Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(5): 5995–6011.
DOI: 10.3934/mbe.2021300
Received: 06 April 2021
Accepted: 24 May 2021
Published: 02 July 2021

Research article

Malware detection based on semi-supervised learning with malware visualization

Tan Gao [1], Lan Zhao [2,*], Xudong Li [1] and Wen Chen [1]

[1] School of Cyber Science and Engineering, Sichuan University, China
[2] Science and Technology on Electronic Information Control Laboratory, China

* **Correspondence:** Email: 93402684@qq.com; Tel: +86-28-85405568.

Abstract: The traditional signature-based detection method requires detailed manual analysis to extract the signatures of malicious samples, and requires a large number of manual markers to maintain the signature library, which brings a great time and resource costs, and makes it difficult to adapt to the rapid generation and mutation of malware. Methods based on traditional machine learning often require a lot of time and resources in sample labeling, which results in a sufficient inventory of unlabeled samples but not directly usable. In view of these issues, this paper proposes an effective malware classification framework based on malware visualization and semi-supervised learning. This framework includes mainly three parts: malware visualization, feature extraction, and classification algorithm. Firstly, binary files are processed directly through visual methods, without assembly, decompression, and decryption; Then the global and local features of the gray image are extracted, and the visual image features extracted are fused on the whole by a special feature fusion method to eliminate the exclusion between different feature variables. Finally, an improved collaborative learning algorithm is proposed to continuously train and optimize the classifier by introducing features of inexpensive unlabeled samples. The proposed framework was evaluated over two extensively researched benchmark datasets, i.e., Malimg and Microsoft. The results show that compared with traditional machine learning algorithms, the improved collaborative learning algorithm can not only reduce the cost of sample labeling but also can continuously improve the model performance through the input of unlabeled samples, thereby achieving higher classification accuracy.

Keywords: Malicious sample detection; collaborative learning; feature fusion; noise robustness

1. Introduction

Malware has become a major threat to network security [1]. Traditional signature-based methods extract binary signatures from malware to build a huge feature library, which provides comprehensive information of malicious samples but requires much time and effort [2]. Meanwhile, the enormous malware variations also brought great challenges to signature-based detection methods.

In recent years, many new malware detection methods have been proposed, ranging from multi-signature methods to static analysis, dynamic detection and heuristic detection. However, the anti-detection technology also constantly improved. Malware has learned to change their feature through object-code obfuscation, code refactoring and etc. According to the report of Symantec and McAfee, approximately 69 new instances of malware are generated per minute, and more than 50% of them are variants of existing malware [3]. The traditional feature extraction method cannot afford the huge cost brought by manually marking new samples. These new variants usually have the same malicious intentions and characteristics as the original malware [4,5]. Such a group of malware samples with similar attacking patterns is called a malware family. Recognition of malware families relies on quickly analyzing the behaviors and functions of malware.

In face of the new challenges, some researchers try to explore malware features using machine learning technologies [4–6]. Nataraj proposed to categorize the malware family by visualizing malware [7]. The method not only shows the visual similarity between different samples in the same family but also adapts to the common fuzzy coding techniques. Accordingly, neural networks are also utilized to analyze visual malware and achieved promising results [8,9]. However, due to the complexity of the neural network, the huge cost of the training process makes it hard to catch up with the rapid growth of malware variants [10–12]. Besides, most of the active neural networks need supervised learning, requiring human experts and special tools to abstract the malware features and labels of new samples, which is an extremely expensive and inefficient process [13].

Regarding these problems, we proposed a malware detection model based on malware visualization and collaborative learning. In the model, firstly the malware binaries are visualized to gray graphs, and then the malware family features are automatically extracted from the graphs using image feature extractors, such as LBP extractor. Finally, the features are sent to the cooperative learning models with multiple classifiers to recognize malware. Furthermore, noise learning theory is incorporated into the training process to exclude noise in the unlabeled samples to ensure that the model's error classification rate could be continuously reduced.

The contributions in this paper are summarized as follows:

- We proposed a malware classification framework, which integrated malware visualization, automatic feature extraction, and collaborative learning. The framework directly processes malware binaries without disassembly, decompression, and decryption.
- This framework continuously improved the classification ability of the model through the introduction of unlabeled samples， which solved the issue of lack of labeled malicious samples in actual scenes.
- Comparative experiments were conducted on two widely studied imbalanced benchmark datasets, Malimg and Microsoft. Experimental results show that the proposed framework can achieve excellent classification performance, with the accuracy of 0.98 and 0.94, respectively. Compared with the state-of-the-art methods, our method is more resistant to the effects of data imbalance.

2. Related work

2.1. Development of malware detection

With the development of machine learning and visualization technology, researchers have begun to draw on the visualization ideas in the field of computer forensics and network monitoring to visualize and classify malware [7]. First, the raw binary data in the malware is converted to grayscale images, in which each byte is represented as a grayscale pixel of the image. These byte sequences are then combined into a 2D array, and feature extraction is performed through texture analysis to convert the malware detection into an image classification task. In such a way, not only the characteristic information of software can be visualized, but also the detection efficiency of malware is improved compared with the traditional methods [14–16]. Furthermore, in contrast with traditional static analysis methods, malware visualization is more suitable for malicious samples adapted by obfuscation technology [17].

Recently, quantities of visual detection methods based on malware have been proposed [18,19]. However, these methods still have some shortages, including the lack of available labeled samples in real applications, the great gap between feature extraction algorithms and visual feedback, and etc. [20] Therefore, we apply a semi-supervised learning algorithm to alleviate the issue of insufficient samples and continuously improves the classification performance through the utilization of unlabeled samples along with noise learning theory.

