Journal of Parallel and Distributed Computing 129 (2019) 50–58

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Malicious code detection based on CNNs and multi-objective algorithm

Zhihua Cui [a], Lei Du [a], Penghong Wang [a], Xingjuan Cai [a,*], Wensheng Zhang [b]

[a] Complex System and Computational Intelligence Laboratory, TaiYuan University of Science and Technology, Taiyuan, 030024, China
[b] State Key Laboratory of Intelligent Control and Management of Complex Systems, Institute of Automation Chinese Academy of Sciences, Beijing, 100190, China

HIGHLIGHTS

- A technique for converting a malware binary to an image was introduced.
- In this paper, a method based on CNN is used to identify and classify the malicious codes.
- An effective data equilibrium approach based on the NSGA-II was designed.
- The proposed method was demonstrated through the extensive experiments.

ARTICLE INFO

Article history:
Received 20 November 2018
Received in revised form 20 February 2019
Accepted 15 March 2019
Available online 21 March 2019

Keywords:
Malicious code
Deep learning
CNN
Imbalance data
NSGA-II

ABSTRACT

An increasing amount of malicious code causes harm on the internet by threatening user privacy as one of the primary sources of network security vulnerabilities. The detection of malicious code is becoming increasingly crucial, and current methods of detection require much improvement. This paper proposes a method to advance the detection of malicious code using convolutional neural networks (CNNs) and intelligence algorithm. The CNNs are used to identify and classify grayscale images converted from executable files of malicious code. Non-dominated Sorting Genetic Algorithm II (NSGA-II) is then employed to deal with the data imbalance of malware families. A series of experiments are designed for malware image data from Vision Research Lab. The experimental results demonstrate that the proposed method is effective, maintaining higher accuracy and less loss.

1. Introduction

Society has entered the era of big data. Today, convenient intelligent algorithms such as artificial intelligence, speech recognition, image recognition, and recommendation algorithms are readily available and avidly consumed. Data plays a significant role in driving algorithms to continuously update and improve. In the process of scientific research and product development, algorithms collect and utilize various user data. As this occurs, the data is exposed, and various methods are used to acquire this private data. When discussing privacy, “the single user” is emphasized and some properties of a group of users will not be considered in the domain of privacy.

The release of personally identifiable information by international companies with great influence such as Facebook has attracted much recent attention. The trouble incurred following Facebook’s information disclosure scandal that has led many enterprises to reflect on the necessity of network security and information protection. Network security is also becoming an increasing concern for internet users, and important factors that threaten network security, such as malicious code [13], are gaining attention. Malicious code can read private data by obtaining illegal rights and is a great threat to privacy security and protection [14]. According to the 2018 Symantec Internet Security Threat Report, coin mining was the biggest growth area in cybercrime in 2017, with antivirus detections up 8500%. There was also a 600% increase in attacks against IoT devices. Additionally, Symantec also reported an increase in attackers injecting malware implants into the supply chain to infiltrate unsuspecting organizations, with a 200% increase in such attacks. This works out as one every month of 2017, as compared to four attacks annually in years prior. At the same time, reports show that mobile users also face privacy security risks from grayware applications. Although these applications are not completely malicious, they can cause much trouble for users, and the development of such technologies has created challenges in the detection of malicious code [24,36,45].

Methods for detecting malicious code have progressed rapidly in recent years [23]. Because of the image field technology development [22], the employment of convolutional neural networks (CNNs) is very common today. Some scholars [10] have used

* Corresponding author.
E-mail addresses: zhihua.cui@hotmail.com (Z. Cui), dulei2468@163.com (L. Du), penghongwang@sina.cn (P. Wang), xingjuancai@163.com (X. Cai), wensheng.zhang@ia.ac.cn (W. Zhang).

https://doi.org/10.1016/j.jpdc.2019.03.010
0743-7315/© 2019 Elsevier Inc. All rights reserved.

CNNs to identify and classify malware images alongside intelligence algorithms to deal with the balance problem of malware families. This approach has greatly improved the accuracy of the model and works well on the malware dataset. However, there are some irrationalities. Single evaluation criteria cannot represent the performance of the built model. For example, there it often happens in the CNNs models that the accuracy of the model is very high, but the recall rate is not very good. The reason is the model built does not think about the imbalance of the dataset carefully [18]. The capacity to deal with the imbalance data is insufficient. To make up for this shortcoming, various criteria are required to evaluate a model, with more criteria proving the effectiveness of the model from different aspects.

Therefore, more than one criterion is employed to evaluate the model in this paper. The more criteria will need corresponding algorithms to deal with. So, the multi-objective algorithm into our view [7,8,11,22,26,44]. The multi-objective algorithm can deal with the imbalance problems effectively [48]. The multi-objective algorithm has been developed in recent years [1,3,4,16,32,43]. More and more strategies and methods are proposed on the multi-objective fields [2,9,25,42,49]. The multi-objective algorithms are also combined with other algorithms to deal with various problems [12,28,37,40]. The non-dominated sorting genetic algorithm II has been applied to various fields [50]. The CNNs and NSGA-II both have good performance in relative fields. Therefore, we combine the CNNs with NSGA-II to deal with the detection of malicious code. Here, the non-dominated sorting genetic algorithm II (NSGA-II) is used to manage the imbalance problem of malware images, then CNNs are used to identify and classify the malware families. The proposed approach could avoid problems raised by dataset imbalance, and results show that this method performs well on the malware dataset.

The remainder of this paper is structured as follows. Section 2 presents related work including malware detection methods. Section 3 introduces the methods and principles used in this article. Section 4 discusses the proposed approach based on CNNs and NSGA-II. Section 5 introduces the experiments conducted on the relative dataset and, finally, conclusions are discussed in Section 6.

2. Related work

In this section, related research regarding malware detection and existing detection methods are introduced and discussed.

Detection methods based on feature analysis are divided into two categories, the static method and the dynamic method.

