IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 June 2024, accepted 24 July 2024, date of publication 29 July 2024, date of current version 7 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435362

RESEARCH ARTICLE

Efficient and Generalized Image-Based CNN Algorithm for Multi-Class Malware Detection

YAJUN LIU, HONG FAN, JIANGUANG ZHAO, JIANFANG ZHANG, AND XINXIN YIN
Information Engineering College, Hebei University of Architecture, Zhangjiakou, Hebei 075000, China

Corresponding author: Yajun Liu (lyj2100@hebiace.edu.cn)

This work was supported in part by the Research on Data Acquisition and Integration Based on Deep Learning under Grant 2221008A, in part by the Research on Data Acquisition and Integration of Tobacco Material Inspection under Grant ZY012022E001, in part by the Non-Invasive Monitoring Research of Office Building Electrical Equipment Based on Machine Learning under Grant 2022CXTD09, in part by the Science Research Project of Hebei Education Department under Grant QN2024148, and in part by the Deep Learning Behavioral Recognition Fall Detection Research under Grant 2022CXTD04.

ABSTRACT With the popularity of electronic devices, the number of malware has increased dramatically, posing a serious threat to the digital world. Accurately identifying malware has become a research focus. However, there are many difficulties in the research, such as insufficient algorithm generalization ability, unbalanced datasets, and long processing and identification times. To address these problems, this study proposes a malware detection framework (VBDN) based on a convolutional neural network (CNN). The framework incorporates data visualization, balanced adoption, data augmentation, and convolutional neural network techniques to achieve over 90% accuracy in classifying malware on all four open-source datasets. The experimental work has two other contributions: first, it not only focuses on the overall recognition effect of the algorithm during the research process, but also on the recognition effect of each category with the help of a confusion matrix, which provides useful information for cybersecurity personnel, researchers, and others to carry out subsequent targeted research. Secondly, the balanced approach adopted in this paper has the following advantages: no need to construct a new dataset, consumes fewer hardware resources, automatically evaluates the sampling weights, etc. Additionally, to enhance the generalization ability of the algorithm and alleviate the overfitting problem, this paper employs data augmentation techniques to improve the adopted method. By comparing with several state-of-the-art algorithms, it can be observed that the VBDN framework proposed in this paper achieves the desired results in time with acceptable accuracy.

INDEX TERMS Malware detection, convolutional neural network (CNN), data visualization, balanced adoption, data enhancement.

I. INTRODUCTION

With the popularity of computers, cell phones, tablets, and other electronic devices, people's lives are becoming more and more convenient. Electronic devices and computer networks have become an inseparable part of people's daily lives. With the popularity of these devices and technologies, security issues have gradually emerged, especially the widespread spread of malicious code, which has caused major threats to the digital world. These threats include the loss of user files, malicious extortion, information theft, telecommunication fraud, company shutdowns, and other issues, causing significant concern among scholars, enterprises, financial institutions, and governments.

The amount of malicious code has been increasing in the long term. In early 2020, AV-Test [1] predicted that the number of malware would reach 160 million samples that year, reaching a new scale. SonicWall [2] reported a doubling of cybersecurity risk from mid-2020 to 2021, with global cybersecurity spending reaching twice as much by the end of the year as it did in 2020. SonicWall Threat experts at Capture Labs say global malware attacks reached 2.8 billion in the

The associate editor coordinating the review of this manuscript and approving it for publication was Ramakrishnan Srinivasan.

first half of 2022, an 11 percent increase over the same period in 2021 [3]. According to Kaspersky [4], the number of global mobile banking Trojan detections has increased significantly, with more than 55,000 attacks in the second quarter of 2022 alone. According to Rising's ''2022 China Network Security Report'' [5], Rising's ''Cloud Security'' system intercepted a total of 73.55 million virus samples in 2022, with 45.15 million new Trojans and 1,520,500 cell phone virus samples, which makes the number of malicious codes frightening and has caused considerable harm to economic and social activities.

Malicious code attacks in 2022 were very numerous, some of the more representative ones are: on January 19, 2022, the International Committee of the Red Cross suffered a cyber attack in which the data of more than 515,000 people were written into. on January 19, 2022, Global Affairs Canada (GAC) faced a network outage and was unable to operate normally after an attack on its systems. on February 8, 2022, international telecommunications The Portuguese company of giant Vodafone said that it suffered a malicious attack that led to the complete disruption of its 4G/5G, fixed-line, and TV networks, which caused inconvenience and even disruption to millions of users in Portugal. On November 2, 2022, Jeppesen experienced an attack that forced the disruption of some of its flights, which was released after the attack.

In short, with the popularity of computer hardware devices and technology, the number of malicious codes has been growing rapidly for a long time in the past and will continue to grow in the foreseeable future. As the number of malicious codes increases dramatically, it poses a significant security threat to organizations and individuals. Malicious code attacks are frequent and threaten governments, enterprises, healthcare, finance, education, and other fields; ransomware, data leaks, hacking, and other attacks are continuous and far-reaching, seriously affecting the construction of critical information infrastructure and the economic and livelihood of countries. Network security research is working hard to defend against malware threats, while malware developers are evading these defense techniques. Traditional static and dynamic analysis methods have disadvantages such as low recognition efficiency, time-consuming, and large memory consumption. It is also difficult to circumvent these techniques by training traditional machine learning classification algorithms based on manual feature training, and feature engineering requires more human and material resources to mine potential features. The world is facing a great challenge for countries, enterprises, and finance due to the frequent incidents caused by malicious code spread on the Internet. How to accurately identify malicious codes with the times is not only of theoretical research significance but also of very important practical value.

The main contributions of this paper are highlighted below:

(1) This paper is based on the proposed VBDN framework which provides efficient detection and identification of malware.

(2) Focusing on the dataset imbalance problem, balanced sampling, and data augmentation techniques are used to ensure the generalization ability of the algorithm.

(3) More satisfactory experimental results are achieved in the multi-classification task, and the results do not only focus on the accuracy of the final algorithm for all data recognition but also focus on the recognition effect of each class of malware, which is analyzed and believed to be instructive for cybersecurity researchers and related scholars.

The rest of this paper is organized as follows: In Section II related work is presented on related malicious code detection and how it differs from our work; in Section III malicious code visualization, balanced adoption, GLCM, and CNN are introduced; in Section IV data enhancement, CNN architecture, and the VBDN framework are introduced; and in Section V the dataset, evaluation metrics, and so on are introduced, and based on this, we validate the VBDN framework effectiveness of the VBDN framework. Finally, we summarize our work and make an outlook for the future.

II. RELATED WORK

Malicious code refers to programming code or scripts designed to corrupt and interrupt normal operations in order to gain unauthorized access to a system. These codes are used for purposes such as stealing data, deleting files, and more. Common operating systems such as Windows, Android, and iOS, as well as applications such as booking, hotel management, and student management, are subject to malicious code attacks.

After the first public virus Brain appeared in 1986, Until now, malicious code identification and development have been in a continuous race to catch up with each other. To date, common malicious codes include viruses, worms, Trojan horses, ransomware, and spyware. The traditional methods for identifying this malware include signature identification; behavioral analysis; heuristic analysis; and sandbox analysis.

Signature recognition: This method analyzes and extracts features from known malware samples, saves these features into a database, and when the features of a new file are highly consistent with the features of a certain type of malware in the database, the file is judged to be malware, and the set of samples in the database is updated. For example, Yang et al. [6] used signature recognition to analyze malicious apps in Android. weight Zhang et al. [7] proposed the DAMBA model to compare with signature-based recognition Mcafee [8] in terms of time and accuracy and achieved better results. Behavior analysis: This method analyzes the behavior of software by monitoring its operations on files, networks, etc. to determine whether it is malware. It's commonly used to detect unknown malware and the misclassification is more serious. For example, Rosli et al. [9] use unsupervised learning K-means algorithm for clustering analysis of malware behavior. Ding et al. [10] perform behavior analysis on a control flow-based approach and use algorithms such as KNN and SVM for classification. Heuristic analysis: This method will analyze the dynamic

and static features of the software, and determine whether the software is malware by evaluating the behavior and core features of the software. It can be used to detect some unknown malware but has the disadvantage of taking a long time and requiring a lot of computational resources. For example, the SigMal [11] framework is a heuristic that uses PE structure information to optimize feature-based signal processing and thus similarity detection of malware. Sandbox analysis: This method is used to identify malware by running software in a secure environment, such as virtual machines, containers, etc. Similar to behavioral analysis, malware is identified by observing its behavior and activities, but it also has the disadvantage of requiring a large amount of computational resources and time. Moreover, as time evolves, the emergence of methods such as anti-virtualization and hook evasion may allow malware to bypass such detection methods, and related simulations are complementing their drawbacks [12], [13].

Earlier malware was mostly written by simple code, so traditional methods of identification were sufficient to cope with it. Nowadays, most malware is designed to execute based on the kernel, which makes identification much more difficult, and traditional methods cannot meet the requirements of malware detection [14]. The emergence of technologies such as machine learning and deep learning [15] provides a feasible solution to the problem, and their methods are more efficient.