2.2. Semi-supervised learning algorithm

Supervised algorithms have achieved promising performance of malware detection, however, they rely on plenty of labeled samples for training, which is difficult to be satisfied in real applications [20,21]. On the other hand, unsupervised learning can employ unlabeled samples for training but often gets lower accuracy [22]. Compared with these two types of classification algorithms, the semi-supervised learning algorithm only needs a small number of labeled samples in the training stage and can continuously enhance detection performance through the use of a large number of unlabeled samples [23,24]. Consequently, semi-supervised learning is more suitable for the applications of malware detection.

The semi-supervised learning algorithm co-training [25] assumes that we have two redundant and independent feature views to deal with data. And then, at the initial training stage, some labeled samples are summited to two basic classifiers in different feature views. After initial training, unlabeled samples with high label confidence are selected and these "pseudo-label" samples are put into the updating set for further training. Through the process of "learning from each other and making progress together", the classifiers iteratively updating in each training round until their performances are stable. However, the conditional independence of the two feature views is difficult to satisfy. S. Goldman and Y. Zhou proposed to improve the classifiers by collaborative learning [26]. Although this method has removed the requirement of redundant feature views, it still restricts the types of base classifiers, and the repeated ten-fold cross-validation in the updating process results in an overwhelming cost. In response to this problem, Zhi-Hua Zhou et al. proposed a Tri-training algorithm that neither requires sufficient redundant views nor restricts the type of classifiers [27]. The algorithm easily handles the problem of labeling confidence estimation and predictive classification of unknown samples by using three collaborative classifiers.

In this paper, the idea of noise learning theory [28] is utilized to improve the collaborative learning based on the original Tri-training algorithm. In each iteration stage, parts of "pseudo-label" samples are extracted for error rate calculations and threshold evaluation to reduce the tag error rate of "pseudo-label". For a detailed description of the model, see Section 3.

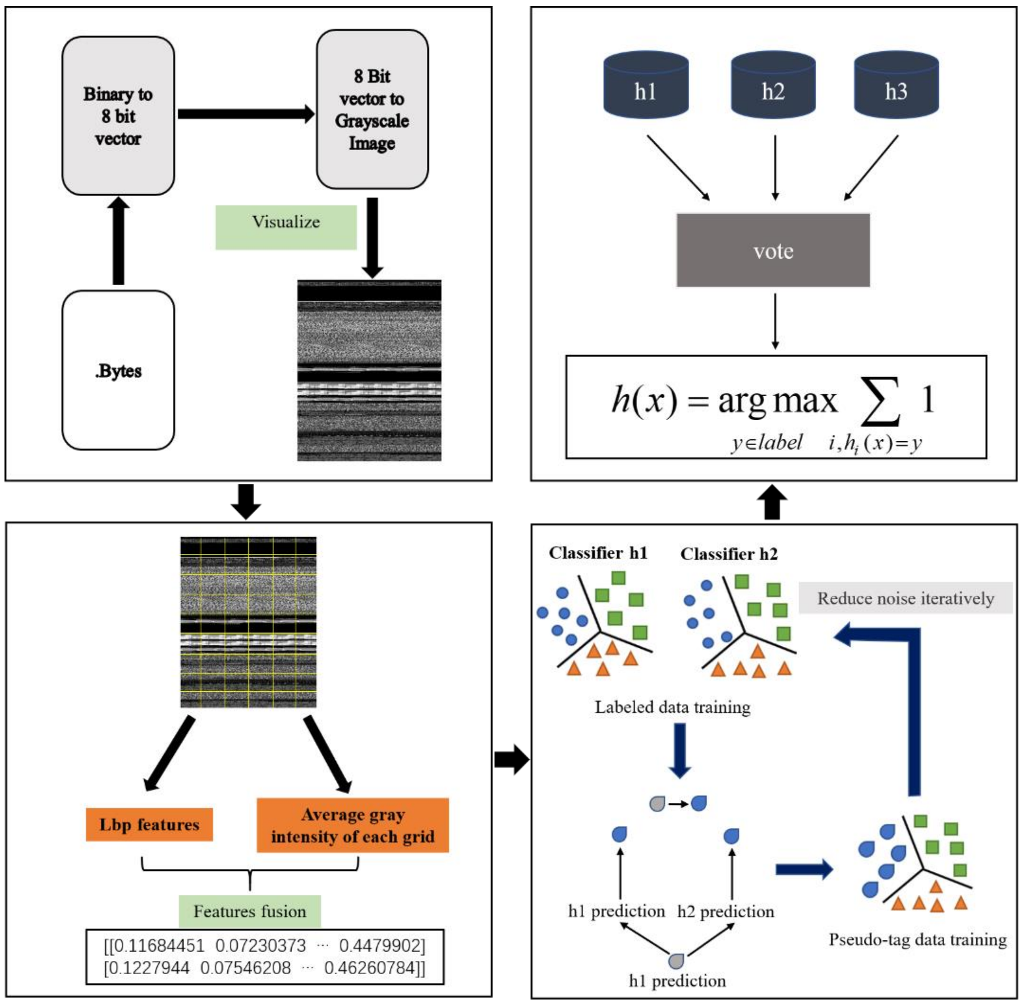

Figure 1. The overall process of malware detection for collaborative learning.

3. Methodology

The work of this paper focuses on the detection of malware. Figure 1 shows the architecture of our malware classification model, which mainly consists of three major components: malware visualization, feature extraction, and the tri-training classification algorithm.