The static method is a commonly used existing approach. Brian et al. [5] developed a method for finding security flaws in source code by way of static analysis. They also demonstrated that the method can be used to identify real vulnerabilities in existing programs. However, an issue with this method is that the validity of static analysis is greatly influenced by obfuscation techniques [27].

To deal with code obfuscation, Sharif et al. [35] presented a framework for detecting malicious patterns in executables that is resilient to common obfuscation transformations. The research demonstrates the efficacy of the architecture; however, the approach is limited to feature extraction and analysis at instruction level, and many techniques exist to easily deceive the static method.

Dynamic analysis monitors and analyzes the runtime characteristics of applications based on assessment of behaviors such as accessing private data and using restricted API calls. Given this information, a behavior model is established to detect malicious code. Such techniques have demonstrated improved detection performance, but were still challenged by the variety of countermeasures developed to generate unreliable results [30]. In addition, dynamic analysis is time consuming because of the large amount of computation required, leading to low efficiency when exposed to a large dataset.

As previously mentioned, the method of malicious code visualization has been used successfully [34,39]. Yoo et al. [31] utilized self-organizing maps to visualize computer viruses. A similar but expanded approach was proposed by Trinius et al. [10], who used two visualization techniques, tree maps and thread graphs, to detect and classify malware. Rather than acquiring a single detection result, Goodall et al. [17] aggregated the results of different malware analysis tools into a visual environment, providing an increase in the vulnerability of detection coverage of single malware tools.

The studies discussed above focus mainly on the visualization of malware behavior [38,41,46], however, the software source code may provide more meaningful patterns. As previously mentioned, Nataraj et al. [29] presented a new visualization approach for malware detection based on binary texture analysis. First, they converted the malware executable file into grayscale images, then they identified malware according to the texture features of these images. Compared with the dynamic analysis method, this approach produced equivalent results [29]. In similar work, Han et al. [15] transformed malware binary information into color image matrices, and classified malware families by using an image processing method. But the method they used have a disadvantage is resource consuming high. Once the malware is visualized as grayscale images, malware detection can be converted into an image recognition problem. In [30], Nataraj et al. used a Generalized Search Tree (GiST) algorithm to extract the features of malware images. However, operation of the GiST algorithm is time consuming, and more powerful image processing techniques have been proposed recently.

For image fusion, Miao et al. [26] proposed an image fusion algorithm based on shearlet and genetic algorithm. In their approach, a genetic algorithm was employed to optimize the weighted factors in the fusion rule. Experimental results demonstrated that this method could acquire better fusion quality than other methods.

Cui et al. [10] developed an approach to combine the CNN with the Bat algorithm to improve the accuracy of the model. The Bat algorithm was used to balance the image dataset and the convolutional neural network was used to identify and classify the images. But they use one evaluation criterion to test the model. As we state in Section 1, Single evaluation criteria cannot represent the performance of the built model.

Deep learning [33,47] is an area of machine learning research that has emerged in recent years from work on artificial neural networks. Neural networks can utilize approximate complex functions by learning the deep nonlinear network structure to solve complex problems. The algorithms for multi-objective problems have been well developed in recent years. This paper proposes the detection of malware based on deep learning and the multi-objective algorithm. CNNs can train the identified models conveniently. And NSGA-II can deal with the imbalance problems of dataset. So we can avoid the problems summarized above effectively.

3. Method and principle

This section introduces some methods and principles used in the proposed method, including malicious code visualization, CNNs and NSGA-II.

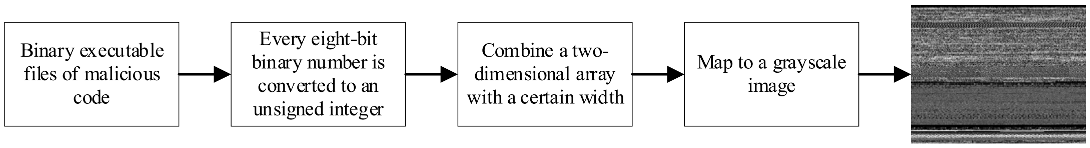

Binary executable files of malicious code
Every eight-bit binary number is converted to an unsigned integer
Combine a two-dimensional array with a certain width
Map to a grayscale image

Fig. 1. Flow chart of malicious code visualization.

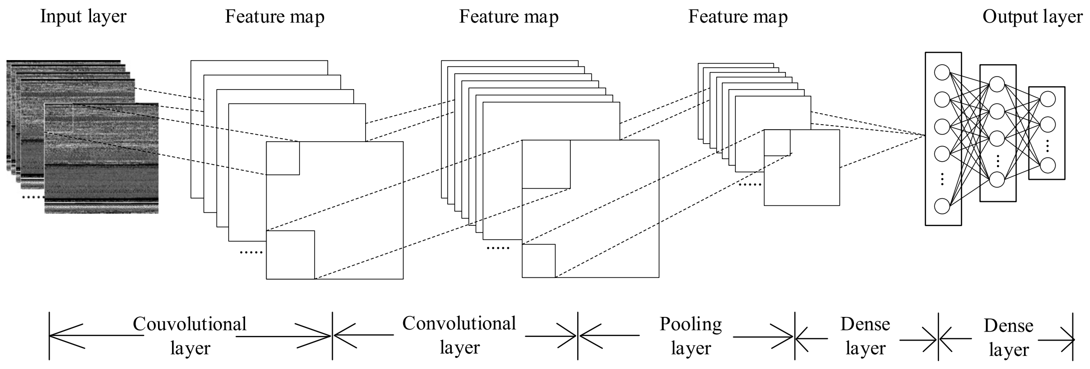


Fig. 2. Structure of neural network.

3.1. Visualization of malicious code

Executable binary files of malicious code can be divided into 8 bit length by algorithm. Every 8 bit can be converted to an unsigned integer number ranging from 0 to 255 (a grayscale pixel). These are then combined into a two-dimensional array according to a certain width provided in [29]. In this way, the two-dimensional image can be mapped to a grayscale image. Fig. 1 shows the process of conversion.

Through the process of conversion, it is visible that the same kind of malware families have the same texture and features. Thus, CNN can be used to identify and classify the gray images converted from the executable binary files of malicious code.

