method without data enhancement, as shown in Figure 8. Through a comparative analysis of each family's accuracy, recall, precision, and F-score, we found that compared with the other two methods,



(a)

(b)

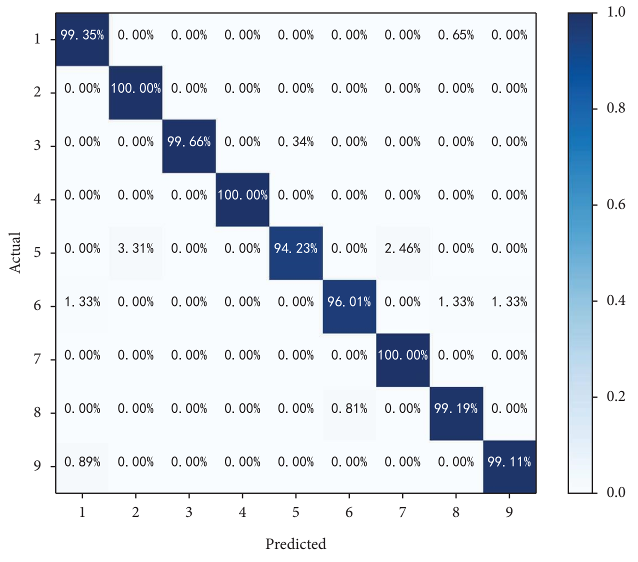
Figure 6: The changing trend of loss value with training epochs on the Microsoft dataset: (a) train logloss and (b) test logloss.



(a)

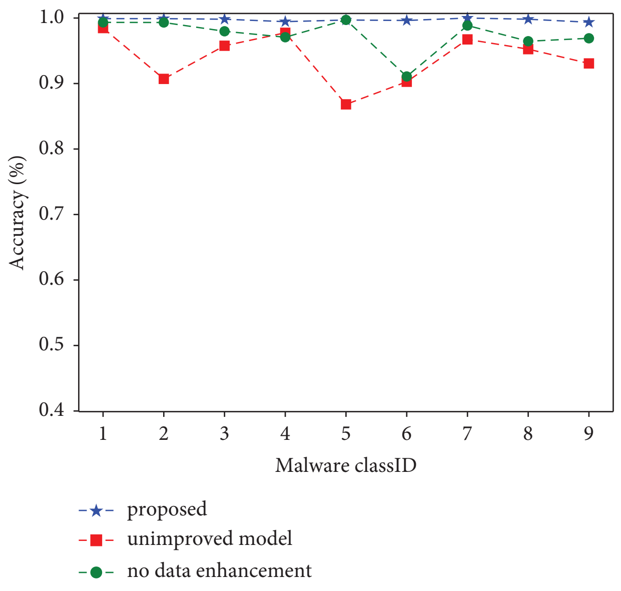
(b)

Figure 7: Continued.

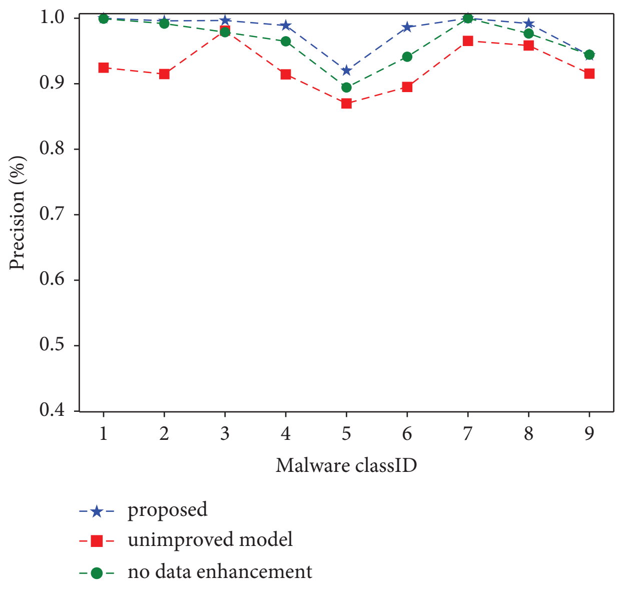


(c)

Figure 7: Confusion matrix. (a) Markov. (b) *Z*-score. (c) Proposed.



(a)



(b)

Figure 8: Continued.