3.1. Malware visualization

As shown in the first stage in Figure 1, The malware visualization transforms the binary codes into images with certain characteristic information [7,14]. For a given binary file, each 8-bit is transformed into one unsigned integer, and the result of these variables is reorganized into a two-dimensional matrix. The corresponding value in the matrix can be expressed as the gray value of the generated image in the range of [0, 255], where 0 and 255 represent black and white respectively. Figure 2 shows the visualization process of a binary malware file into a grayscale image.

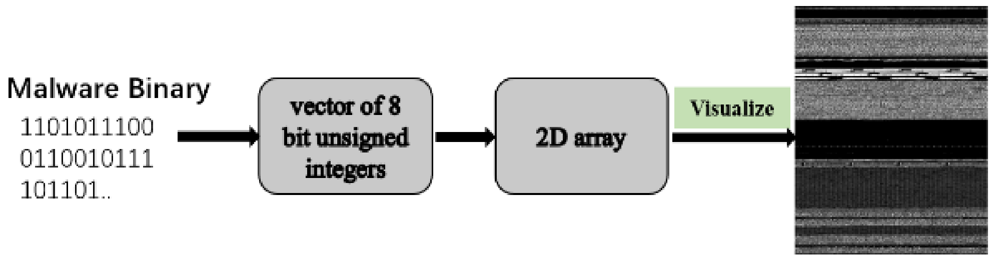

Figure 2. Process of malware visualization.

The images converted from malware have the same width and different heights. The widths of images of different sizes are required to be pondered carefully, in case the generated images are extremely high or too wide, which degrades the performance of feature extraction [7]. Table 1 gives some recommended image widths for the malicious samples.

Table 1. Image width for various sizes.

File size	Optional width
<10KB	32
10KB–30KB	64
…	…
1000KB–2000KB	1024
2000KB–4000KB	1280
4000KB–8000KB	1536
8000KB–10MB	1792
10MB–15MB	2048
15MB–20MB	2560
20MB–25MB	3072
25MB–30MB	4096
>30MB	5120

3.2. Feature extraction

Take the obfuscated code technologies into consideration, such as fragment encryption and instruction substitution in consideration, the texture features of the generated gray-scale image may be disturbed and deformed. Hence, to ensure that the image features utilized in the training are stable and robust, both the local texture features and global features of the images are extracted and fused through the canonical correlation analysis (CCA) method [29].

3.2.1. Feature selection

Before feature extraction, the SIFT feature [30], HOG feature [31], and LBP feature [32] are analyzed and compared for the local feature extraction of malicious samples' gray images [33]. SIFT features can keep the rotation, brightness, and scale of the image unchanged, but demand enormous computation. Since the characteristic gamma correction is performed at the end of the image grayscale calculation, the feature extracted by HOG can reduce the negative impact of local shadow and light changes. Nevertheless, the HOG extraction has some defects, such as the prolix generation process and sensitivity to noise points, which leads to high-risk costs. Thus, a more steady and efficient feature extraction method is required.

Many malicious samples are derived from some classic malware. Source from the structure similarity of the malware family, the pixels of the converted gray images have some equal proportions on the whole or in some local areas [7,20]. Similarly, for the LBP feature extraction, the relative size of the central pixel and the overall gray level of the neighborhood remain unchanged even if they change simultaneously [34,35]. Furthermore, The LBP descriptor has benign adaptability to the effect of image rotation. Depending on the nature of this event, the LBP method has high adaptability and robustness for the detection of different groups of malicious samples.

Therefore, LBP is utilized to extract the local features. It employs a descriptor window as a circular area, for a neighborhood with p pixels containing a one-pixel set$\{g_0, g_1, \dots, g_{p-1}\}$, and the encoding method of LBP is as:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)\, 2^p, s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{1}$$

where g_c and g_p represent the gray value of the central pixel and circular neighborhood pixel respectively, and R is the radius of the neighborhood. For such a circular LBP operator, the relative position of the center pixel$g_c$and g_p changes with the image rotation, resulting in various LBP values. Consequently, we adopt the uniform rotation invariant LBP operator, which can adapt to image rotation and has anti-noise property for a large number of modes generated in the circular neighborhood of different sizes, defined by:

$$LBP_{P,R}^{riu^2} = \begin{cases} \sum_{p=0}^{p-1} s(g_p - g_c) & if\ U(LBP_{P,R}) \leq 2 \\ P+1 & otherwise \end{cases} \tag{2}$$

where $s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$ and U represents the LBP uniformity space conversion times (the transition between binary bits 0/1), expressed as:

$$U(LBP_{P,R}) = \sum_{p=1}^{P} |s(g_p - g_c) - s(g_{p-1} - g_c)| \tag{3}$$

The LBP feature has the advantage of stability and noise anti-interference for the regional feature description, which mainly focuses on the local feature while lacking the global feature description of the images. Therefore, after extracting the LBP features of the malicious sample image, the global feature of the image was also extracted, this reduced the noise through Gaussian fuzzy processing. The image is divided into grids by the average size of 16 × 16. And the global image feature computes the mean value and average variance of pixel intensity in each grid of image where the two-dimensional Gaussian function is employed to calculate the weight of each pixel shown in (4).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{4}$$

where σ is the variance of x.