Earlier classification of malware generally used n-gram and other extracted text feature extraction, which in turn combined with machine learning algorithms to complete the classification task [16], [17], or used CNN to classify features extracted from malware [18]. In contrast, today malware shows family features when found, and some of them execute in the kernel, which makes it difficult to detect malware and also more challenging to classify it. The current use of deep learning to identify and classify viruses from various operating systems. Kim et al. [19] used strings, APIs, and permissions for feature extraction of Android malware, trained multi-modal deep learning, and evaluated 41,260.DroidDetector [20] correlated features from static and dynamic analysis using deep learning techniques to analyze malware for Android. Huang et al. [21] use software visualization combined with convolutional neural networks in visualization, use sandboxes to analyze samples dynamically and use designed algorithms to convert them into visual images, and then train neural networks. Cui et al. [22] convert executable files of malicious code into grayscale images, and use Convolutional neural networks and intelligent algorithms were used to identify the malicious code, and the problem of data set imbalance was noticed during the identification process. Hemalatha et al. [23] used data visualization and the DenseNet algorithm to complete the classification task and used a reweighted class balance loss function in the classification layer, which in turn improved the performance of the algorithm.

Research in the last two years has demonstrated the wide range of applications of deep learning in the field of malware detection, from Internet of Things (IoT) security [24], [25] to malware detection under adversarial attacks [26], [27], [28], as well as the use of multimodal deep learning in Android malware identification [29], [30] to malware identification on Windows platforms. Specifically, the DCEL model proposed by Xu et al. [29] improves the detection accuracy of Android malware through classifier fusion. the system developed by Sathyaraj et al. [30] effectively detects industrial environment with QR code based attacks.Poulomi et al. utilised the ExtraTreeClassifier() function module to select relevant features for binary classification, which in turn accomplished malware identification [31]. Maniriho et al. introduced API- MalDetect, an automated Windows malware detection framework based on API calls and deep learning techniques [32]. In addition, a review conducted by Gopinath and Sethuraman evaluated the application of deep learning in various malware detection scenarios, including the Windows platform [15]. In addition, Twardawa et al. [33] and Sadhwani et al. [34] developed efficient threat detection and monitoring systems.The dynamic analysis data preprocessing technique explored by Kim and Kim ([35] provides a new approach for deep learning models to process time series data.

These advances not only improve the detection accuracy but also enhance the ability of the models in dealing with emerging threats. However, it can be observed that most of these research methods, which are specific to a certain system and platform, are very restrictive and cannot be generalized. In view of this, this research adopts malicious code visualisation to achieve maximum uniformity for subsequent extension.

III. ALGORITHM AND DESIGN

This section introduces malware visualization, balanced sampling, GLCM, and neural networks.

A. MALWARE VISUALIZATION

Malicious code is generally an executable binary file on a computer, divided into 8 bits, each assumed to be b_7 to b_0 from high to low, and converted to decimal using Equation 1, corresponding to a value in the grayscale image [0, 255].

$$D = \sum_{i=0}^{7} (b_i \cdot 2^i) \tag{1}$$

In Equation 1, D represents the final decimal value obtained from the binary digits. b_i represents the binary digit (either 0 or 1) at position i. i is the index of the binary digit, ranging from 0 to 7 in this case. 2^i represents 2 raised to the power of i, which is the positional value of the binary digit b_i.

The process of converting malicious code to image features is illustrated in Figure 1.

Similar to extracting grayscale map features from malicious code, the malicious code is feature extracted from

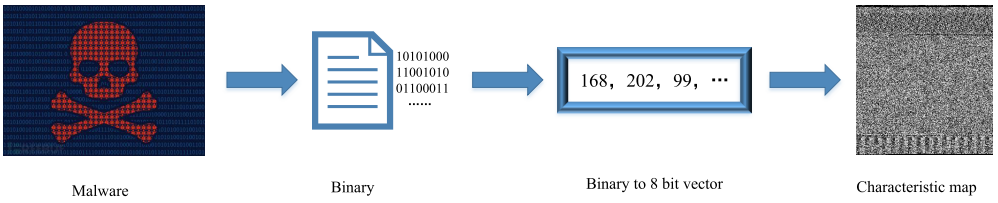


FIGURE 1. Malware converted to grayscale feature map.

different dimensions and mapped to RGB three channels to obtain the characteristic map, as shown in Figure 2. Commonly used methods for feature extraction of malware in different dimensions include PE view, assembly view, byte view, character information, byte stream information, PE structure letter, etc.

B. BALANCED SAMPLING

In the following sampling diagram, the left side represents the original dataset, and the right side represents the balanced dataset obtained after processing. As shown in Figure 3, balancing the number of sample categories can be achieved through both up-sampling and down-sampling methods. However, its drawbacks are also quite evident. Upsampling is achieved through replication to achieve a balance in the number of samples, which makes it very easy for the algorithm to overfit in subsequent processing; Downsampling refers to deleting samples with a large number of samples, which may cause information loss. Moreover, these two methods require the additional creation of new balanced datasets, which consumes hardware resources.

C. GLCM AND NEURAL NETWORKS

The commonly used methods for feature extraction from malicious code feature maps include Gray Level Co-occurrence Matrix (GLCM) and deep learning.

1) GLCM

The principle of the core of the GLCM is illustrated in Equation 2.

$$P(i,j|d,\theta) = \{(x,y)|f(x,y) = i, f(x+dx, y+dy) = j; x, y = 0, 1, \ldots, N-1\} \quad (2)$$

In Equation 2, $P(i,j|d,\theta)$ represents the joint probability of pixel pairs having gray levels i and j at a distance d and direction θ, where $f(x,y)$ is the gray level of the image at coordinates (x,y). i is the gray level of a pixel in the image, j is the gray level of a pixel that is at a distance d and direction θ from a pixel with gray level i. (x,y) are the coordinates of a pixel in the image. dx and dy are the offsets in the x and y directions respectively, depending on the distance d and direction θ. N is the size of the image, assuming the image is an $N \times N$ square image.

Using Equation 2, we obtain the gray-level co-occurrence matrix of malware feature images. Based on this, this paper utilizes six texture features, namely contrast, dissimilarity, homogeneity, angular second moment, correlation, and energy, to extract gray-level co-occurrence matrix features and combines them with traditional machine learning algorithms to complete multiple classification tasks.

2) NEURAL NETWORK

A neural network is a computational algorithm that mimics biology and is now widely used in machine learning and deep learning tasks. Neural networks consist of multiple layers, each layer contains many neurons that are connected by weights to achieve information transfer and processing between inputs and outputs. The composition formula is shown in Equation 3.

$$a = g\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad (3)$$

where a denotes the output, $g(\cdot)$ denotes the activation function (e.g., ReLU, Sigmoid, or Tanh), w_i denotes the weight of the i-th input signal, x_i denotes the i-th input signal, and b denotes the bias.

In the training process, the input data is first passed to the neural network and the output of the network is calculated. Then, the loss function (e.g., mean square error or cross-entropy loss) is calculated by comparing the network output with the actual target value. Next, the gradient of the loss function with respect to the weights is calculated using a backpropagation algorithm and the weights are updated using a gradient descent method to minimize the loss function.

Neural networks are now used in several fields such as image classification, speech recognition, natural language processing, and reinforcement learning. With the development of neural network structures, such as CNN, Recurrent Neural Networks (RNN), and Transformers, neural networks have achieved significant performance improvements in various fields, and this paper will also use CNN for feature extraction of feature maps of malicious codes, the details of which are described in detail in subsection IV-C.

IV. PROPOSED ALGORITHM

As described in Section V-A, four datasets with grayscale and RGB color image features and varying image sizes are used in this study. To facilitate uniform learning of the algorithm, the neural network structure in Section IV-C is designed for three channels, allowing the algorithm to

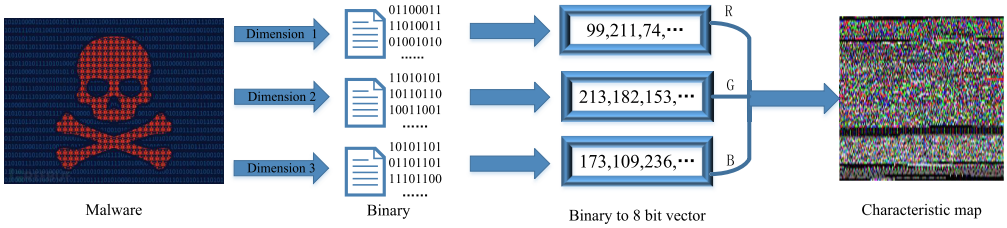
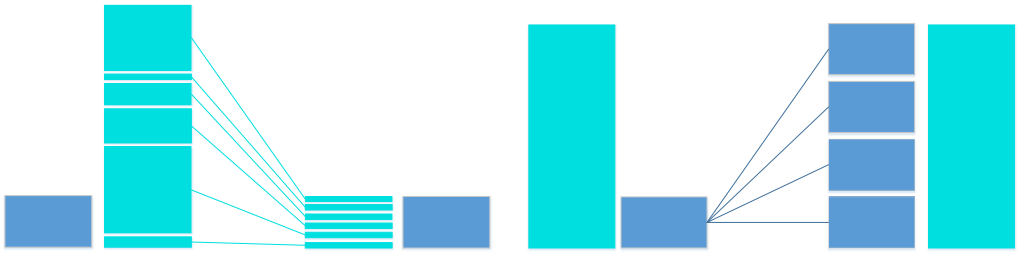


FIGURE 2. Malware converted to RGB feature map.