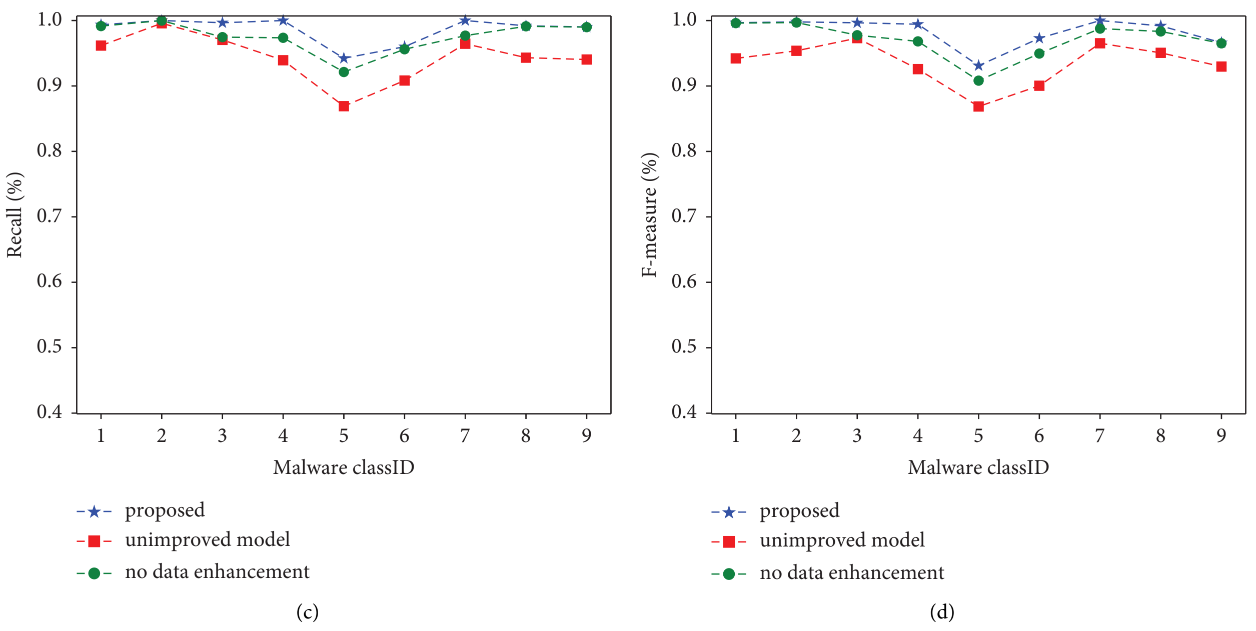


FIGURE 8: Comparative analysis of accuracy, precision, recall, and *F*-measure for each family. (a) Accuracy. (b) Precision. (c) Recall. (d) *F*-measure.

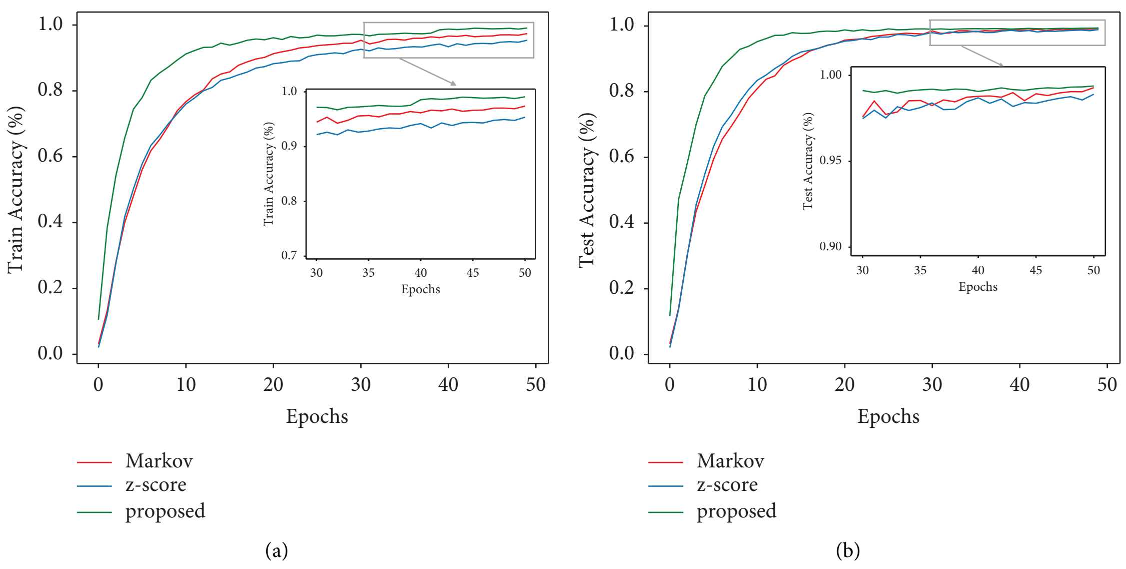


FIGURE 9: The changing trend of accuracy with training epochs on the GCJ dataset: (a) train accuracy and (b) test accuracy.