3.2.2. Feature fusion

To enhance the correlation between features and images, canonical correlation analysis (CCA) is applied to the fusion of image features after extracting the global and local features. CCA is an effective multi-data processing method [36], which can mine the potential association relationship between two sets of variables to obtain more representative data. The general idea of CCA is to obtain the maximum correlation coefficient between the linear combination of two groups of variables through a large number of matrix calculations, to establish the relationship between the two groups of variables [37]. The CCA feature fusion produces a structure containing D null hypothesis (H0) and related information after the fusion of the two features X(local) and Y(global), which D takes the minimum values of the ranks of the two feature matrices X and Y: $D = min(rank(X), rank(Y))$. Relevant information statistics of fusion are shown in Table 2.

Table 2. Statistics on CCA fusion methods.

Wilks	**chisq**	**pChisq**	**F**	**pF**	**df1**	**df2**
Wilks Lambda	Approximate chi-squared statistic for H0	The right-tail significance level for chisq	Approximate chi-squared statistic for H0_K	The right-tail significance level for F	The numerator degrees of freedom for F	The denominator degrees of freedom for F

3.3. *Tri-Training algorithm based on Noise learning judgment*

The Tri-training algorithm contains three basic classifiers: C_1, C_2, C_3. $C_i (i \in \{1,2,3\})$. A small number of labeled samples L along with a large number of unlabeled samples U are applied to the training process to carry out co-training [25] of the basic classifiers.

For the three classifiers C_i, C_j, C_k (where $i, j, k \in \{1, 2, 3\}$ and $j, k \neq i$) in each round of co-training, C_j and C_k are used to randomly select samples from the unlabeled sample set U to make predictions, where the samples $\{x | predict_k(x) = predict_j(x)\}$ labeled by C_j and C_k are considered to have higher labeling confidence, and then they are taken as new labeled data and put into the updating set L_i of the classifier C_i. L_i along with the known labeled sample set L to update the classifier C_i. After each round of updating, the newly labeled samples are put back in U to start a new round of iteratively training until all the classifiers do not change. As shown in formula (5), after the iterative training, the class of the sample is predicted through a voting mechanism.

$$h(x) = \underset{y \in label}{argmax} \sum_{i, h_i(x) = y} 1 \tag{5}$$

If the prediction of C_j and C_k on x is correct, C_i will get a valid new example for further training; otherwise, C_i will get a noise example with an incorrect label. According to the noise learning [28] theory of Angluin and Laird, for a training sequence γ_i containing m samples, if the sample size m is:

$$m \geq \frac{2}{\varepsilon^2(1-2\eta)^2} ln(\frac{2N}{\delta}) \tag{6}$$

where ε is the upper limit of the classifier error and η is the classification error of the training set rate, N is the number of categories, δ is the confidence parameter, then the PAC (probably approximately correct identification) judgment for the real classification h* is:

$$P_r[d(h_j, h^*) \geq \varepsilon] \leq \delta \tag{7}$$

where $d(h_j, h^*)$ represents the sum of the probabilities of the elements of the symmetric difference between the hypothesis (classifier) h_j and the real situation h^*. Given the confidence parameter δ and the upper limit of classification error ε, formula (8) can be transformed into formula (9):

$$(1-2\eta)^2 > \frac{2}{\varepsilon^2 m} ln(\frac{2N}{\delta}) \tag{8}$$

Expand the left side of formula (8), we can get:

$$1 - \frac{2}{\varepsilon^2 m} ln(\frac{2N}{\delta}) \geq 4(\eta - \eta^2) \geq 4\eta \tag{9}$$

Thus

$$1 - \frac{2}{\varepsilon^2 m} ln(\frac{2N}{\delta}) \geq 4\eta \tag{10}$$

That is, for the given confidence parameter δ and the upper limit of classification error ε, We need to guarantee that:

$$\eta \leq \frac{1-\frac{2}{\varepsilon^2 m} ln(\frac{2N}{\delta})}{4} \tag{11}$$

To simplify the calculation of equation (10), let$c = 2\mu ln(\frac{2N}{\delta})$, where $\mu$the coefficient that makes equation (11) is equal, and then we can get:

$$\frac{c}{\varepsilon^2} = m(1-4\eta) \tag{12}$$

It can be seen from formula (12) that the square term of the upper limit of error ε is inversely proportional to $m(1-4\eta)$. Samples that are temporarily marked by two of the classifiers in each round can be called pseudo-marked samples. Since the number of unlabeled examples selected for each round of tri-training is not fixed, let L^t be the pseudo-sample set labeled C_1 for the t^{th} round, and the error rate of sample detection is η_t, then for the sample size of this round $M_t = |L| + |L^t|$. Compared with the previous round, if the training result of this round is improved, it is necessary to ensure that the (13) must be satisfied:

$$(|L| + |L_u^t|)(1-4\eta^t) > (|L| + |L_u^{t-1}|)(1-4\eta^{t-1}) \tag{13}$$

That is, updating the classifier by the $|L^t|$ unknown samples introduced in this round can continue to improve the classification performance, because the upper limit error $\varepsilon$of the training result of the t^{th} round is lower than that of the$t-1^{th}$ round; otherwise, the newly-labeled samples of this round are abandoned and start re-sample training in U. For equation (13), the sample detection error rate is $\eta^t = \frac{e_L^t|L|+e_u^t|L_u^t|}{|L+L_u^t|}$, where e_L^t and e_u^t represent the error rate of the classifier on the labeled sample set L and the unlabeled sample set L_u during the t^{th} round of training respectively. To ensure that the training process can continuously reduce the upper limit of classification error ε, substituting the η_t expression into equation (13), we can get:

$$|L_u^t| - 4(e_L^t|L| + e_u^t|L_u^t|) > |L_u^{t-1}| - 4(e_L^{t-1}|L| + e_u^{t-1}|L_u^{t-1}|) \tag{14}$$