3.2. Structure of CNN

Convolutional neural networks are a recent neural network development which has gathered much attention [20]. The CNNs are mainly used for reorganization and classification of images. The neural network is an operational model in which a large number of nodes (or neurons) are connected to each other. Every node represents a special output function which is called an activation function. Each connection between two nodes is represented by a weighted value for continuous signal called weight [6]. The weight amounts to the memory of the neural network. The outputs of the neural networks are different because of the connection mode, weight value, and activation function. The network itself is generally an approximation of algorithms or functions, or an expression of a logical strategy.

The structure of the neural network used in this paper has two convolution layers, one pooling layer, and two dense layers. The structure is illustrated in Fig. 2.

(1) Convolutional layer

The convolutional layer of CNN consists of several convolution units. Convolution is a common linear filtering method in image processing. The premise of convolution is to extract the different features of the input image.

The formula for the calculation of the convolutional layer is as follows:

$$FM_{i,j} = f(\sum_m \sum_n w_{m,n} x_{i+m,j+n} + w_b) \tag{1}$$

The $x_{i,j}$ represent the row i and column j pixel of image input. The $w_{m,n}$ represent the row m column n pixel of filter. The w_b represent the bias. The f here represents the activation function. The activation function used is Rectified Linear Unit (ReLU). The $FM_{i,j}$ represent row i and column j element of feature map output. The ReLU is as follows:

$$f(x) = \max(0, x) \tag{2}$$

which means $f(x)$ is the bigger one between 0 and x.

(2) Pooling layer

The pooling layer is mainly downsampling, which is to lower the dimensions. The number of parameters is reduced by taking out the unnecessary samples in the features map. There are many methods for pooling, with mean pooling and max pooling generally used by the authors. Mean pooling the average value of the image area calculated as the pooling value of the area, while in max pooling the max value of the image area is the pooling value of the area. Max pooling is used in this paper.

(3) Dense layer

After convolution and pooling, there are dense layers in the CNN. Each node in the dense layer is fully connected to all nodes in the previous layer. The function of the dense layer is to integrate the local information with category differentiation in the convolutional layer or pooling layer. The activation function of the dense layer used in this paper is ReLU. The output of the last layer used is the softmax regression.

3.3. NSGA-II

Intelligent optimization algorithm is a heuristic algorithm. It is challenged by optimization problems but performs well for nondeterministic optimization problems.

Generally, optimization problems which have 2∼3 objects are called multi-objective optimization problems (MOPs) [21]. The definition of MOPs is as follows:

$$\min f(X) = \min[f_1(X), f_2(X), \ldots, f_m(X)] \tag{3}$$

$$X = (x_1, x_2, \ldots, x_n) \in R^n \tag{4}$$

$$\begin{cases} g_i(X) \geq 0, i = 1, 2, \ldots, k \\ h_j(X) = 0, j = 1, 2, \ldots, p \end{cases} \tag{5}$$

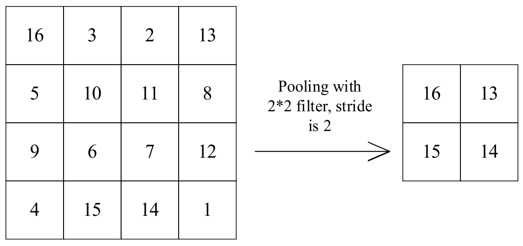


Fig. 3. Max pooling.

The $f_m(X)$ is mth objective function, X is a solution vector with n-dimensional variable, and satisfies the conditions above. R^n is decision variable space. $g_i(X) \geq 0$ is inequality constraint. $h_j(X) = 0$ is equality constraint.

The imbalance dataset in this paper can be seen as a MOP, and NSGA-II is used.

(1) Srinivas and Deb et al. came up with NSGA in 1993 [17]. The drawbacks to NSGA are the shared parameters, that the time complexity of constructing Pareto optimal solution set is too high, and

(2) there is no optimal individual retention mechanism

To mitigate these issues, Deb et al. proposed NSGA-II based on NSGA in 2000 [19]. The NSGA-II is a non-dominated sorting genetic algorithm with elitist strategy which is widely used at present. The important aspects of NSGA-II include fast non-dominating sorting and calculation of crowding distance.

In order to maintain the distribution and diversity of solution groups, Deb et al. proposed that the crowding distance of each individual in an evolutionary population is calculated, then according to the layer and crowding distance of individual, a partial order set is defined. The individual ordinally is then chosen when building the new population in the partial order set.

Here, when the new population is generated, the individuals which are appropriate and have a small aggregation density are generally retained and will then participate in the next evolution. Individuals with small aggregation density have larger aggregation distances. The aggregation distance of an individual can be obtained by calculating the sum of the distance difference between two neighboring individuals on each sub-target. The features of the NSGA-II mean it can be used in the proposed method.

4. Proposed work

An imbalanced dataset will have a negative influence on the results of the model. As seen in Fig. 3, the quantities of some malware images are much larger than others. To deal with this problem, as described in Section 2, some scholars have proposed the use of a single-objective algorithm to treat the imbalance problem. However, this technique has limitations as a single evaluation criterion cannot evaluate a model correctly. The use of a multi-objective algorithm is suggested to deal with the imbalance problem of a dataset.

The resampling method is proposed to resolve the imbalanced dataset by processing the training set and producing a balanced dataset. This technique consists of two implementations: oversampling and undersampling. Undersampling is used to remove particular samples from the sample set. We utilize the undersampling in this paper.

The design thought is achieving solution in iterative algorithm by NSGA-II. We employ it to optimize the sampling weights of multiple malware families, which means the quantity of each kind of the malware families. Suppose the number of malicious code families is N, the quantity of each kind of the malware families is denoted by ω_i, $i \subset N$. For this optimization, each individual of the population is a combination of sampling quantity that is given by the following equation:

$$quantities = \omega_1, \omega_2, \ldots, \omega_n. \tag{6}$$

According to the solution, the part corresponding to numbers of images is input into the neural network for training. After training, the model is verified with the other elements of the dataset. The True Positive Rates and False Positive Rates are calculated. Then "1- True Positive Rates" and False Positive Rates are as two objects for NSGA-II. So we use NSGA-II obtain the best two objects by serial Loops and iterations operations, the result can then be acquired.