the performance of the method designed in this study has been improved to a certain extent. Further, it has excellent recognition ability, even if the sample distribution is not uniform.

4.3.3. Experiments on the GCJ Dataset. We view source code authors as family categories of malware and complete the classification problem by studying the source code authorship attribution issues, comparing the unknown source code with unique patterns in the source code of known authors, and identifying authors who program in different languages. Most experiments verify the model's generalization ability by using different malware datasets. However, this study also conducts experiments on the GCJ dataset to verify the performance of the model used in this experiment with sample sets from different domains. The experimental setup and evaluation metrics are consistent with Sections 4.1 and 4.2.

This experiment compares standardized images, Markov images, and the method proposed in this study to generate images on the GCJ dataset. Using three methods to classify the GCJ dataset, it is observed that the proposed method for generating malware images outperforms the other two methods, with significantly faster convergence. On the GCJ dataset, the schematic diagram of the variation trend of the

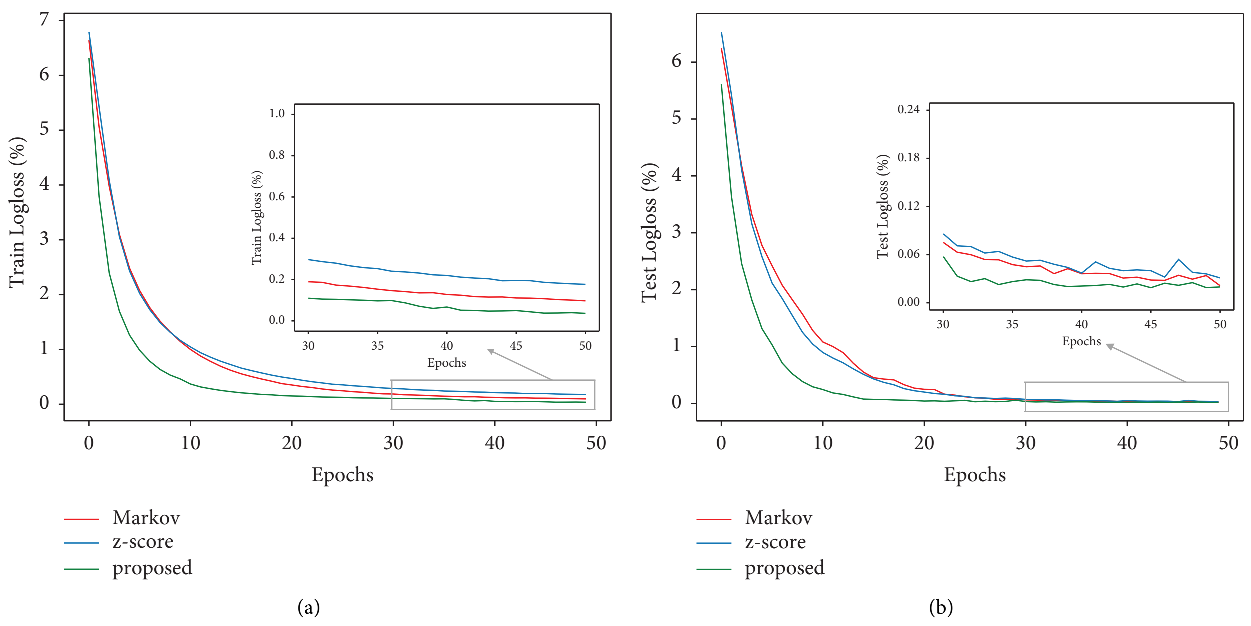


Figure 10: The changing trend of loss value with training epochs on the GCJ dataset: (a) train logloss and (b) test logloss.

Table 3: Evaluation of the effect of the improved model.