In most cases, for the classification error rate of the model on the labeled samples, $e_L^t << e_u^t$, thus e_L^t can be ignored. So equation (14) can be simplified as (15).

$$|L_u^t| - (1 - 4e_u^t) > |L_u^{t-1}|(1 - 4e_u^{t-1}), \text{ that is}$$

$$\frac{|L_u^t|}{|L_u^{t-1}|} > \frac{1-4e_u^{t-1}}{1-4e_u^t} \tag{15}$$

Therefore, in the tri-training process of the classifiers $C_i, C_j, C_k (i,j,k \in \{1,2,3\},\ and\ j,k \neq i)$, for the two adjacent rounds of training, when the size and error rate of the new labeled samples meet equation (15), the new labeled samples with the same labels from C_j and C_k are submitted to C_i for updating; otherwise, the newly labeled samples in this round are discarded, and unlabeled samples are re-selected from U for training. For the classification error rate e_u^t of the "pseudo-labeled" samples that cannot be directly calculated, this paper adopts the idea of ten-fold cross-validation. In each round of the iterative training, $\frac{1}{10}$ of the labeled samples are randomly selected from L as the test set to estimatee_u^t, the remaining samples in L are combined with U for trained together. The Tri-training training with noise judgment is listed in Table 3.

4. Experiment

4.1. Dataset

To evaluate the performance of the proposed method, we carried a group of comparisons based on the Malimg data set [7] and the Microsoft Malware Classification Challenge data set [38] (referred to as the Microsoft data set). The description of The Microsoft and Malimg datasets are shown in Table 4 and Table 5, respectively.

Tables 4 and 5 show that the distribution proportions of malicious samples in the Malimg dataset and the Microsoft dataset are imbalanced, and the proportions of different sample groups are different. For example, in the Malimg data set, these two sample families of ALLAPLE in the 25 sample families accounted for 48.6% of the total, while the remaining 23 samples only accounted for 51.4%; in the Microsoft data set, the Simda samples in the 9 sample families only accounted for 0.4% of the total. For traditional supervised learning algorithms, the imbalanced distribution of malicious samples often leads to overfitting and poor classification performance.

Table 3. Improved Tri-training algorithm based on noise learning theory.

```
Algorithm 1: Periodic tri-training with noise judgment
Input:  $\delta$: Confidence   $\sigma$: Error rate threshold   $N$: Class Number
        $\varepsilon_e$: Upper limit of expected error   $L$: Labeled sample set
        $U$: Unlabeled sample set   $Learn$: Learning algorithm
        $Vote$: Vote algorithm   $Random$: Random sampling algorithm
        $e_u^i$: Error rate of round i
Output: $\underset{y\in label}{\text{argmax}} \sum_{i:c_i(x)=y} 1$
1 Initialize $\delta \leftarrow 0.95$,  $\sigma \leftarrow 0.1$,  $N \leftarrow 8$ ,  $\varepsilon_e \leftarrow 0.5$,  $e_u^0 \leftarrow 1$;
2 while $\eta > \varepsilon_e$ do
3     $T \leftarrow Random(0.1L)$,  $L' \leftarrow L - T$,    $e_u^0 \leftarrow 1$;
4     for $i \in [1,...,3]$ do
5         $c_i = Learn(c_i, L')$
6         $update_i \leftarrow FALSE$
7     endfor
8     while $i<3$ do
9         for $every\ x \in U$ do
10            if $c_j(x)\ is\ equal\ to\ c_k(x)$ then
11                $L_i \leftarrow L_i \cup \{(x, c_j(x))\}$
12            end
13        endfor
14        $c_i = Learn(c_i, L)$
15        $\eta^t \leftarrow 1 - accurency(c_i, \text{L}, \text{T})$
16        if $\eta^t > \frac{1}{4} - \frac{1}{2\varepsilon_e^2 m} \ln(\frac{2N}{\delta})$ then
17            continue
18        end
19        $c_i = \text{Learn}(c_i, L_i)$,   $e_u^{t-1} \leftarrow e_u^t$,   $e_u^t \leftarrow$ 1-accurency$(c_i, L_i, \text{T})$
20        if $\frac{L_u^t}{L_u^t} <= \frac{1-4e_u^{t-1}}{1-4e_u^t}$ then
21            continue
22        end
23        if $|e_u^t - e_u^{t-1}| > \sigma$ then
24            continue
25        end
26        $c_i = Learn(c_i, (L' \cup L_i))$
27        $update_i \leftarrow TRUE$,  $i \leftarrow i+1$
28    end
29    $\eta = Vote(c_1, c_2, c_3, L_1, L_2, L_3)$
30 end
```

Table 4. Sample distribution of malware for the Malimg dataset.

Class ID	Class	Family	Sample size	Population
1	Adialer.C	Dialer	122	1.3%
2	Agent.FYI	Backdoor	116	1.2%
3	Allaple.A	Worm	2949	31.6%
4	Allaple.L	Worm	1591	17.0%
5	Alueron.gen!J	Trojan	198	2.1%
6	Autorun.K	Worm: AutoIT	106	1.1%
7	C2Lop.gen!G	Trojan	200	2.1%
8	C2Lop.P	Trojan	146	1.6%
9	Dialplatform.B	Dialer	177	1.9%
10	Dontovo.A	Trojan downloader	162	1.7%
11	Fakerean	Rogue	381	4.1%
12	Instantaccess	Dialer	431	4.6%
13	Lolyda.AA 1	PWS	213	2.3%
14	Lolyda.AA 2	PWS	184	2.0%
15	Lolyda.AA 3	PWS	123	1.3%
16	Lolyda.AT	PWS	159	1.7%
17	Malex.gen!J	Trojan	136	1.5%
18	Obfuscator.AD	Trojan downloader	142	1.5%
19	Rbot!gen	Backdoor	159	1.7%
20	Skintrim.N	Trojan	80	0.9%
21	Swizzor.gen!E	Trojan downloader	128	1.4%
22	Swizzor.gen!I	Trojan downloader	132	1.4%
23	VB.AT	Worm	408	4.4%
24	Wintrim.BX	Trojan downloader	97	1.0%
25	Yuner.A	Worm	800	0.086

Table 5. Sample distribution of malware for the Microsoft dataset.