The pseudocode of algorithm is as follows.

The pseudocode of algorithm in this paper

```
Begin
    P_i: initial population;
    MaxGen: generation numbers;
    Training the CNN model;
    Calculate TPR and FPR; //See in Section V
    F=nondominated-sort(P_i); //Non-dominated sort
    D=Crowding-distance(F); //Calculate the crowding distance
    Genetic operation;
        Select();//Roulette wheel selection
        Crossover();//Multiple-point crossover
        Mutation();//Simple Mutation
    While(Gen<MaxGen)
        R= $P \cup Q$; Q: children population;
        Training the CNN model;
        Calculate TPR and FPR;
        F=fast nondominated-sort(P_i); //Non-dominated sort
        D=Crowding-distance(F); //Calculate the crowding distance
        P=new P;
        Genetic operation;
            Select();//Roulette wheel selection
            Crossover();//Multiple-point crossover
            Mutation();//Simple Mutation
        Generate Q;
    End while
End
```

Through the pseudocode, it can be seen that the quantity of malware families can be regarded as an individual unit. The NSGA-II is an effective solution for complex optimization problems and thus is used to acquire the solution. The object values will be obtained from the training of CNNs. The 1-TPR and FPR are two objective functions (Section 5). We draw the Pareto set in the figure. And we choose the solution that 1-TPR and FPR are both less at the same time. The aim of the approach is to find the optimal individual which meets the evaluation criteria.

5. Experimental evaluation

5.1. Dataset and evaluation criterion

The dataset used in this paper is the malware image data from Vision Research Lab. Here, 24 are selected to conduct the

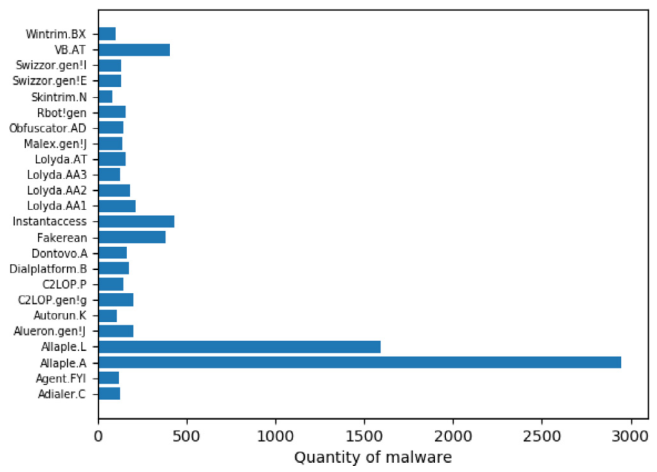


Fig. 4. Quantity of malware images in different families.

Table 5.1
Confusion matrix.

True category	Predict category	
	Positive samples	Negative samples
Positive samples	TP	FN
Negative samples	FP	TN

experiments. In addition, for the convenience of experiment, we use 50*50 100*100 and 120*120 resolution images. The quantity of each malicious code is as follows:

Through the table in Fig. 4 it can be seen that the number of different species varies greatly. Overfit or bad states can be avoided by solving the imbalance problems.

Correct selection of evaluation criteria is crucial for evaluating the merits of a model. In the classification problems, the confusion matrix is used to evaluate analytical models. For the two-classification problems, the samples can be divided according to the true category and predicted category as follows:

True Positive (TP): The true category is positive, predicted category is positive.

False Positive (FP): The true category is negative, predicted category is positive.

False Negative (FN): The true category is positive, predicted category is negative.

True Negative (TN): The true category is negative, predicted category is negative.

The confusion matrix is created as in Table 5.1.

Through the confusion matrix, other evaluation indicators can be calculated including:

Accuracy, representing the proportion of all modules correctly predicted:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

Recall, also called true positive rate (TPR), which means the proportion of predict positive in all positive samples.

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

False Positive Rate (FPR), which means the proportion of predict positive in all negative samples.

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

Multi-objective balance problems generally select TPR and FPR as the objects which need to be dealt with.

However, these evaluation indicators concern two classification problems. The problems in this paper belong to multiclass classification and so a method to calculate the TPR and FPR is required.

Results of the prediction show that some predicted samples are bigger than true samples, some are less, and some are the same as the true samples. Therefore, the predicted samples are divided into two parts according to the category's numbers of dataset. for example, the category's number of datasets in this paper we use is 24. So we regard the category 1 to 12 as the front half, and the category 13 to 24 as the back half. The cases are provided in Table 5.2.

Following this process, the evaluation indicators above can be calculated. This method demonstrates strong performance through verification by experiments.

The loss function, cross-entropy, describes the distance between two probability distributions, and the smaller the cross-entropy is, the closer the two are. Cross-entropy is a kind of loss function widely used in classification problems.

Given two probability distributions, p and q, the cross-entropy of p is expressed by q as follows:

$$H(p, q) = -\sum_{x} p(x) \log q(x) \tag{10}$$

The cross-entropy describes the distance between two probability distributions, but the output of the neural network is not necessarily a probability distribution. Probability distribution describes the probability of different events. When the total number of events is limited, the probability distribution function $p(X = x)$ satisfies:

$$\forall x\ p(X = x) \in [0, 1]\ \&\ \sum_{x} p(X = x) = 1 \tag{11}$$

Softmax regression is a common method to change the result of forward propagation of neural network into probability distribution. Softmax regression itself can be used as a learning algorithm to optimize classification results, but in TensorFlow, the parameters of Softmax regression are removed, and it is just an extra layer of processing that turns the output of the neural network into a probability distribution. Therefore, we can use the loss function value to evaluate the effect of the model we built.