Image enhancement	AlexNet	Improved AlexNet	Accuracy
✓	✓	×	0.9695
×	×	✓	0.9915
✓	×	✓	0.9938

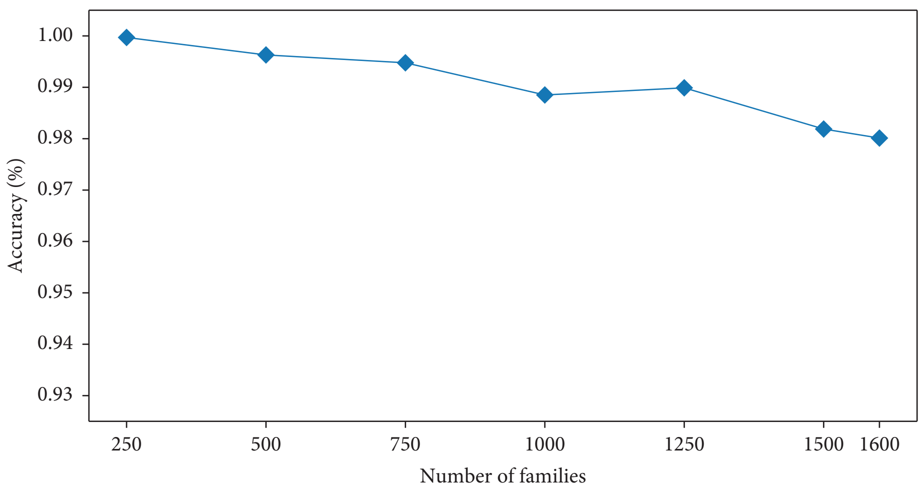


Figure 11: The trend of accuracy with an increasing number of families.

accuracy of the training set and the test set for the training period is shown in Figure 9. When the accuracy tends to be stable, it mainly fluctuates slightly up and down between 99.0% and 99.4%. On the GCJ dataset, the changing trend of the loss values of the training set and the test set for the training period is shown in Figure 10. With the continuous increase of the training period, the loss value of the loss function is constantly decreasing and gradually becomes stable. We can observe that the loss effect of the proposed method is significantly better than the other two methods.

This study designs the method for combining the improved CNN model with data enhancement. To verify the effect before and after improvement, the proposed method is compared with the CNN model without data enhancement method and before improvement. The selected experimental results are the highest accuracy on the training set, as shown

Table 4: Comparison with other relevant literatures when using the Microsoft dataset.

	Methods	Accuracy	*F*-measure
Gibert et al. [46]	Grayscale image + CNN	0.9750	0.9400
Le et al. [47]	Byte + CNN	0.9861	0.9714
Çayır et al. [48]	CapsNet + bagging	0.9956	0.982
Tekerek and Yapici [49]	Byte + B2IMG	0.9986	0.9237
Zhu et al. [50]	Multiple feature	0.9905	0.9852
This article	Proposed method	0.9999	0.9899

in Table 3. Through observation, we find that combining the improved network model with data augmentation, better results are achieved, and the accuracy is significantly higher than the other two methods.

To verify the experiment's scalability, data were collected from 1600 families in different years, nine samples were selected for each family for training, and 7 datasets of various sizes were created, including 250, 500, 750, 1000, 1250, 1500, and 1600 families. The variation trend of the accuracy with the increase in the number of families is shown in Figure 11. Through observation, the accuracy rate stays stable with the increase of sample categories. When the sample category reaches the maximum, the accuracy rate can still reach more than 98%, and it does not drop too much.

4.4. Comparison with Other Methods. We compare the method proposed in this experiment with state-of-the-art classification models in the other literature to evaluate it better. This section presents a comparative analysis of only the relevant literature on classification model experiments using the Microsoft dataset. The final results are shown in Table 4, showing the accuracy and F-measure corresponding to different methods, such as byte- and image-based malware classification (Gibert et al.[46], Le et al. [47], Çayır et al. [48], and Tekerek and Yapici [49]) and multi-feature-based malware classification (Zhu et al. [50]). Based on the research of Nataraj et al. [27], Gibert et al. [46] converted malware binary samples into 128×128 grayscale images. Deep learning was the first method to find patterns from the binary content image. Le et al. [47], based on a purely data-driven approach, used CNN to classify byte sequences of malware samples. Çayır et al. [48] used the bagging ensemble technology and capsule network-based model to classify malware images generated from bytes. Tekerek and Yapici [49] proposed a new method of conversion called B2IMG, which converts the extracted features into grayscale and RGB images. Zhu et al. [50] combined global structural features with local semantic features and extracted bytecodes and opcodes were converted into images for feature fusion. They completed malware sample classification through a dual-branch CNN. From Table 4, we can see that the accuracy of our proposed method is significantly better than the existing classification techniques, which proves that the algorithm has significant advantages over the most advanced methods in classifying malware families.