(a) Down sampling (b) Up sampling

FIGURE 3. Up and down sampling.

perform feature extraction for grayscale images. To address the issue of algorithm overfitting due to repeated sampling of small samples in Section IV-B, the data enhancement technique in Section IV-A is introduced.

A. DATA ENHANCEMENT

The data enhancement technique used in this article is implemented using the compose method in Python. The core data enhancement process is shown in Figure 4.

B. BALANCEDDATASETSAMPLER ALGORITHM OPTIMIZATION

When dealing with category-imbalanced datasets, four common methods are SMOTE (Synthetic Minority Over-sampling Technique), ADASYN (Adaptive Synthetic Sampling Approach), SMOTE-IPF (Synthetic Minority Over-sampling Technique with the Inverse Probability of Failure), and BalancedDatasetSampler. Unlike SMOTE, ADASYN, and SMOTE-IPF, which generate new samples, the BalancedDatasetSampler algorithm balances the dataset by weighted sampling. It samples according to the sample weights of each category, allowing the model to focus more on the minority class samples, thus enhancing the ability to learn from them. This method works directly during the data loading phase and adjusts the sampling of the samples. It also works directly on the dataset indexing, saving hardware space. However, since the algorithm does not generate new samples, data enhancement techniques are introduced to mitigate overfitting and improve generalization capability.

In this article, the BalancedDatasetSampler algorithm based on PyTorch is used, and the algorithm diagram is shown in Figure 5. The implementation of the code can be accessed through the link at the end of the article.

As shown in Figure 5, the sampling implemented by the BalancedDatasetSampler has the same features as upsampling. However, when using the Python implementation, the index implementation of the returned data avoids creating a new balanced dataset. Additionally, due to the presence of oversampling, the data enhancement technique in Section IV-A is introduced during feature extraction to reduce algorithm overfitting and enhance generalization capability.

C. STRUCTURE OF CNN

Neural networks have achieved excellent results in recent years, especially in feature extraction and recognition of graphical images. In this paper, CNN is used for feature extraction of feature maps of malicious codes. Its core structure is shown in Figure 6.

The network structure is a convolutional neural network composed of multiple layers, including convolutional layers, pooling layers, linear layers, and activation functions. The input data is 3-channel image data. After passing through the first convolution layer, the number of channels becomes 32, the convolution core size used is 3×3, and the step size is 1. Next, it passes through the pooling layer, with a size of 2×2 and a step size of 2, to perform dimensionality reduction. The number of convolutional channels in the second layer becomes 64, and a 3×3 convolutional core is used again, with a step size of 1. Continue pooling, with a size of 2×2 and a step size of 2. The number of channels in the third layer of the convolutional layer becomes 128, and the convolutional core of 3×3 is also used with a step size of 1. Pool again, 2×2 in size, in step 2. After that, perform global average pooling, and

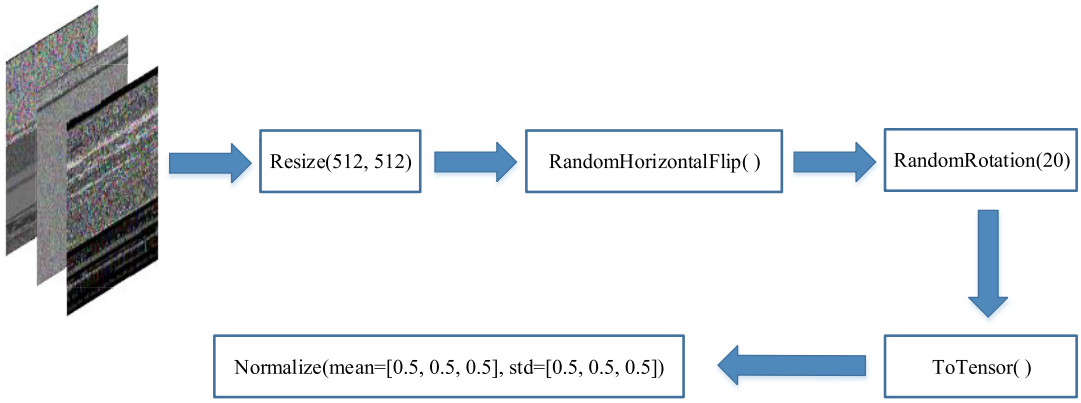


FIGURE 4. Malicious code feature map data enhancement steps.

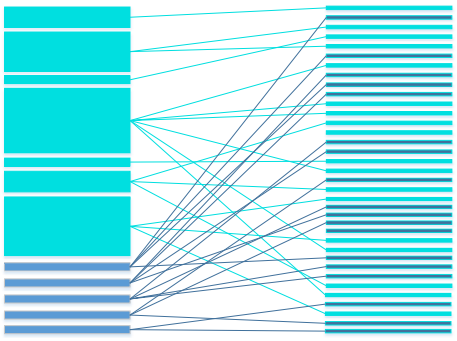


FIGURE 5. BalancedDatasetSampler algorithm.

the output size is 28 × 28. Finally, two linear layers are used for classification, with the output size of the first linear layer being 64 and the output size of the second linear layer being a number, which is the number of categories classified. The activation function uses the ReLU function.

D. VBDN

Based on code visualization, balanced sampling, data enhancement, and neural network techniques, the core framework for malicious code (VBDN) detection proposed in this study is shown in Figure 7.

The framework enables the visualization of malicious codes, as well as the determination of whether the dataset is balanced, and imbalance calls ImbalancedDatasetSampler to complete the balancing process of the dataset. To improve the generalization ability of the algorithm, data enhancement techniques are introduced, while the heat map of the confusion matrix is used to analyze the identification of each category in the classification task. Understanding the algorithm learning to better help will technicians focus on those malicious codes that are difficult to identify.

In terms of hyperparameter tuning, we use cross-validation and grid search to find the best hyperparameter combinations. We tuned each hyperparameter within a certain range, and chose the hyperparameter combination that achieves the best performance on the validation set.Batch Size: We set the batch size to 64 for training and 32 for testing.Epochs: We chose to train 200 epochs in order to adequately train the algorithm and observe its convergence.Learning Rate: We set the learning rate to 0.01, which is a common initial learning rate. We found through experiments that this learning rate can balance the convergence speed and performance of the algorithm to a certain extent.Momentum: We choose the momentum parameter of SGD to be 0.5, which can help the algorithm to converge to a better local optimal solution faster during the training process. Random Seed: We set the random seed to 50 to ensure the repeatability of the experiment and the stability of the results.

TABLE 1. Dataset statistics.

Dataset	Training size	Test size	Labels	Balanced
Malimg	8408	935	25	N
Big2015	10868	0	9	N
Malevis	9100	5126	26	Y
Blended	9867	3879	31	N

V. EXPERIMENTAL EVALUATION

A. DATASETS

In this article, four datasets (Malimg dataset, Big2015 dataset, Malevis dataset, and Blended dataset) are used to evaluate the algorithm.

The histograms of the test sets of the datasets are shown in Figure 8.

It can be found that the Malevis dataset is a balanced dataset, but during the experiment, we found that the other class is not balanced on the test set, and we conducted the experiment again after doing the corresponding processing. The other three data sets are non-balanced data sets. In this article, the algorithm in Section IV-B is used for balancing, and after data enhancement, algorithm training and evaluation are conducted.

Statistics were collected from the training and test sets of the four datasets, as shown in Table 1.

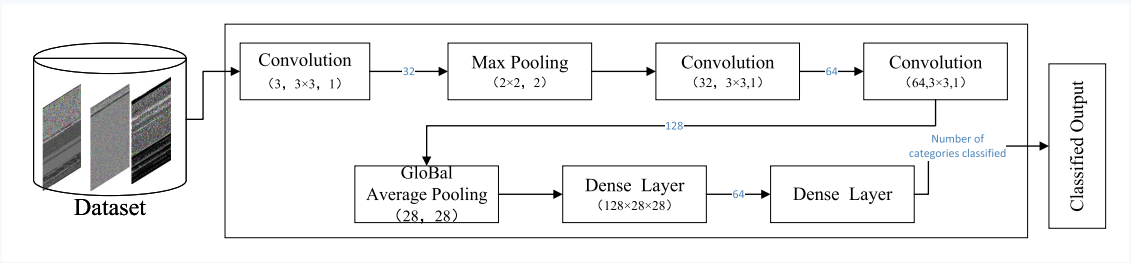


FIGURE 6. Malicious code feature extraction and classification.