5.2. Experimental results

To validate the effectiveness and efficiency of the proposed approach, experiments are designed to clear out the following:

(1) How the data balance method used will affect the results;

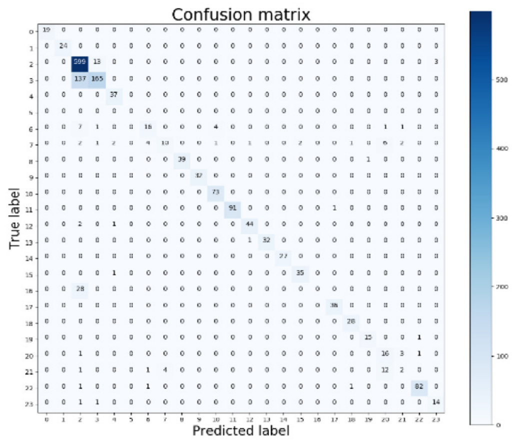
(2) The effect of image resolution on experimental results.

Experiment 1 aims to verify the effect of different data equalization methods on the results. Results are provided in Table 5.3.

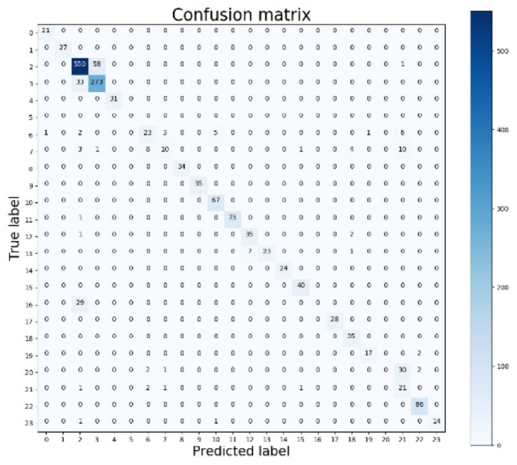
The results show that there is a large accuracy and recall improvement, as well as reduced loss and TPR. Compared with the

Table 5.2
The classification cases.

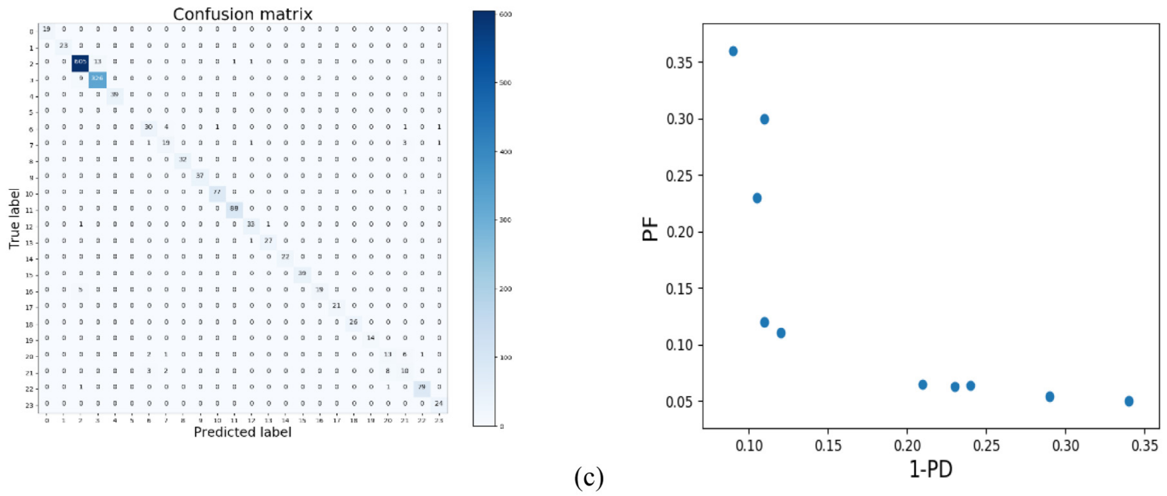
Cases	Predict correct category belong to the front half	Predict correct category belong to the back half	Predict samples are bigger than true samples	Predict samples are less than true samples
Representation	0	1	0	1
	0	1	1	0



(a)



(b)



(c)

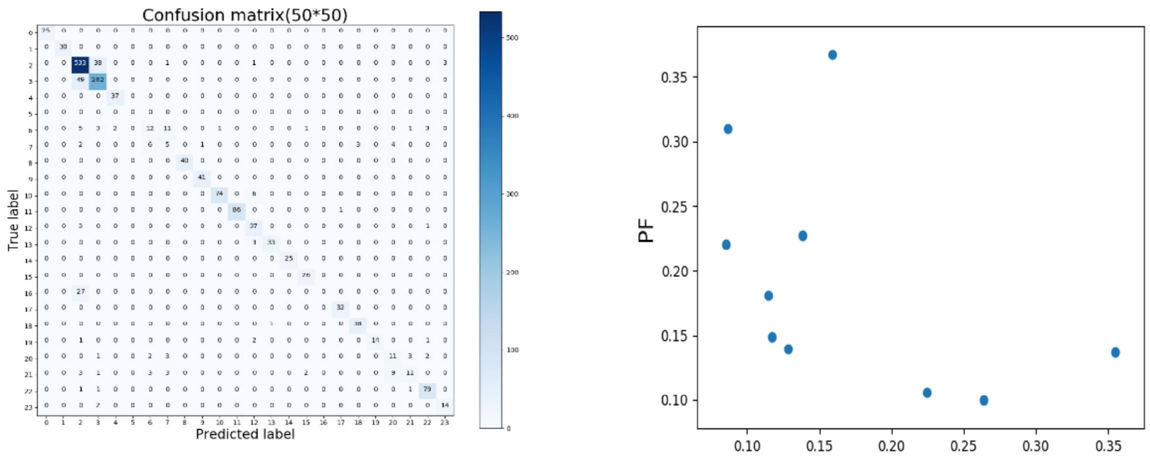
Fig. 5. The results of experiment 1. (a) Without data equalization. (b) With single-objective algorithm. (c) With multi-objective algorithm.

single-objective algorithm, the proposed approach has a positive effect on the malware images dataset. The confusion matrices are as follows.