Class ID	Class	Sample size	Population
1	Ramnit	1541	14.2%
2	Lollipop	2478	22.8%
3	Kelihos ver3	2942	27.1%
4	Vundo	475	4.4%
5	Simda	42	0.4%
6	Tracur	751	6.9%
7	Kelihos ver1	398	3.7%
8	Obfuscator.ACY	1228	11.3%
9	Gatak	1013	9.3%

4.2. Feature fusion methods analysis

Firstly, the LBP feature and the average grid gray intensity of the malicious sample images in the

Malimg data set are extracted. And then CCA fusion is carried out for these two features. Finally, the fusion results are submitted to a co-learning model for the training of classifiers. The results of the classifications are shown in Figure 3. It can be seen that as the proportion of training data increases, the classification results on CCA are better， which is attributed to the suppression of the repellency between the two variables by the CCA method. By calculating the correlation coefficient of the two one-dimensional data obtained by linear transformation projection, the CCA method maximizes the correlation between the two dimensions, thus obtaining more discriminative data characteristics. Moreover, after CCA fusion the dimensionality of the data is reduced, which greatly saved the training cost of co-learning. The time comparison between traditional serial fusion and CCA fusion in different sample sizes (in seconds) is shown in Table 6. And from the table, we can see that, after CCA fusion, the time cost is dramatically reduced.

Table 6. Time cost for fusion methods.

Method / **Sample size**	**Serial Fusion**	**CCA Fusion**
4000	378.15s	54.04s
5000	589.81s	106.16s
6000	846.73s	319.62s

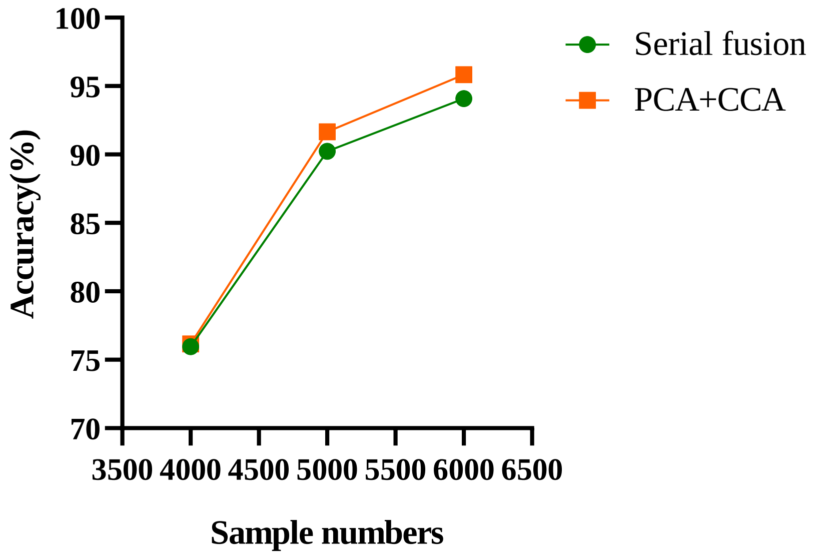

Figure 3. Accuracy for fusion methods.

4.3. Algorithm performance analysis

5 alternative linear classifiers are employed for collaborative learning. To ensure the degree of divergence among the cooperative learning classifiers, we eventually selected random forest, KNN, and LR as the three candidate classifiers C_1,C_2 and $C_3$for tri-training. In the experiment, the three classifiers are initialized with differentiated sample characteristics at first. Then the three classifiers are trained by a cooperative learning algorithm periodically. After the training, different

types of malicious samples are predicted on the test set and compared with traditional machine learning classifiers.

We have conducted an experimental effect analysis of the collaborative learning algorithm on the Malimg data set, as shown in Figure 4. In the experiment, C_1,C_2 and C_3 respectively represent the selected single classical classifier, and T represents the cooperative learning algorithm satisfying the noise learning theory. As shown in Figure 4, due to the low sample size at the beginning, the classification performance of all training algorithms is poor. With the continuous learning from unlabeled samples, the classifier is continuously improved, while the classification performance of collaborative learning has raised more obviously; when the number of unlabeled samples reaches 3000, the accuracy of fusion classification can exceed 95%, and the accuracy is raised until the number reaches 3400. The results demonstrated that the collaborative learning algorithm can continuously improve classification accuracy through a continuous utilization of unlabeled samples.