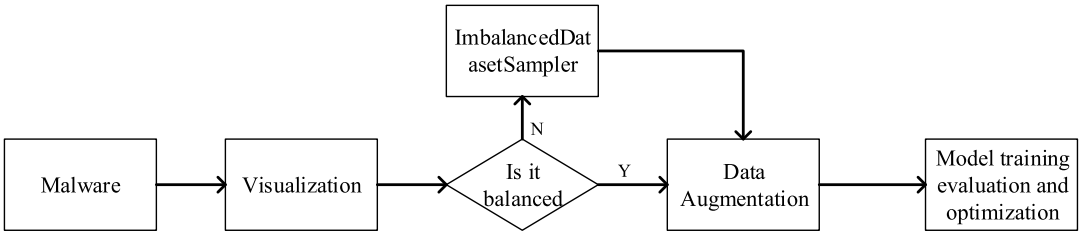


FIGURE 7. Malicious code detection framework(VBDN).

Since there is no test set for the Big2015 dataset, we divided 30% of the original dataset to use as a test set.

Further analysis of the datasets revealed that Malevis was a balanced dataset on the training set, and the other category appeared as 1482 samples on the test set, while the other category was roughly in the range of 120-150 samples, so the other category was correlated.

The Blended dataset is a combination of the Malimg and Malevis datasets. The dataset uses all the data from Malevis and 5 categories from Malimg to form a dataset with 31 categories, which has more categories and both grayscale and RGB color images in the dataset, making classification more challenging.

B. EVALUATION CRITERION

In machine learning and deep learning classification tasks, common metrics used to evaluate algorithms include Accuracy (Acc), Precision (Pr), Recall (Re), and F1-score (F1). Accuracy, Recall, and F1-score are commonly used in the case of unbalanced categories, but in this paper, the data set is balanced after processing, so using accuracy can really reflect the effect of the algorithm. The formula for calculating the accuracy is shown in Equation 4.

$$\text{Acc} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

where TP denotes the number of correctly classified positive cases; FP denotes the number of incorrectly classified positive cases; TN denotes the number of correctly classified negative cases; and FN denotes the number of incorrectly classified negative cases.

Additionally, to better demonstrate the improvement effect of the models, we calculated the time improvement percentage for each model on different datasets, as shown in Equation 5.

$$\text{Improvement} = \left(\frac{T_{\text{model}} - T_{\text{ConvNet}}}{T_{\text{model}}}\right) \times 100\% \tag{5}$$

where T_{model} represents the testing time (in seconds) of a certain model on a specific dataset, T_{ConvNet} represents the testing time (in seconds) of the ConvNet model on the same dataset, and Improvement represents the percentage improvement in time.

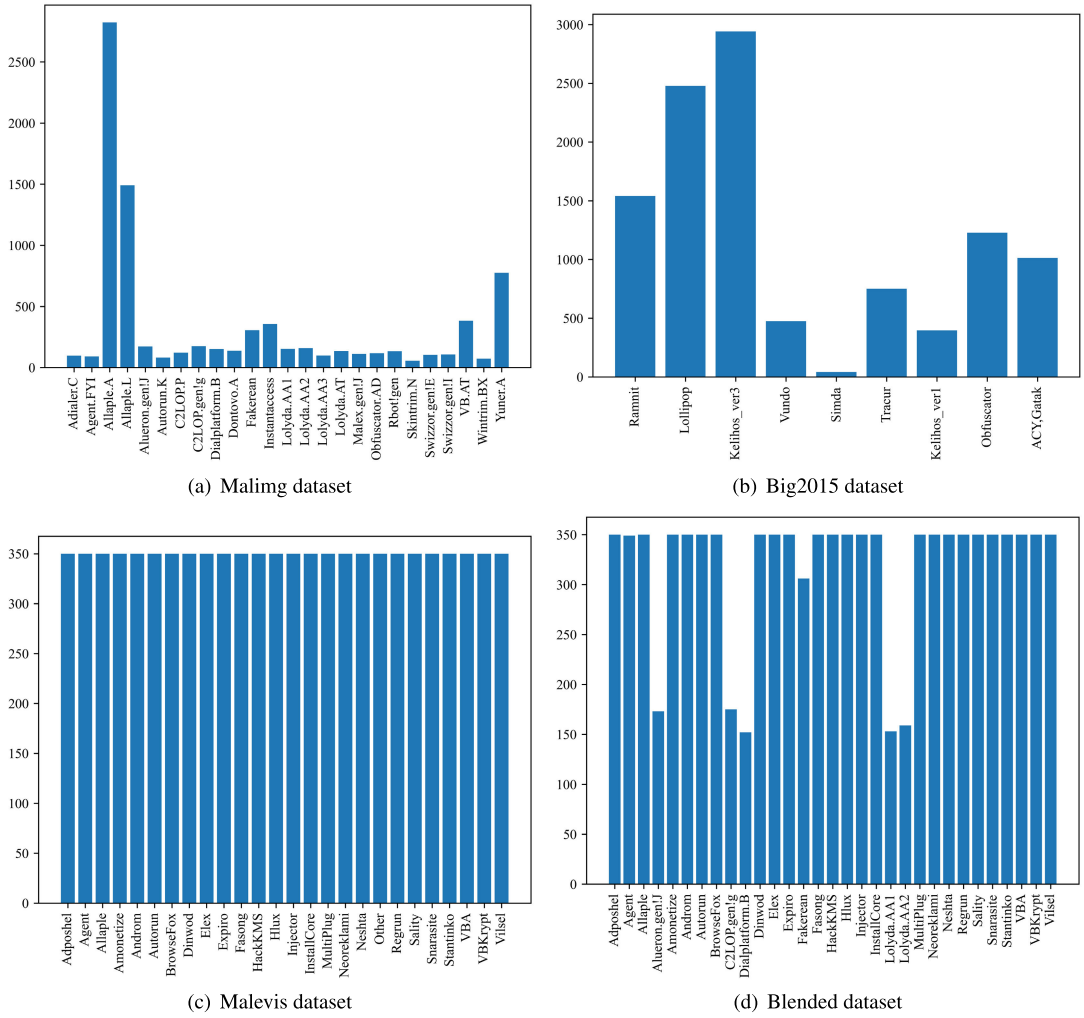
C. EXPERIMENT AND ANALYSIS

Hardware used in this study includes: Intel(R) Core(TM) i5-10600KF, 32GB RAM, 3060 graphics card, etc. Software includes:Python 3.10, PyTorch, cuDNN, Jupyter Notebook, etc.

In Section III-C.1 GLCM is used for feature extraction of malicious code feature maps and combined with machine learning algorithms to accomplish the classification task. The CNN network structure in subsection IV-C and the malicious code detection framework(VBDN) in subsection IV-D are used in subsection V-C.2 to complete the feature extraction and classification tasks.

1) ANALYSIS OF GLCM EXPERIMENTAL RESULTS

Based on Section III-C.1, we extracted the feature map dataset of malicious code by using BalancedDatasetSampler in Section IV-B after balancing the dataset extracted six texture characteristics, namely contrast, dissimilarity, homogeneity, angular second moment, correlation, and energy. And Logistic Regression(LR), Naive Bayes(NB), K-Nearest Neighbor(KNN), Decision Tree(DT), Random Forest(RF), Gradient Boosting Decision Tree(GBDT), Extreme Gradient



(a) Malimg dataset (b) Big2015 dataset

(c) Malevis dataset (d) Blended dataset

FIGURE 8. Training set sample data statistics.

Boosting Decision Tree (XGBoost), Light Gradient Boosting Machine(LightGBM), Support Vector Machine(SVM), and Multilayer Perceptron Classifier(MLPC) were used to complete the classification task, and the experimental structure was obtained as shown in Table 2.

The importance of the six features in different models was further analysed and feature_importance was used to view the importance of different features in the tree models and it was found that the main features affecting the tree models (RF, DT, GBDT, XGBoost) were: 'contrast' and 'correlation'. The features of the non-tree models were visualised using the SHAP library and it was found that the main features affecting the features of the non-tree models were 'contrast' and 'dissimilarity'. It can be found that contrast plays a very big role in the model, which means that the difference in grey level of malicious code is more pronounced.

After comparing the data in Table 2, it is found that DT, RF, GBDT, XGBoost, and LightGBM achieve over 90% accuracy on the Malimg and Big2015 datasets, and perform relatively poorly on the Malevis and Blende datasets. The traditional machine learning algorithms of LR, NB, KNN, SVM, and MLPC perform poorly overall, and the individual algorithms perform similarly to the accuracy of random guesses.

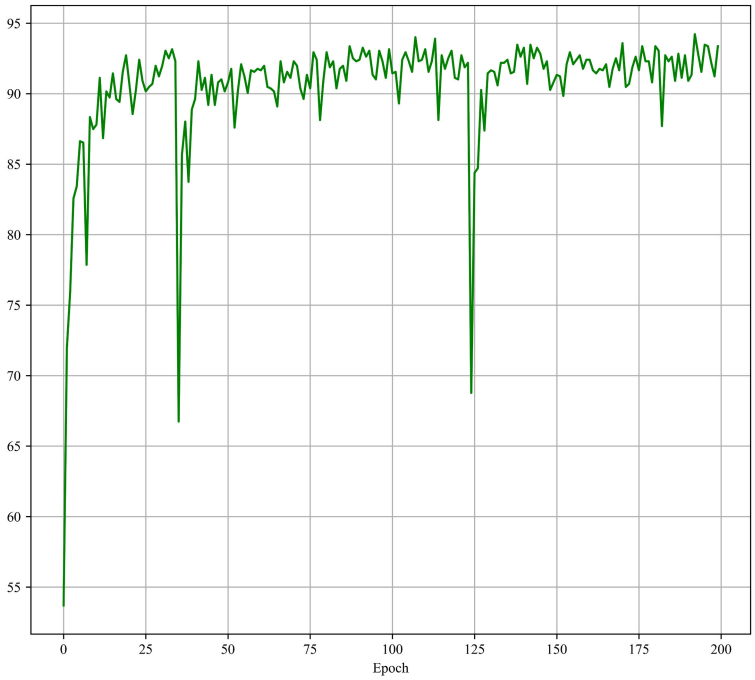
TABLE 2. Classification results based on GLCM features.