5. Conclusion and Future Work

In this study, we propose a malware classification method based on images and deep learning, which visualizes malware binary files as color images, directly generates the required image size, and uses data augmentation methods to improve the algorithm's performance. This method does not require reverse analysis and can directly extract sample features. The designed model has good generalization ability, saves time, and reduces space consumption. Further, it has excellent classification ability and effectively improves the accuracy of malware classification. Experiments show that the accuracy of using the CNN model designed in this study and the method of generating images can reach up to 99.99% for the Microsoft dataset and 99.38% for the GCJ dataset. Compared with other methods in the literature, our method is significantly better than other methods in the accuracy of malware samples.

Although our approach has yielded a good performance, there are still many challenges and shortcomings. For example, the number of model parameters used was large and only one feature of the sample was analyzed. In the follow-up work, we will extract more features (such as entropy images and APIs.) for sample classification and combine dynamic analysis based on static analysis. Meanwhile, we will conduct more experiments on real-world malware datasets.

Data Availability

In this article, the Microsoft malware dataset and the Google Code Jam dataset are used which can be found from the address "https://www.kaggle.com/c/malware-classification/" and "https://code.google.com/codejam/," respectively.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Major Program for Technological Innovation 2030-New Generation Artifical Intelligence (2020AAA0107700), the National Natural Science Foundation of China (62172244), the Shandong Provincial Natural Science Foundation (ZR2020YQ06 and ZR2021MF132), the Young Innovation Team of Colleges and Universities in Shandong Province (2021KJ001), the Taishan Scholars Program (tsqn202211210) and the Pilot Project for Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) (2022JBZ01-01).

References

[1] D. Zhao, L. Wang, Z. Wang, and G. Xiao, "Virus propagation and patch distribution in multiplex networks: modeling, analysis, and optimal allocation," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1755–1767, 2019.

[2] D. Zhao, G. Xiao, Z. Wang, L. Wang, and L. Xu, "Minimum dominating set of multiplex networks: definition, application, and identification," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 12, pp. 7823–7837, 2021.

[3] D. Gibert, C. Mateu, and J. Planes, "Hydra: a multimodal deep learning framework for malware classification," *Computers & Security*, vol. 95, Article ID 101873, 2020.

[4] S.-H. Seo, A. Gupta, A. Mohamed Sallam, E. Bertino, and K. Yim, "Detecting mobile malware threats to homeland security through static analysis," *Journal of Network and Computer Applications*, vol. 38, pp. 43–53, 2014.

[5] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using opcode sequences," *Information Sciences*, vol. 535, no. 1–15, pp. 1–15, 2020.

[6] A. Shalaginov, S. Banin, D. Ali, and K. Franke, "Machine learning aided static malware analysis: a survey and tutorial," *Cyber Threat Intelligence*, vol. 7, 45 pages, 2018.

[7] Z. Zhao, D. Zhao, S. Li, and S. Yang, "Malware classification based on visualization and feature fusion," in *Proceedings of the 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, pp. 53–60, IEEE, Shenzhen, China, September 2021.

[8] 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.

[9] Y. Sun, A. K. Bashir, U. Tariq, and F. Xiao, "Effective malware detection scheme based on classified behavior graph in IIoT," *Ad Hoc Networks*, vol. 120, Article ID 102558, 2021.

[10] N. L. Htun, "Malicious software family classification using machine learning multi-class classifiers," in *Computational Science and Technology*, pp. 423–433, Springer, Berlin, Germany, 2019.

[11] S. M. Bidoki, S. Jalili, and A. Tajoddin, "Pbmmd: a novel policy based multi-process malware detection," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 57–70, 2017.

[12] T. Kim, S. C. Suh, H. Kim, J. Kim, and J. Kim, "An encoding technique for cnn-based network anomaly detection," in *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, IEEE, Seattle, WA, USA, December 2018.

[13] D. Gibert, C. Mateu, J. Planes, and J. Marques-Silva, "Auditing static machine learning anti-malware tools against metamorphic attacks," *Computers & Security*, vol. 102, Article ID 102159, 2021.