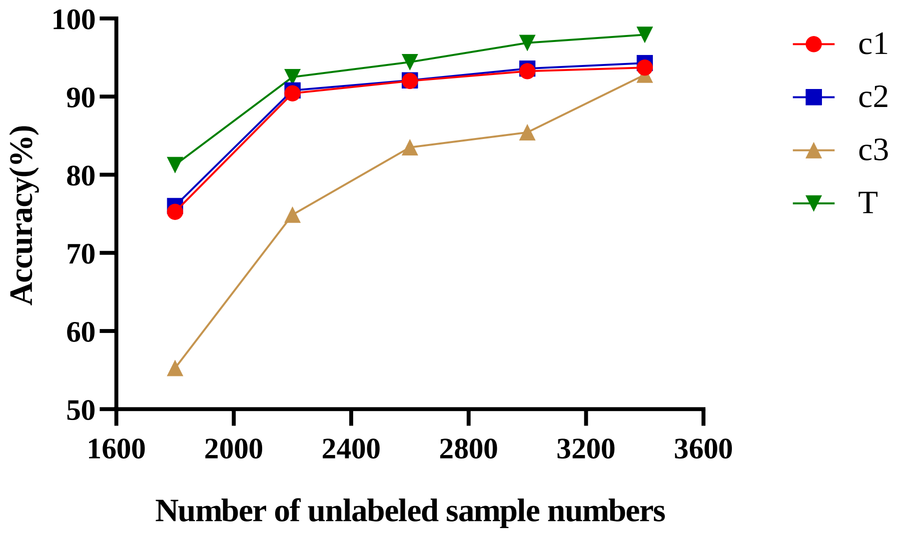

Figure 4. Accuracy of each algorithm.

However, for such a data set with imbalanced samples, simply calculating the correct rate of the model cannot comprehensively reflect the advantages of the proposed model. Consequently, in addition to calculating the accuracy of the overall model, the Precision, Recall, and F1- score of the model are also calculated to further evaluate our model. The P (precision), R (recall) and F (F1-score) of each given class are calculated first, and then average the F1 scores of all classes to calculate the weighted-average F1 score. The calculation formulas of P, R and F are shown in Equations (16)–(19).

$$P_i = \frac{TP_i}{FP_i} \tag{16}$$

$$R_i = \frac{TP_i}{FN_i} \tag{17}$$

$$F_i = \frac{2\times P_i \times R_i}{P_i + R_i} \tag{18}$$

and

$$weighted - averaged\ F1 = \frac{1}{n}\sum_{i=1}^{n} F_i \tag{19}$$

where TP_i is the number of samples that are correctly classified in the i^{th} category, FP_i is the number of samples that are misclassified into the i^{th} category, and FN_i indicates how many samples belonging to the i^{th} class are misclassified to other classes.

The evaluation results based on Malimg and Microsoft are shown in Table 7 and Table 8. The Malimg dataset has an F1-score of up to 97.23%, while the accuracy rate of the Microsoft dataset can reach 94.09%. Therefore, compared with other classic classification algorithms, our method can not only reduce the input cost of labeled samples but also has better detection accuracy.

Table 7. Classification representation of different approaches on the Malimg dataset (%).

Approach	Accuracy	Precision	Recall	F1-Score
RandomForest	95.06	99.48	93.00	96.16
SVM	93.38	1	95.37	96.57
KNN	93.31	99.87	93.31	96.48
LogisticRegression	92.69	98.39	92.68	95.43
GBDT	93.37	1	93.57	96.57
Our method	97.95	99.95	95.06	97.23

Table 8. Classification representation of different approaches on the Microsoft dataset (%).

Approach	Accuracy	Precision	Recall	F1-Score
RandomForest	92.72	99.69	92.74	96.08
SVM	93.93	99.69	93.93	96.73
KNN	93.78	99.37	92.66	96.50
LogisticRegression	93.63	99.06	93.45	96.27
GBDT	92.42	96.19	92.42	94.24
Our method	94.09	1	94.09	96.96

5. Conclusions and future work

We propose a new malware classification model based on malware visualization, and co-training of classifiers, and shows that combining the malware visual method with tri-training can provide a better discriminative pattern of malware families. In this framework, the malware is transformed into grayscale images by visual methods, then a fusion method based on CCA is utilized to fuse the local and global features extracted from the gray image to reduce time cost and improve feature relevance; finally, three basic classifiers are collaboratively training based on the tri-training schemes. In each round of collaborative learning, the new labeled samples are filtered by noise learning theory which ensures a continuous improvement of the overall performance of the co-learning results and alleviates the problem that the labeled samples are difficult to obtain in practical applications through the incorporate of unlabeled data into the training process.

The advantages of our method are manifold. Firstly, the experimental results show that the proposed method achieves good classification performances of 0.98 and 0.94 on Malimg dataset and Microsoft dataset, respectively. Second, our approach is more resistant to data imbalances. Thirdly, the tri-training algorithm improves the classification ability of the model through the introduction of a large number of cheap unlabeled samples and reduces the noise impact caused by the lack of labeled samples. Although the accuracy of the collaborative learning algorithm is improved after iterative training, it increases the time overhead. In future work, the iterative updating efficiency of the collaborative learning algorithm needs to be further improved, such as introduce some more complex models which are more suitable for image classification.

Acknowledgments

We would like to thank you for following the instructions above very closely in advance. It will definitely save us lot of time and expedite the process of your paper's publication.

Conflict of interest

The authors have no conflict of interest.