Algorithm	Malimg(%)	Big2015(%)	Malevis(%)	Blende(%)
LR	29.01	67.74	47.23	49.71
NB	68.82	69.91	52.21	48.67
KNN	59.42	65.82	64.52	64.16
DT	90.71	92.51	87.23	87.44
RF	91.15	95.10	89.12	89.70
GBDT	91.36	91.92	86.06	87.01
XGBoost	92.37	94.3	88.33	88.60

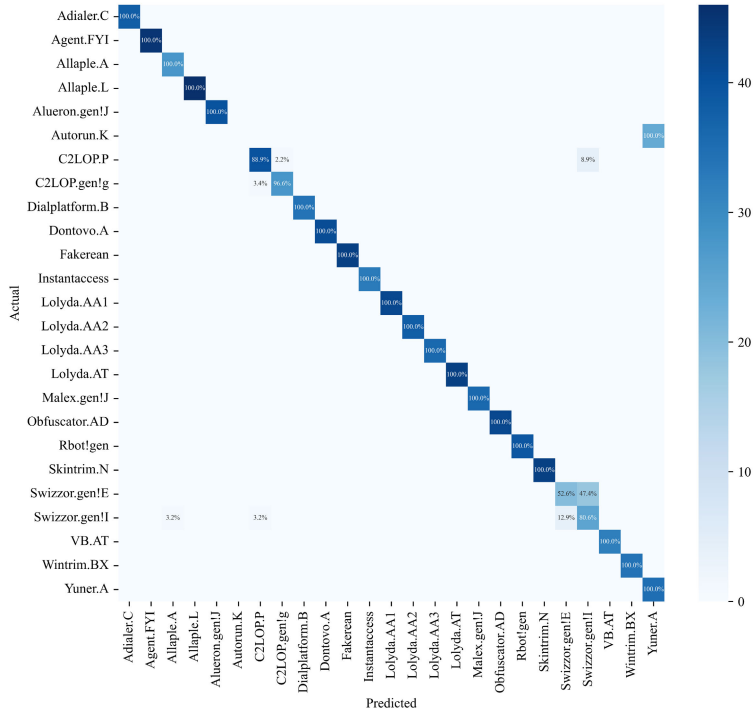
2) ANALYSIS OF CNN EXPERIMENTAL RESULTS

The CNN network algorithm in subsection IV-C and the VBDN framework in subsection IV-D are used to complete the experiments on four datasets.

In order to visualize more how well the algorithm predicts each category, here we use accuracy as a metric and also use a confusion matrix to see how well each category is



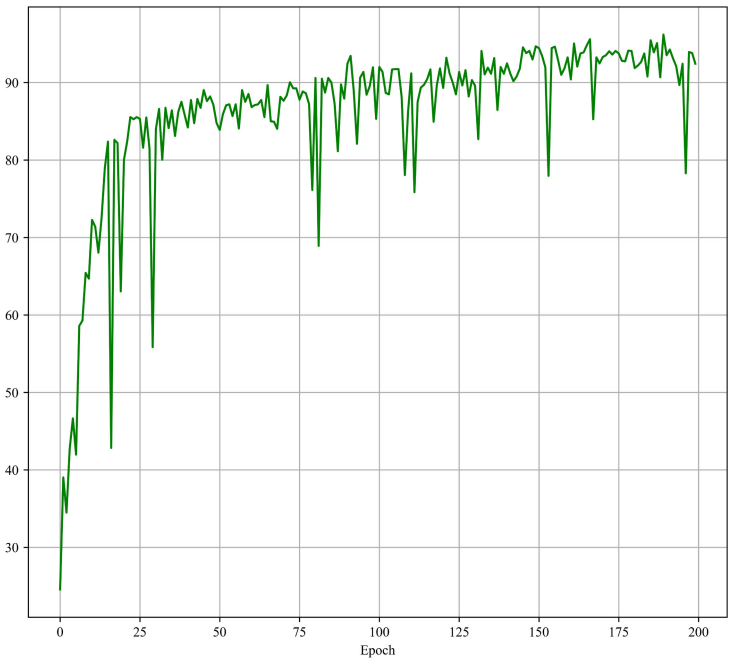
(a) Accuracy for the Malimg dataset



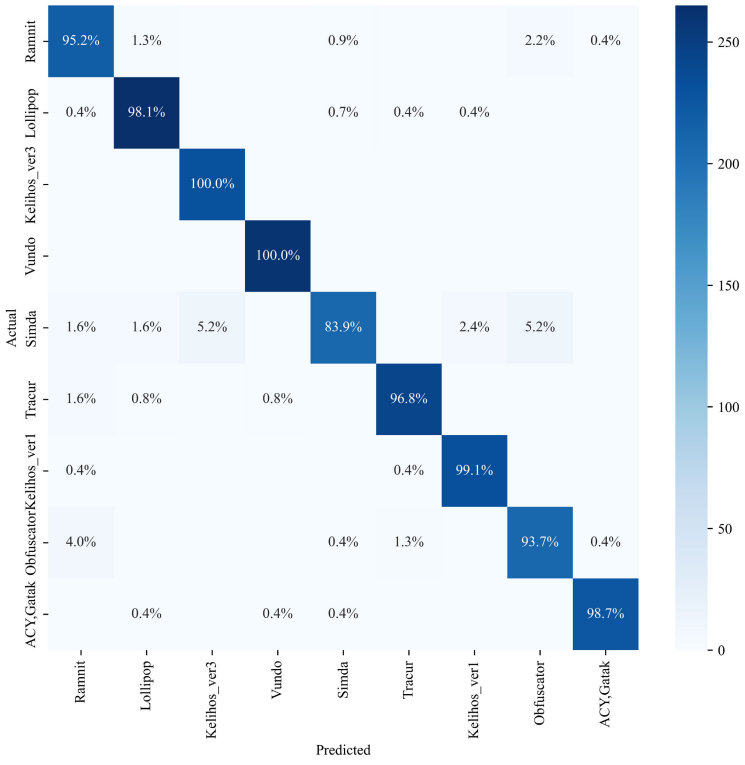
(b) Confusion matrix for the Malimg dataset

FIGURE 9. Experimental results for the Malimg dataset.

predicted.In this confusion matrix, the rows represent the true categories, and the columns represent the predicted categories. The diagonal line represents the percentage of correct categorization by the algorithm. Through the confusion matrix and its visualization, it is possible to intuitively find out which categories the algorithm is prone to confusion, which facilitates a deeper understanding of the algorithm's performance and shortcomings, and at the same time provides some guidance for cybersecurity researchers.