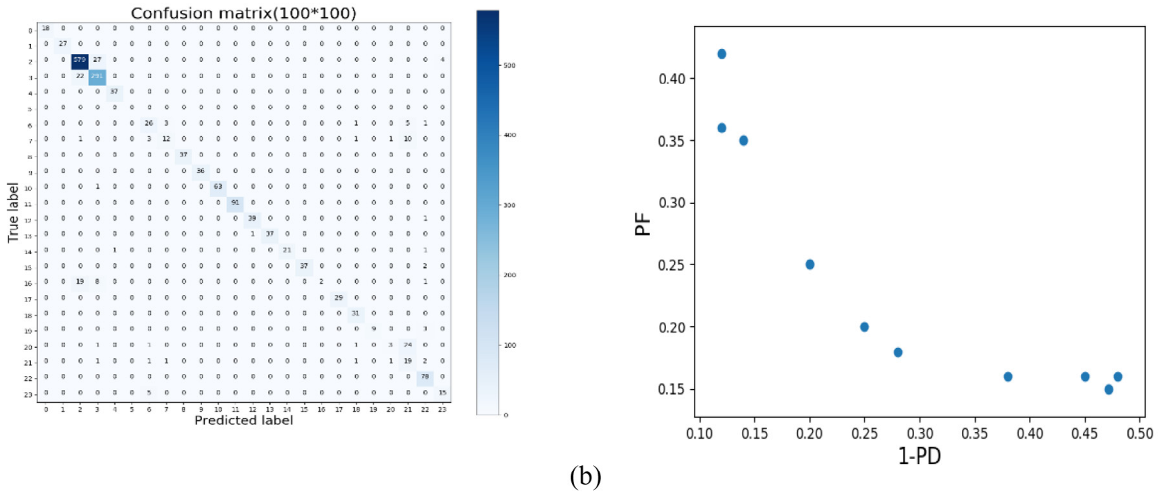
From Fig. 5(a), (b) and (c), it can be seen that without data equalization, the confusion matrix in (c) is better than the confusion matrix in (a) and (b). Training without data equalization may cause overfit. The multi-objective algorithm shows the best effect on the malware image dataset, and data balance deals with this problem effectively.

The second experiment aims to verify the effect of image resolution on the experimental results.

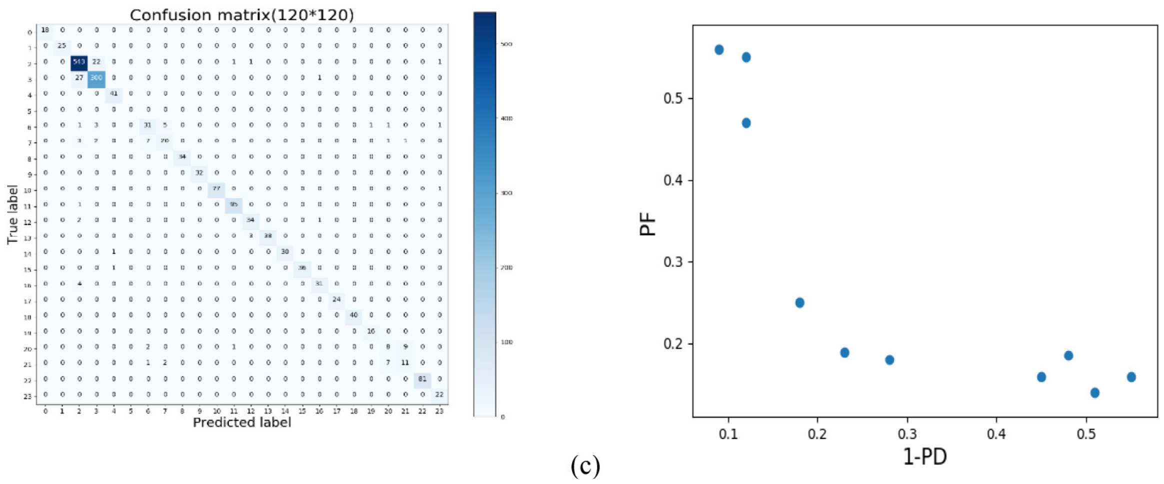
As seen in Table 5.4, if the image resolution is large, the accuracy is improved and loss reduced. The recall and TPR are also enhanced.



(a)



(b)



(c)

Fig. 6. The results of experiment 2.

In Fig. 6, it can be seen that the resolution of images is crucial to the results. Higher resolution will obtain higher accuracy and less loss. The Pareto-optimal fronts visible in Fig. 6 clearly illustrate the large influence of image resolution on the dataset.

6. Conclusion

In this paper, a method using the NSGA-II is proposed to balance the dataset. First, the binary converted the executable files of malicious code into grayscale images and then CNN was

Table 5.3
Results of experiment 1.

Batch number	Without data equalization				With single-objective algorithm				With multi-objective algorithm			
	ACY	Loss	Recall	FPR	ACY	Loss	Recall	FPR	ACY	Loss	Recall	FPR
1	93.4%	0.21	83.5%	9.8%	96.0%	0.13	85.6%	9.3%	97.6%	0.10	87.5%	8.2%
2	92.4%	0.25	80.1%	11.2%	95.4%	0.19	84.8%	10.5%	96.9%	0.11	87.2%	8.6%
3	92.6%	0.26	79.4%	10.9%	95.8%	0.13	85.8%	8.9%	96.2%	0.10	86.8%	9.3%
4	91.9%	0.25	79.6%	13.2%	95.9%	0.11	86.2%	11.2%	96.5%	0.10	85.9%	10.2%
5	92.1%	0.22	81.3%	10.8%	96.1%	0.14	85.3%	9.8%	97.1%	0.09	88.1%	7.8%

Table 5.4
Results of experiment 2.