[14] M. Woźniak, M. Grana, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, no. 3–17, pp. 3–17, 2014.

[15] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

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

[17] X. Liu, Y. Lin, H. Li, and J. Zhang, "A novel method for malware detection on ML-based visualization technique," *Computers & Security*, vol. 89, Article ID 101682, 2020.

[18] E. Amer and S. El-Sappagh, "Robust deep learning early alarm prediction model based on the behavioural smell for android malware," *Computers & Security*, vol. 116, Article ID 102670, 2022.

[19] Y. Zhao, C. Xu, B. Bo, and Y. Feng, "Maldeep: a deep learning classification framework against malware variants based on texture visualization," *Security and Communication Networks*, vol. 2019, Article ID 4895984, 11 pages, 2019.

[20] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, IEEE, Paris, France, February 2018.

[21] J. Wang and S. Wang, "Malicious classification based on deep learning and visualization," in *Proceedings of the 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 223–228, IEEE, Chengdu, China, September 2019.

[22] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, and Z. Qin, "Mthael: cross-architecture IoT malware detection based on neural network advanced ensemble learning," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1654–1667, 2020.

[23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[24] 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 ID 101748, 2020.

[25] 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.

[26] Z. Zhao, S. Yang, and D. Zhao, "A new framework for visual classification of multi-channel malware based on transfer learning," *Applied Sciences*, vol. 13, no. 4, p. 2484, 2023.

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

[28] A. A. Alipour and E. Ansari, "An advanced profile hidden Markov model for malware detection," *Intelligent Data Analysis*, vol. 24, no. 4, pp. 759–778, 2020.

[29] 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.

[30] I. Santos, K. Yoseba, J. Devesa, and P. Garcia Bringas, "N-grams-based file signatures for malware detection," *ICEIS*, vol. 9, no. 2, pp. 317–320, 2009.

[31] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 183–194, New Orleans, LO, USA, March 2016.

[32] H. Naeem, B. Guo, M. Rashid Naeem, and D. Vasan, "Visual malware classification using local and global malicious pattern," *Journal of Computers*, vol. 6, pp. 73–83, 2019.

[33] Y. Liu, Y. Lai, Z.-H. Wang, and H.-B. Yan, "A new learning approach to malware classification using discriminative feature extraction," *IEEE Access*, vol. 7, pp. 13015–13023, 2019.

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

[35] S. Li, L. Jiang, Q. Zhang, Z. Wang, Z. Tian, and M. Guizani, "A malicious mining code detection method based on multi-

features fusion," *IEEE Transactions on Network Science and Engineering*, vol. 1, 2022.

[36] V. P. Nair, H. Jain, Y. K. Golecha, M. Singh Gaur, and V. Laxmi, "Medusa: metamorphic malware dynamic analysis usingsignature from API," in *Proceedings of the 3rd International Conference on Security of Information and Networks*, pp. 263–269, Nice, France, September 2010.

[37] H. Lim, Y. Yamaguchi, H. Shimada, and H. Takakura, "Malware classification method based on sequence of traffic flow," in *Proceedings of the 2015 International Conference on Information Systems Security and Privacy (ICISSP)*, IEEE, Angers, France, February 2015.

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

[39] Z. Xu, X. Fang, and G. Yang, "Malbert: a novel pre-training method for malware detection," *Computers & Security*, vol. 111, Article ID 102458, 2021.

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

[41] D. Xue, J. Li, T. Lv, W. Wu, and J. Wang, "Malware classification using probability scoring and machine learning," *IEEE Access*, vol. 7, pp. 91641–91656, 2019.

[42] 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.

[43] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "Efficientnet convolutional neural networks-based android malware detection," *Computers & Security*, vol. 115, Article ID 102622, 2022.

[44] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning*, Corvalis, OR, USA, June 2015.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[46] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019.

[47] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, 2018.

[48] A. Çayır, U. Ünal, and H. Dağ, "Random capsnet forest model for imbalanced malware type classification task," *Computers & Security*, vol. 102, Article ID 102133, 2021.

[49] A. Tekerek and M. M. Yapici, "A novel malware classification and augmentation model based on convolutional neural network," *Computers & Security*, vol. 112, Article ID 102515, 2022.

[50] X. Zhu, J. Huang, B. Wang, and C. Qi, "Malware homology determination using visualized images and feature fusion," *PeerJ Computer Science*, vol. 7, p. e494, 2021.