(a) Accuracy for the Big2015 dataset

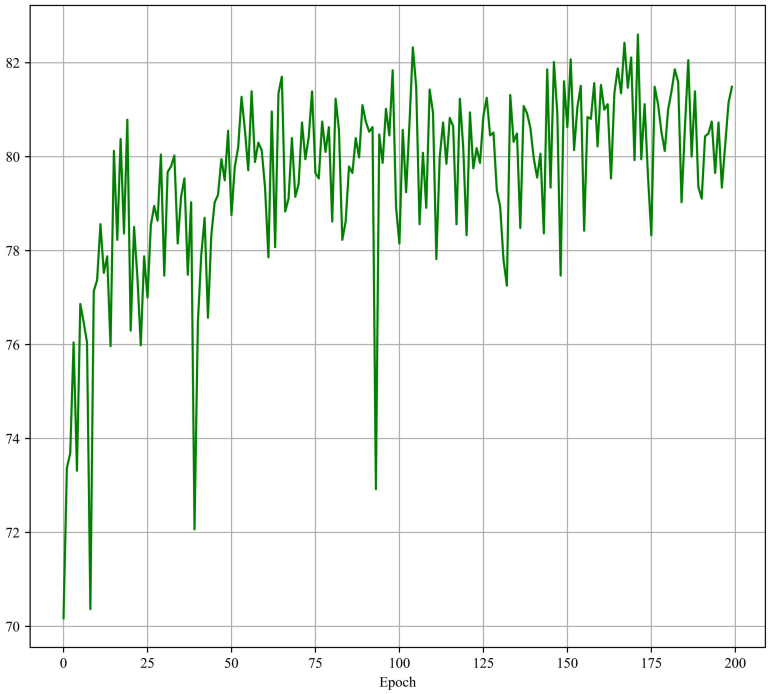


(b) Confusion matrix for the Big2015 dataset

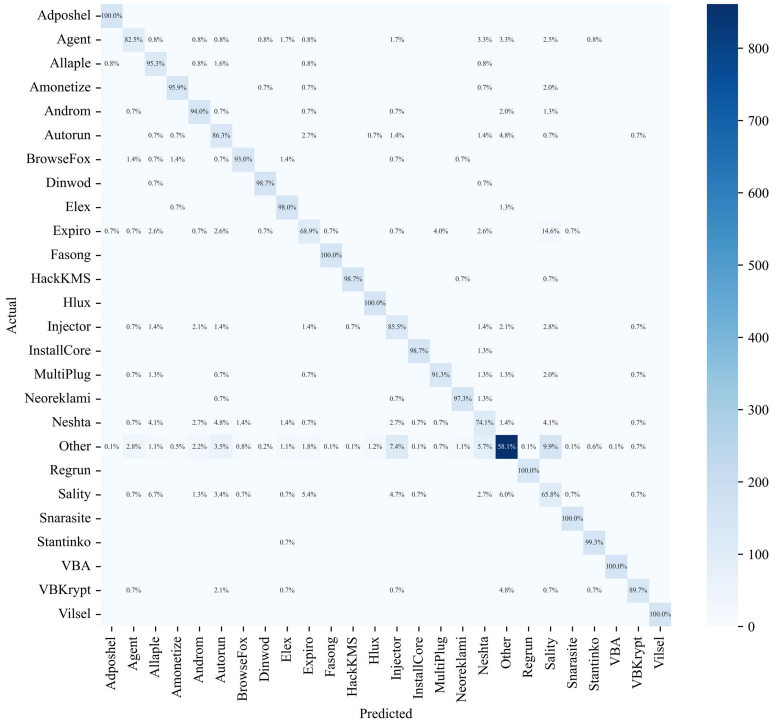
FIGURE 10. Experimental results for the Big2015 dataset.

Figure 9 shows the results of the algorithm training evaluation for the Malimg dataset. The accuracy reached 94.22%, which is an improvement of 1.85% compared to the best-performing XGBoost algorithm in Table 2. A high similarity between Autorun.K and Yuner.A categories was also found, and the Swizzor.gen!E category was not identified very well.

Figure 10 shows the results of the algorithm training evaluation for the Big2015 dataset. Its accuracy reaches 96.19%, and the algorithm's overall accuracy is 1.09% higher



(a) Accuracy for the Malevis dataset

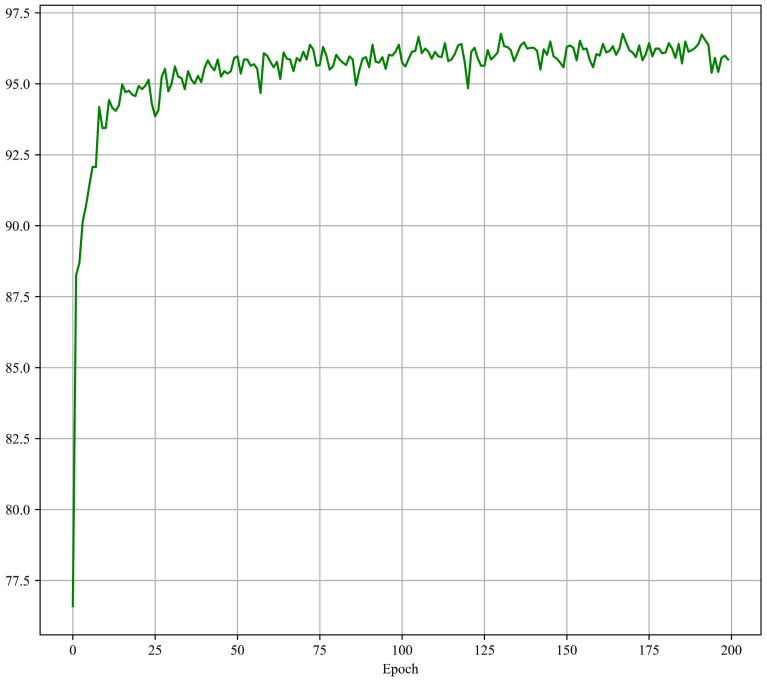


(b) Confusion matrix for the Malevis dataset

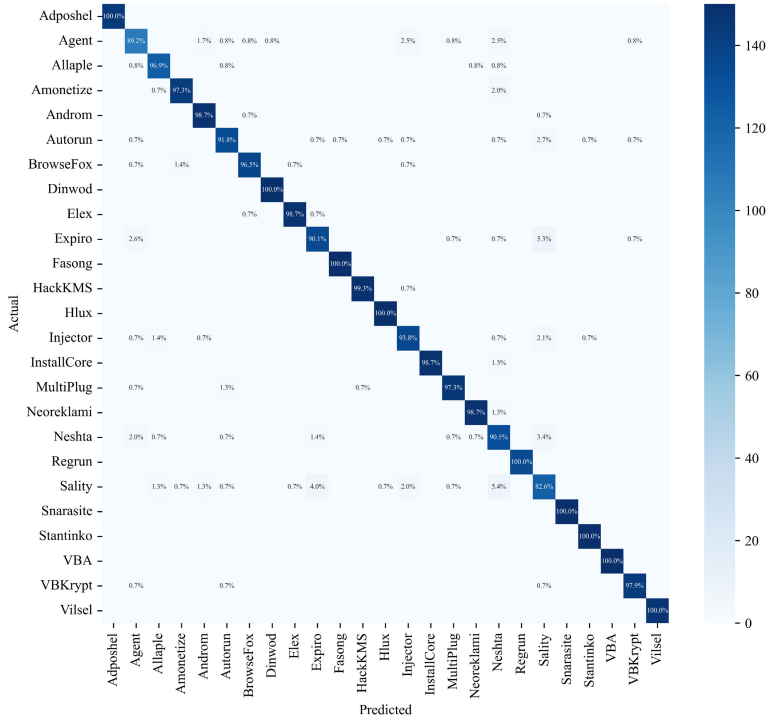
FIGURE 11. Experimental results for the malevis dataset(with other categories).

than that of the random forest with the best recognition effect in Table 2. At the same time, it can be found that the algorithm is relatively less effective in recognizing viruses of Simda class, with an accuracy of only 83.9%.

Figure 11 contains the Other category, whose accuracy reaches the highest 83.22%, and the heat map shows that the accuracy of the Other category is only 58.1% on this Malevis dataset. The analysis of the dataset reveals