Resolution	ACY	Loss	Recall	FPR
50*50	96.0%	0.17	82.6%	11.5%
100*100	96.9%	0.10	85.6%	10.3%
120*120	97.6%	0.08	88.4%	8.3%

used to identify and classify the images. An imbalanced dataset was found to have a negative influence on the experimental results. Additionally, NSGA-II was found to preform very well on uncertain problems. Thus, NSGA-II was selected to balance the dataset. Detecting the malicious code based on deep learning presented a great improvement when compared with traditional methods and the experiment results verified the effectiveness of the method proposed in this paper. In this study, accuracy was improved and loss of the model was reduced. We conducted the work on 50*50 100*100 and 120*120 resolution problem in this paper. But in natural world, the resolution of images is much bigger than these. We will do more research on the bigger scale images in the future work, and future research also would focus on time reduction and accelerated network training.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61806138, U1636220 and 61663028, Natural Science Foundation of Shanxi Province, China under Grant No. 201801D121127, Scientific and Technological innovation Team of Shanxi Province, China under Grant No. 201805D131007, PhD Research Startup Foundation of Taiyuan University of Science and Technology, China under Grant No. 20182002.

References

[1] H. Bao, Existence and stability of anti-periodic solutions for FCNNs with time-varying delays and impulsive effects on time scales, Int. J. Comput. Sci. Math. 9 (5) (2018) 474–483.
[2] S.K. Biswas, M. Chakraborty, B. Purkayastha, A rule generation algorithm from neural network using classified and misclassified data, Int. J. Bio-Inspired Comput. 11 (1) (2018) 60–70.
[3] X. Cai, X. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, Int. J. Bio-Inspired Comput. 8 (4) (2016) 205–214.
[4] X. Cai, H. Wang, Z. Cui, et al., Bat algorithm with triangle-flipping strategy for numerical optimization, Int. J. Mach. Learn. Cybern. 9 (2) (2018) 199–215.
[5] B.V. Chess, Improving computer security using extended static checking, in: Proceedings 2002 IEEE Symposium on Security and Privacy, IEEE, 2002, pp. 160–173.
[6] D. Cireşan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, arXiv preprint arXiv:1202.2745, 2012.
[7] Z. Cui, Y. Cao, X. Cai, J. Cai, J. Chen, Optimal LEACH protocol with modified bat algorithm for big data sensing systems in Internet of Things, J. Parallel Distrib. Comput. (2017) http://dx.doi.org/10.1016/j.jpdc.2017.12.014.
[8] Z. Cui, F. Li, W. Zhang, Bat algorithm with principal component analysis, Int. J. Mach. Learn. Cybern. (2018) 1–20.
[9] Z. Cui, B. Sun, G. Wang, Y. Xue, J. Chen, A novel oriented cuckoo search algorithm to improve DV-hop performance for cyber-physical systems, J. Parallel Distrib. Comput. 103 (2017) 42–52.
[10] Z. Cui, F. Xue, X. Cai, et al., Detection of malicious code variants based on deep learning, IEEE Trans. Ind. Inf. 14 (7) (2018) 3187–3196.
[11] Z. Cui, J. Zhang, Y. Wang, Y. Cao, X. Cai, W. Zhang, J. Chen, A pigeon-inspired optimization algorithm for many-objective optimization problems, Sci China Inf. Sci. (2019) http://dx.doi.org/10.1007/s11432-018-9729-5.
[12] Taciana Araújo De Souza, Vinícius J.D. Vieira, Micael Andrade De Souza, Suzete E.N. Correia, Silvana C. Costa, Washington C. De Almeida Costa, Feature selection based on binary particle swarm optimisation and neural networks for pathological voice detection, Int. J. Bio-Inspired Comput. 11 (2) (2018) 91–101.
[13] E. Gandotra, D. Bansal, S. Sofat, Malware analysis and classification: A survey, J. Inf. Secur. 5 (02) (2014) 56.
[14] K. Grosse, N. Papernot, P. Manoharan, et al., Adversarial examples for malware detection, in: European Symposium on Research in Computer Security, Springer, Cham, 2017, pp. 62–79.
[15] K.S. Han, J.H. Lim, E.G. Im, Malware analysis method using visualization of binary files, in: Proceedings of the 2013 Research in Adaptive and Convergent Systems, ACM, 2013, pp. 317–321.
[16] Y. He, G. Li, Y. Sun, et al., Temperature intelligent prediction model of coke oven flue based on CBR and RBFNN, Int. J. Comput. Sci. Math. 9 (4) (2018) 327–339.
[17] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[18] W. Jie, Y. Jiangjun, A high-efficient multi-deme genetic algorithm with better load-balance, Int. J. Comput. Sci. Math. 9 (3) (2018) 240–246.
[19] D. Kalyanmoy, A. Pratap, S. Agarwal, et al., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[20] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.
[21] B. Li, J. Li, K. Tang, et al., Many-objective evolutionary algorithms: A survey, ACM Comput. Surv. 48 (1) (2015) 13.
[22] Y. Li, H. Yu, B. Song, et al., Image encryption based on a single-round dictionary and chaotic sequences in cloud computing, Concurr. Comput.: Pract. Exper. (2019) http://dx.doi.org/10.1002/cpe.5182, (in press).
[23] Y. Li, Z. Zuo, An overview of object-code obfuscation technologies, Comput. Technol. Dev. (2007) 4.
[24] C. Liu, N. Beaugeard, C. Yang, X. Zhang, J. Chen, HKE-BC: hierarchical key exchange for secure scheduling and auditing of big data in cloud computing, Concurr. Comput.: Pract. Exper. 28 (3) (2016) 646–660.
[25] C. Ma, Network optimisation design of hazmat based on multi-objective genetic algorithm under the uncertain environment, Int. J. Bio-Inspired Comput. 12 (4) (2018) 236–244.
[26] Q. Miao, R. Liu, Y. Wang, et al., Remote sensing image fusion based on shearlet and genetic algorithm, in: Bio-Inspired Computing-Theories and Applications, Springer, Berlin, Heidelberg, 2015, pp. 283–294.
[27] A. Moser, C. Kruegel, E. Kirda, Limits of static analysis for malware detection, in: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), IEEE, 2007, pp. 421–430.
[28] M. Narasimhan, B. Balasubramanian, S.D. Kumar, et al., EGA-FMC: enhanced genetic algorithm-based fuzzy k-modes clustering for categorical data, Int. J. Bio-Inspired Comput. 11 (4) (2018) 219–228.
[29] L. Nataraj, S. Karthikeyan, G. Jacob, et al., Malware images: visualization and automatic classification, in: Proceedings of the 8th International Symposium on Visualization for Cyber Security, Vol. 4, ACM, 2011.
[30] L. Nataraj, V. Yegneswaran, P. Porras, et al., A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, ACM, 2011, pp. 21–30.
[31] L. Nataraj, V. Yegneswaran, P. Porras, et al., A comparative assessment of malware classification using binary texture analysis and dynamic analysis, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, ACM, 2011, pp. 21–30.
[32] B. Niu, J. Liu, L. Tan, Multi-swarm cooperative multi-objective bacterial foraging optimisation, Int. J. Bio-Inspired Comput. 13 (1) (2019) 21–31.
[33] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117.