References

1. A. P. Namanya, A. Cullen, I. U. Awan, J. P. Disso, The world of Malware: An overview, in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2018, pp. 420–427, doi: 10.1109/FiCloud.2018.00067.
2. A. Martín, H. D. Menéndez, D. Camacho, MOCDroid: Multi-objective evolutionary classifier for Android malware detection, *Soft. Comput.*, **21** (2017), 7405–7415.
3. P. Foran, Of digital reliance, risk and resilience, *Progres. Railro.*, **62** (2019), 30–30,32–33.
4. K. Rieck, P. Trinius, C. Willems, T. Holz, Automatic analysis of malware behavior using machine learning, *J. Comput. Secur.*, **19** (2011), 639–668.
5. F. Touchette, The evolution of malware, *Network Security*, **1 (**2016), 11–14.
6. D. Gavrilut, M. Cimpoesu, A. Dan, L. Ciortuz, Malware detection using machine learning, *Int. Multiconfer. Comput. Sci. Inform. Tech.*, 2010.
7. L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, Malware images: Visualization and automatic classification, *Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec)*, 2011, Available from: https://doi.org/10.1145/2016904.2016908.
8. W. Huang, J. W. Stokes, MtNet: A multi-task neural network for dynamic Malware classification, *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016.
9. J. Saxe, K. Berlin, Deep neural network based Malware detection using two dimensional binary program features, *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, **(**2015), 11–20.
10. E. David, N. S. Netanyahu, DeepSign: Deep learning for automatic malware signature generation and classification, *Internat. Joint Confer. Neural Networks*, (2015), 1–8.

11. M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, Novel feature extraction, selection and fusion for effective Malware family classification, *6th ACM Conference on Data and Applications Security and Privacy (CODASPY)*, 2016.
12. Z. A. Genç, G. Lenzini, P. Y. A. Ryan, *Next Generation Cryptographic Ransomware*, (2018), 385–401.
13. J. Sahs, L. Khan, A machine learning approach to Android Malware detection, *Intelligence and Security Informatics Conference (EISIC)*, (2012), 141–147.
14. X. G. Han, W. Qu, X. X. Yao, C. Y. Guo, F. Zhou, Research on malicious code variants detection based on texture fingerprint, *J. Commun.*, 2014.
15. K. S. Han, J. H. Lim, B. Kang, E. G. Im, Malware analysis using visualized images and entropy graphs, *Int. J. Inf. Secur.*, **14** (2015), 1–14.
16. J. Y. Kim, S. J. Bu, S. B. Cho, Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders, *Inform. Ences,* (2018), 83–102.
17. S. Shang, N. Zheng, X. Jian, M. Xu, H. P. Zhang, Detecting malware variants via function-call graph similarity, *International Conference on Malicious & Unwanted Software*, (2010), 113–120.
18. B. Anderson, C. Storlie, T. Lane, Improving malware classification: Bridging the static/dynamic gap, *ACM Workshop on Security & Artificial Intelligence*, 2012.
19. P. Zhang, B. Sun, R Ma, A Li, A novel visualization Malware detection method based on Spp-Net, *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, (2019), 510–514.
20. G. Xiao, J. Li, Y. Chen, K. Li, MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks, *J. Parall. Distribut. Comput.*, **141** (2020), 49–58.
21. J. E. Engelen, H. H. Hoos, A survey on semi-supervised learning, *Mach. Learn.*, **109** (2020), 373–440.
22. K. Nigam, A. Mccallum, T. Mitchell, Semi-supervised text classification Using EM, *MIT Press*, (2006), 33–55.
23. F. D. Frumosu, M. Kulahci, Outliers detection using an iterative strategy for semi-supervised learning, *Quality Reliab. Eng.*, **35** (2019).
24. Z. H. Zhou, M. Li, Tri-training: Exploiting unlabeled data using three classifiers, *IEEE Trans. Knowl. Data Eng.*, **17** (2005), 1529–1541.
25. A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, *Proceed Conference Computer Learning*, (1998), 92–100.
26. Y. Zhou, S. Goldman, Democratic co-learning, *Proc. 16th IEEE Int. Conf. Tools Artif. Intell.,* (2004), 594–602.
27. D. G. Kong, G. H. Yan, Discriminant malware distance learning on structural information for automated malware classification, *Performance Evaluation Review*, **41** (2013), 347–348.
28. D. Angluin, P. Laird, Learning from noisy examples, *Mach. Learn.,* **4** (1988), 343–370, Available from: https://doi.org/10.1007/BF00116829
29. M. J. Sullivan, Distribution of edaphic diatoms in a mississippi salt marsh: A canonical correlation analysis, *J. Phycol.*, (1982), 130–133.
30. D. G. Lowe, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vision.*, **60** (2004), 91–110.

31. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, *IEEE Computer Society Conference on Computer Vision & Pattern Recognition (CVPR)*, 2005.
32. T. Ojala, M. Pietikäinen, T. Mäenpää, Gray scale and rotation invariant texture classification with local binary patterns, *European Conference on Computer Vision (ECCV)*, (2000), 404–420.
33. J. Ma, X. Jiang, A. Fan, J. Jiang, J. Yan, Image matching from Handcrafted to deep features: A survey, *Int. J. Comput. Vision*, **1** (2020), 1–57.
34. T. Ahonen, J. Matas, H. Chu, M. Pietikäinen, Rotation invariant image description with local binary pattern histogram fourier features, *Image Analys.*, (2009), 61–70.
35. H. Ran, W. Qi, Z. Guo, Feature reduction of multi-scale LBP for texture classification, *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, (2015), 397–400.
36. I. H. Witten, E. Frank, Data mining: Practical machine learning tools and techniques, Second Edition, *ACM Sigmod. Record.*, **31** (2005), 76–77.
37. M. Haghighat, M. Abdel-Mottaleb, W. Alhalabi, Fully automatic face normalization and single sample face recognition in unconstrained environments, *Expert Syst Appl.,* **47**(2016), 23–34.
38. R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, M. Ahmadi, Microsoft Malware Classification Challenge, *CORR*, 2018.