(a) Accuracy for the Malevis dataset

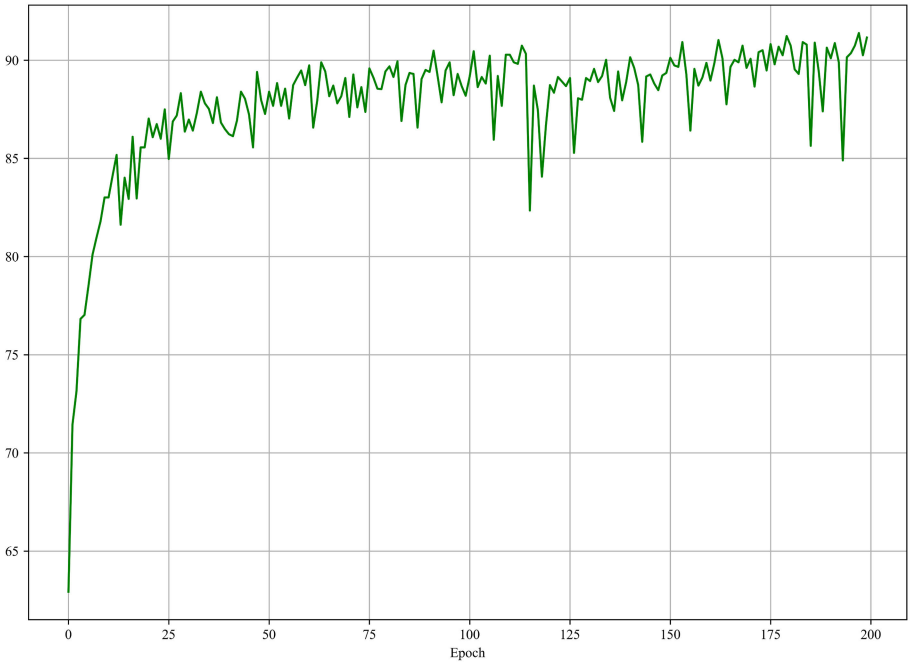


(b) Confusion matrix for the Malevis dataset

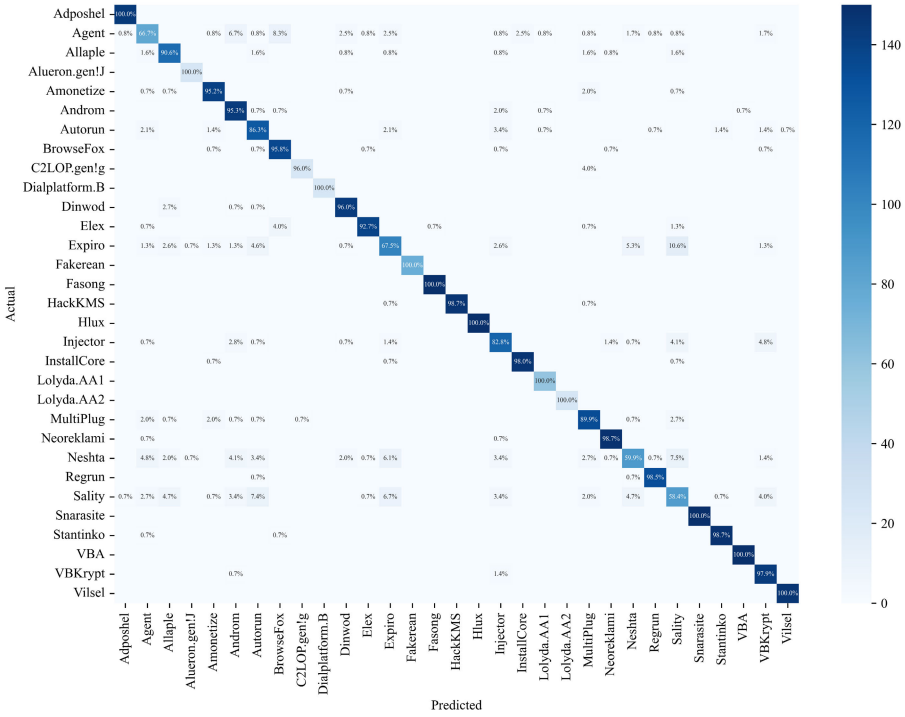
FIGURE 12. Experimental results for the malevis dataset(without other categories).

two possible reasons for the poor results: 1. The training sample of the 'Other' category in the training set has only 350 data points, while there are 1482 in the test set, so the algorithm cannot learn the features of this category well; 2. The software features in the 'Other' category are more complex and have similarities with other malicious codes.

After removing the 'Other' category, the algorithm was retrained to obtain Figure 12, which shows a significant improvement in the Malevis dataset, with an accuracy of



(a) Accuracy for the Blend dataset



(b) Confusion matrix for the Blend dataset

FIGURE 13. Experimental results for the blend dataset.

96.76%. This is 7.64% higher than the best-performing random forest in Table 2.

Figure 13 shows the results of the algorithm training evaluation for the Blend dataset. Its accuracy reaches 91.39%, and the overall accuracy of the algorithm is 1.69% higher compared to the random forest with the best recognition in Table 2. And it is found that the recognition of four categories, Agent, Expiro, Neshta, and Sality, is poor.

Through the experimental comparison of the four datasets using machine learning and deep learning algorithms, it can

TABLE 3. Deep learning algorithm comparisons.

Algorithm	Malimg			Big2015			Malevis			Blende		
	ACC (%)	Time (s)	Imp. (%)	ACC (%)	Time (s)	Imp. (%)	ACC (%)	Time (s)	Imp. (%)	ACC (%)	Time (s)	Imp. (%)
VGG16	85.67	92808.03	89.15	13.70	112406.22	43.45	86.51	91834.41	82.00	83.71	108910.73	82.69
AlexNet	90.03	14925.47	32.54	94.52	104448.56	39.11	90.12	18782.83	11.92	90.57	20818.47	9.42
DenseNet-121	95.36	53317.17	81.11	93.17	117057.16	45.65	96.70	20466.65	19.18	91.22	62622.49	69.90
MobileNetV2	93.71	19923.96	49.45	92.17	105207.88	39.55	95.83	20466.65	19.18	91.03	23570.72	20.01
ResNeXt-50 (32x4d)	94.03	59163.44	82.98	95.43	107543.06	40.86	93.18	60597.31	72.72	90.14	68743.93	72.58
ShuffleNet V2 x1.0	90.32	13280.06	24.18	95.03	109635.93	42.00	92.69	18860.12	12.31	91.32	22139.80	14.80
ConvNet	94.22	10069.97	-	96.19	63603.29	-	96.76	16539.89	-	91.39	18855.47	-

be found that the random forest algorithm performs better than the traditional machine learning algorithms (except on the Malimg dataset where it is slightly weaker than GBDT and XGBoost). The overall accuracy of the classification tasks done with CNN is above 90%, and the accuracy is improved by 1 to 2 percentage points compared with traditional machine learning algorithms. In particular, the accuracy on the Malevis dataset was improved by 7.64%.

In order to further validate the effectiveness of the model, here the current advanced deep learning models (VGG16, AlexNet, DenseNet-121, MobileNetV2, ResNeXt-50 (32 $\times$ 4d), ShuffleNet V2 $\times$ 1.0) are introduced to complete the classification task and compared by three dimensions: accuracy, time and improvement. Due to the different sizes of different models, in order to uniformly test the running time of the models, here we then modify the batc_size to 8 uniformly, and test the four datasets to obtain Table 3.

Through Table 3, we can find that we are the design of the neural network structure (ConvNet) compared with the current state-of-the-art often deep learning models, it can be found that in terms of accuracy is not weaker than the current more advanced models (only slightly weaker than the Malimg, and Big2015 dataset DenseNet-121), but the time spent on top of the time spent is significantly reduced, and can be applied to the just-in-time tasks.

As seen in Table 3, our model on the Malimg dataset uses 89.15% shorter time compared to VGG16, and still 24.18% shorter compared to ShuffleNet V2 $\times$ 1.0, which is the shortest time to use. On the Big2015 dataset, our model is 45.65% shorter than DenseNet-121 and 39.11% better than the shortest-used AlexNet model. The Malevis dataset is 82.00% shorter than the longest-used VGG16 and 11.92% better than the shortest-used AlexNet. On the Blende dataset, there was an 82.69% improvement compared to the longest used model, VGG6, and a 9.42% improvement compared to the shortest-used AlexNet.

An in-depth analysis of the reasons for this reveals that the network (ConvNet) used in this paper is simpler and has fewer layers than the current state-of-the-art deep learning models, so the training time is shorter. And due to the simple structure of the network, it has better performance in tasks that require timely processing with limited hardware resources. How to test (ConvNet) alone can be adjusted batch_size parameter to 64, I believe the time will be further reduced.

VI. CONCLUSION AND OUTLOOKS

In this paper, we optimize the BalancedDatasetSampler algorithm using data augmentation techniques and propose a malicious code detection framework (VBDN) based on malicious code visualization, balanced sampling, data enhancement, and neural network techniques. The framework is applied to four malicious code detection datasets that are currently open-source, and all of them achieve an accuracy of over 90%. It is also compared with traditional machine learning classification algorithms, and the higher accuracy rate verifies the superiority of VBDN. The excellent experimental results obtained on multiple datasets verify its robustness. Meanwhile, we further compare with advanced deep learning models, and we can find that the neural network structure we designed has a great improvement in training and recognition speed with acceptable accuracy, and we believe that we can achieve desirable results in some real-time tasks.

While verifying the effectiveness of VBDN, we also use the confusion matrix to identify some hard-to-detect malware, which we believe is of significant research interest for cybersecurity experts and researchers.

In the future, we will conduct deeper research on the virus types that are not well identified by the proposed malicious code detection framework and contribute to building a more secure cyberspace.

DATA AND CODE AVAILABILITY

BalancedDataset Sample:
https://github.com/ufoym/imbalanced-dataset-sampler/archive/master.zip

Malimg dataset:
https://www.kaggle.com/datasets/keerthichee-purupalli/malimg-dataset9010

Big2015 dataset:
https://www.kaggle.com/competitions/malware-classification/data

Malevis dataset:
https://web.cs.hacettepe.edu.tr/ selman/malevis/

Blend Dataset:
https://www.kaggle.com/datasets/gauravpendharkar/blended-malware-image-dataset

REFERENCES

[1] (2019). *Av-Test: Security Report 2019/2020*. [Online]. Available: https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf

[2] *SonicWall: 2022 Sonicwall Cyber Threat Report*. Accessed: 2022. [Online]. Available: https://www.sonicwall.com/resources/white-papers/2022-sonicwall-cyber-threat-report

[3] *SonicWall: 2022 Sonicwall Cyber Threat Report*. Accessed: 2022. [Online]. Available: https://www.sonicwall.com/2022-cyber-threat-report/sonicwall-cyberthreat-report-thank-you/

[4] (2022). *Kaspersky: A Look Back on the Year 2022 and What to Expect in 2023*. Accessed: 2022. [Online]. Available: https://securelist.com/crimeware-financial-cyberthreats-2023/108005/

[5] (2022). *RISING: 2022 China Network Security Report*. Accessed: 2022. [Online]. Available: https://www.wenjuan.com/s/FvYNrmw/

[6] X. Yang, D. Lo, L. Li, X. Xia, T. F. Bissyandé, and J. Klein, "Characterizing malicious Android apps by mining topic-specific data flow signatures," *Inf. Softw. Technol.*, vol. 90, pp. 27–39, Oct. 2017, doi: 10.1016/j.infsof.2017.04.007.

[7] W. Zhang, H. Wang, H. He, and P. Liu, "DAMBA: Detecting Android malware by ORGB analysis," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 55–69, Mar. 2020, doi: 10.1109/TR.2019.2924677.