[34] S.Z.M. Shaid, M.A. Maarof, Malware behavior image for malware variant identification, in: 2014 International Symposium on Biometrics and Security Technologies (ISBAST), IEEE, 2014, pp. 238–243.
[35] M. Sharif, V. Yegneswaran, H. Saidi, et al., Eureka: A framework for enabling static malware analysis, in: European Symposium on Research in Computer Security, Springer, Berlin, Heidelberg, 2008, pp. 481–500.
[36] K. Tam, A. Feizollah, N.B. Anuar, et al., The evolution of android malware and android analysis techniques, ACM Comput. Surv. 49 (4) (2017) 76.
[37] Y.M. Tavares, N. Nedjah, Luiza De Macedo Mourelle, Embedded implementation of template matching using correlation and particle swarm optimization, Int. J. Bio-Inspired Comput. 11 (2) (2018) 102–109.
[38] P. Trinius, T. Holz, J. Göbel, et al., Visual analysis of malware behavior using treemaps and thread graphs, in: 2009 6th International Workshop on Visualization for Cyber Security, IEEE, 2009, pp. 33–38.
[39] M. Wagner, F. Fischer, R. Luh, et al., A survey of visualization systems for malware analysis, in: EG conference on visualization (EuroVis)-STARs, The EGA, 2015, pp. 105–125.
[40] G. Wang, X. Cai, Z. Cui, G. Min, J. Chen, High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm, IEEE Trans. Emerg. Top. Comput. (2017) http://dx.doi.org/10.1109/TETC.2017.2703784.
[41] Z. Wang, F. Wang, G. Chi, A research on defect image enhancement based on partial differential equation of quantum mechanics, Int. J. Comput. Sci. Math. 9 (2) (2018) 122–132.
[42] H. Wang, W. Wang, X. Zhou, et al., Firefly algorithm with neighborhood attraction, Inform. Sci. 382 (2017) 374–387.
[43] P. Wang, F. Xue, H. Li, Z. Cui, L. Xie, J. Chen, A multi-objective DVHop localization algorithm based on NSGAII in Internet of Things, Mathematics 7 (2) (2019) 184, http://dx.doi.org/10.3390/math7020184.
[44] Y. Wang, et al., A novel bat algorithm with multiple strategies coupling for numerical optimization, Mathematics 7 (2) (2019) 135, http://dx.doi.org/10.3390/math7020135.
[45] Y. Ye, T. Li, D. Adjeroh, et al., A survey on malware detection using data mining techniques, ACM Comput. Surv. 50 (3) (2017) 41.
[46] I.S. Yoo, Visualizing windows executable viruses using self-organizing maps, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, 2004, pp. 82–89.
[47] Z. Yuan, Y. Lu, Z. Wang, et al., Droid-sec: deep learning in android malware detection, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 371–372, ACM.
[48] M. Zhang, H. Wang, Z. Cui, et al., Hybrid multi-objective cuckoo search with dynamical local search, Memetic Comput. 10 (2) (2018) 199–208.
[49] J. Zhang, A. Zhou, G. Zhang, Preselection via classification: a case study on global optimisation, Int. J. Bio-Inspired Comput. 11 (4) (2018) 267–281.
[50] B. Zhao, Y. Xue, B. Xu, et al., Multi-objective classification based on NSGA-II, Int. J. Comput. Sci. Math. 9 (6) (2018) 539–546.

Zhihua Cui is a Professor of Computer Science and Technology, and Director of Complex System and Computational Intelligence Laboratory at Taiyuan University of Science and Technology, China. He is a member of IEEE and ACM, and a senior member of China Computer Federation (CCF) and member of Chinese Association of Artificial Intelligence (CAAI). He received his Ph.D. in System Engineering from Xi'an Jiaotong University, China in 2008, and B.Sc. in Computer Science from Taiyuan Heavy Machinery Institute in 2003. He is the founding Editor-in-Chief of International Journal of Bio-inspired Computation. His research interest includes computational intelligence, stochastic algorithm, and combinatorial optimization. He has published over 60 peer reviewed journal papers, 60 peer reviewed full conference papers, and five books in computational intelligence.

Lei Du is currently working toward M.S. degree in computer science and technology, at Taiyuan University of Science and Technology, Taiyuan, China. His main research includes Computational intelligence and Deep learning.

Penghong Wang is currently working toward M.S. degree in computer science and technology, at Taiyuan University of Science and Technology, Taiyuan, China. His main research includes computational intelligence and combinatorial optimization.

Xingjuan Cai received her Ph.D. degree in Control Science and Engineering from Tongji University, China, in 2017. She is an Associate Professor with the School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan, China. Her research interest includes bio-inspired computation and application.

Wensheng Zhang received the Ph.D. degree in Pattern Recognition and Intelligent Systems from the Institute of Automation, Chinese Academy of Sciences (CAS), in 2000. He joined the Institute of Software, CAS, in 2001. He is a Professor of Machine Learning and Data Mining and the Director of Research and Development Department, Institute of Automation, CAS. His research interests include computer vision, pattern recognition, artificial intelligence and computer human interaction.