[8] *McAfee: Mcafee Mobile Security*. Accessed: 2018. [Online]. Available: https://pccw.mcafeemobilesecurity.com/

[9] N. A. Rosli, W. Yassin, F. M. A, and S. Rahayu, "Clustering analysis for malware behavior detection using registry data," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 12, 2019, doi: 10.14569/ijacsa.2019.0101213.

[10] Y. Ding, W. Dai, S. Yan, and Y. Zhang, "Control flow-based opcode behavior analysis for malware detection," *Comput. Secur.*, vol. 44, pp. 65–74, Jul. 2014, doi: 10.1016/j.cose.2014.04.003.

[11] D. Kirat, L. Nataraj, G. Vigna, and B. S. Manjunath, "SigMal: A static signal processing based malware triage," in *Proc. 29th Annu. Comput. Secur. Appl. Conf.*, Dec. 2013, pp. 89–98, doi: 10.1145/2523649.2523682.

[12] V. Vouvoutsis, F. Casino, and C. Patsakis, "On the effectiveness of binary emulation in malware classification," *J. Inf. Secur. Appl.*, vol. 68, Aug. 2022, Art. no. 103258, doi: 10.1016/j.jisa.2022.103258.

[13] S. Liu, P. Feng, S. Wang, K. Sun, and J. Cao, "Enhancing malware analysis sandboxes with emulated user behavior," *Comput. Secur.*, vol. 115, Apr. 2022, Art. no. 102613, doi: 10.1016/j.cose.2022.102613.

[14] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Secur.*, vol. 5, no. 2, pp. 56–64, 2014, doi: 10.4236/jis.2014.52006.

[15] G. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Comput. Sci. Rev.*, vol. 47, Feb. 2023, Art. no. 100529, doi: 10.1016/j.cosrev.2022.100529.

[16] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious Web content detection by machine learning," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 55–60, Jan. 2010, doi: 10.1016/j.eswa.2009.05.023.

[17] Y. Lai and Z. Liu, "Unknown malicious code detection based on Bayesian," *Proc. Eng.*, vol. 15, pp. 3836–3842, Jan. 2011, doi: 10.1016/j.proeng.2011.08.718.

[18] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-based malware classification using ensemble of CNN architectures (IMCEC)," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101748, doi: 10.1016/j.cose.2020.101748.

[19] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for Android malware detection using various features," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 773–788, Mar. 2019, doi: 10.1109/TIFS.2018.2866319.

[20] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: Android malware characterization and detection using deep learning," *Tsinghua Sci. Technol.*, vol. 21, no. 1, pp. 114–123, Feb. 2016, doi: 10.1109/TST.2016.7399288.

[21] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *J. Signal Process. Syst.*, vol. 93, nos. 2–3, pp. 265–273, Mar. 2021, doi: 10.1007/s11265-020-01588-1.

[22] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *J. Parallel Distrib. Comput.*, vol. 129, pp. 50–58, Jul. 2019, doi: 10.1016/j.jpdc.2019.03.010.

[23] J. Hemalatha, S. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, p. 344, Mar. 2021, doi: 10.3390/e23030344.

[24] S. Abbas, S. Alsubai, S. Ojo, G. A. Sampedro, A. Almadhor, A. A. Hejaili, and I. Bouazzi, "An efficient deep recurrent neural network for detection of cyberattacks in realistic IoT environment," *J. Supercomput.*, vol. 80, no. 10, pp. 13557–13575, Jul. 2024, doi: 10.1007/s11227-024-05993-2.

[25] H. H. Ali, J. R. Naif, and W. R. Humood, "Deep learning algorithms for IoT security (survey)," *AIP Conf. Proc.*, vol. 2885, no. 1, 2024, Art. no. 060002, doi: 10.1063/5.0181698.

[26] H. Li, G. Xu, L. Wang, X. Xiao, X. Luo, G. Xu, and H. Wang, "MalCertain: Enhancing deep neural network based Android malware detection by tackling prediction uncertainty," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng.*, Apr. 2024, p. 934.

[27] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using deep learning," *J. Inf. Secur. Appl.*, vol. 72, Feb. 2023, Art. no. 103402, doi: 10.1016/j.jisa.2022.103402.

[28] Y. Zhang, J. Jiang, C. Yi, H. Li, S. Min, R. Zuo, Z. An, and Y. Yu, "A robust CNN for malware classification against executable adversarial attack," *Electronics*, vol. 13, no. 5, p. 989, Mar. 2024, doi: 10.3390/electronics13050989.

[29] X. Xu, S. Jiang, J. Zhao, and X. Wang, "DCEL: Classifier fusion model for Android malware detection," *J. Syst. Eng. Electron.*, vol. 35, no. 1, pp. 163–177, Feb. 2024, doi: 10.23919/jsee.2024.000018.

[30] P. Sathyaraj, A. S. Kumar, R. Sabitha, R. Dhanalakshmi, T. Chandrasekar, and S. Lalitha, "Efficient detection of QR code image-based attacks in industries through lightweight deep learning models and monarch butterfly optimization algorithm," in *Industry Applications of Thrust Manufacturing: Convergence With Real-Time Data and AI*. Hershey, PA, USA: IGI Global, 2024, pp. 280–313, doi: 10.4018/979-8-3693-4276-3.ch012.

[31] P. Deb, N. Kar, N. Das, and V. Datta, "Detecting malware in windows environment using machine learning," in *Proc. Int. Conf. Commun., Electron. Digit. Technol.* Cham, Switzerland: Springer, 2023, pp. 117–128, doi: 10.1007/978-981-99-1699-3_7.

[32] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques," *J. Netw. Comput. Appl.*, vol. 218, Sep. 2023, Art. no. 103704, doi: 10.1016/j.jnca.2023.103704.

[33] M. G. Twardawa, M. Smolik, F. Rakowski, J. Kwiatkowski, and N. Meyer, "SCADvanceXP—An intelligent Polish system for threat detection and monitoring of industrial networks," *Secur. Defence Quart.*, Mar. 2024, doi: 10.35467/sdq/177655.

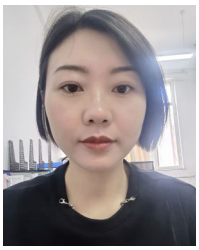
[34] S. Sadhwani, U. Modi, R. Muthalagu, and P. Pawar, "SmartSentry: Cyber threat intelligence in industrial IoT," *IEEE Access*, vol. 12, pp. 34720–34740, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10456888/

[35] M. Kim and H. Kim, "A dynamic analysis data preprocessing technique for malicious code detection with TF-IDF and sliding windows," *Electronics*, vol. 13, no. 5, p. 963, Mar. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/5/963

YAJUN LIU was born in Zhangjiakou, Hebei, China, in August 1992. He received the bachelor's and master's degrees in computer science, in June 2016 and 2019, respectively. He has been teaching and researching in computer science, since 2019. He is currently a Lecturer. He is with Hebei University of Architecture. He has published several academic papers and chaired or participated in several projects. He also leads students to participate in a variety of big data and data modeling codes and win awards. His expertise lies in deep learning, modeling, and analysis. His current main research interests include machine learning, deep learning, image processing, and artificial intelligence.

HONG FAN was born in Qingdao, China, in 1968. He received the master's degree from Hebei University of Technology. She is currently a Professor. She is with the Department of Mathematics and Physics, Hebei University of Architecture. She has led and participated in a number of national and provincial projects and has published several high-level articles. Her current research interests include deep learning, scientific computing, and optical imaging technology.

JIANGUANG ZHAO was born in 1978. He is currently pursuing the Ph.D. degree. He is a Professor. His main representative research achievements: Presided more than eight scientific research projects at provincial and municipal levels; received One-Third Prize of Hebei Provincial Teaching Achievements and three scientific and technological progress prizes at municipal and departmental levels; and presided more than three scientific research projects at all levels under research. He published more than 20 related academic articles, (SCI and EI included ten) and applied for eight patents. His main research interests include network security, pattern recognition and intelligent systems, and computer applications.



JIANFANG ZHANG was born in Zhangjiakou, Hebei, in July 1992. She received the bachelor's degree majoring in network engineering from the Inner Mongolia University of Science and Technology, in June 2016, and the master's degree majoring in electronics and communication engineering from Northeastern University, in 2019. She has been engaged in teaching and scientific research in network engineering, since 2019. Currently, she is with the School of Information Engineering, Hebei University of Architecture. She is a Lecturer. She published several academic articles and presided over or participated in a number of horizontal and vertical scientific research topics. Obtained a number of software copyrights. Also, she led students to participate in various big data and data modeling specifications and won awards. Her main research interests include elastic optical networks, network security, and data analysis.

XINXIN YIN was born in Qingdao, Shandong, China, in December 1993. She received the bachelor's degree majoring in communication engineering, in June 2016, and the master's degree majoring in systems engineering, in 2019. She has been engaged in teaching and research in computer science, since 2021. She is currently a Lecturer. She is with Hebei University of Architecture. Her current main research interests include machine learning and artificial intelligence.