Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

Image-based malware classification using section distribution information

Mao Xiao, Chun Guo, Guowei Shen, Yunhe Cui, Chaohui Jiang*

Guizhou Provincial Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

ARTICLE INFO

Article history:
Received 12 July 2020
Revised 26 January 2021
Accepted 26 July 2021
Available online 3 August 2021

Keywords:
Malware classification
Malware visualization
Gray images
Machine learning
Deep learning

ABSTRACT

Recently, with the rapid increase in the number of malware, the traditional machine learning-based malware classification methods are faced with the severe challenge of efficiently and accurately detecting a large number of malicious programs. To meet this challenge, malware classification based on malware image and deep learning has become an effective solution. However, it is difficult to identify the section distribution information such as the number, order, and size of sections from the current gray images converted by the binary sequences of PE files. Therefore, this article proposes a novel visualization method that introduces the Colored Label boxes (CoLab) to mark the sections of a PE file to further emphasize the section distribution information in the converted malware image. Moreover, a malware classification method called MalCVS (Malware classification using CoLab image, VGG16, and Support vector machine) is constructed. The experimental results of the malware collected from VX-Heaven and Virusshare as well as the Microsoft Malware Classification Challenge dataset showed that MalCVS can effectively classify malware into families with high accuracy. The average accuracies of MalCVS are respectively 96.59% and 98.94% on the two datasets.

1. Introduction

Malware is a program that is developed to damage computer systems (Alazab, 2015) and is divided into categories such as viruses, worms, and trojans. Nowadays, because malware can change code characteristics through obfuscation technology to produce a large number of variants in a short time (Wang et al., 2011), the number of malware has shown a sharp increase in recent years. As shown in Fig. 1, the number of malware in the past decade has increased year by year and the total number of malware in 2020 has increased more than 15 times compared to 2011.

To defend against malware, malware analysis methods have emerged and developed with the development of malware. Digital signature analysis, static analysis, and dynamic analysis technologies are usually employed in today's malware analysis methods. Malware analysis based on digital signature technology is widely used in commercial products, the digital signature of software is often obtained via manual analysis and compared with the signature in the signature library to determine whether it is malware. However, along with the rapidly increasing number of malware, the number of digital signatures will show an exponential increase (Nataraj et al., 2011). Static analysis technology analyzes the binary content or disassembled codes of malware without performing it, but in most cases, static analysis is not a trivial task because obfuscation technology is widely used in today's malware (Ni et al., 2018). Dynamic analysis technology usually induces malware into a virtual environment, such as Sandbox (Jamalpur et al., 2018), to induce malware to run automatically to observe the malware's operation (Damodaran et al., 2017; Lindorfer et al., 2015). However, this technology usually needs to take a lot of time to analyze the reports generated by the

* Corresponding author.
E-mail address: gc_gzedu@163.com (C. Guo).
https://doi.org/10.1016/j.cose.2021.102420

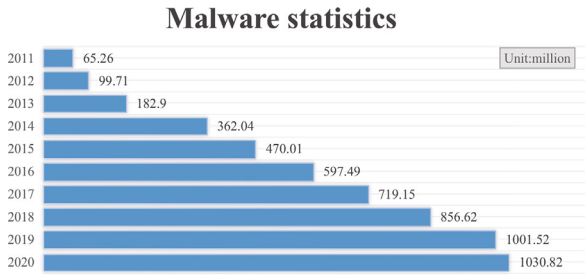


Fig. 1 – Statistics of the number of malware in the past ten years (AV-TEST, 2020).

controlled environment, and it would be invalid if malware can detect the virtual environment and change its behavior.

To cope with the constantly increasing number of malware, machine learning techniques have been used to tackle the tasks of malware detection and classification over the past decade (Gibert et al., 2020). In machine learning-based methods, malware detection/classification is a two-step process: feature extraction and classification/clustering. The performance of such methods undoubtedly depends on the extracted features and the classification/clustering techniques (Ye et al., 2017). However, traditional machine learning-based methods rely mainly on manually designed features based on expert knowledge, and the processes of feature engineering and feature extraction are time-consuming (Gibert et al., 2020). In the past few years, malware classification methods based on malware images and deep learning have received a lot of attention because they can eliminate a lot of feature engineering works (Yuan et al., 2020). The most widely used malware image in such methods is the gray image that is converted by malware's binary sequences and contains malware's binary information. This article refers to the number, order, and size of sections of a PE file as its section distribution information. Considering that the similarity of the PE files belonging to the same malware family, the section distribution information might be useful to classify malware. However, it is difficult to identify this information from the current malware images converted by the binary sequences of PE files. Therefore, in this article, we propose a novel malware visualization method that further emphasizes the section distribution information of a PE file in the generated image. Colored Label boxes are employed to mark the sections of malware in the image generated by the proposed visualization method and this article calls this image "CoLab image". The CoLab image contains the binary content and section distribution information of malware, and the generation of CoLab images does not require execution nor disassembling of any malware's code.

The main contributions of this article are as follows:

(1) A malware visualization method that further emphasizes the section distribution information of a PE file in the converted image is proposed, which is beneficial to enhance the similarity of the malware images converted by the PE files belonging to the same family.

(2) Malware classification method using CoLab images, VGG16 (Simonyan and Zisserman, 2015), and Support vector machine (MalCVS) is proposed to classify malware.

(3) Experimental results on malware samples collected from VX-Heaven (Vxheaven, 2010) and Virusshare (Virusshare, 2010) (VXV) as well as Microsoft Malware Classification Challenge (BIG-2015) (Microsoft, 2015) dataset show that MalCVS is superior to the gray image-based and opcode-based malware classification methods in terms of accuracy, recall, precision, and F1-score.

The rest of this article is organized as follows. Section 2 gives a brief introduction to the current malware classification methods. The MalCVS proposed in this article is detailed in Section 3. Section 4 presents the experimental results of the MalCVS on the VXV and BIG-2015 datasets. Finally, Section 5 summarizes the work of this article and future work.

2. Related work

Malware classification differentiates different families of malware to provide a better understanding of their capabilities (Gibert et al., 2020). Over the past decade, more and more researchers have employed machine learning to tackle the task of malware classification. Static analysis and dynamic analysis are used in traditional machine learning-based malware classification methods to extract features (Gandotra et al., 2014). In recent years, deep learning is increasingly used in the field of malware classification (Gibert et al., 2018b; Yuan et al., 2020). In the deep learning-based malware classification methods, the methods based on malware images and deep learning have received a lot of attention. Therefore, traditional machine learning-based malware classification methods and malware classification methods based on malware images and deep learning are reviewed in this section.

2.1. *Traditional machine learning-based malware classification methods*

In traditional machine learning-based malware classification methods, the features can be divided into two groups: static features and dynamic features. The extracted features are then used for building a model to classify malware into families. Correspondingly, traditional machine learning-based malware classification methods can be classified into two categories, namely, methods based on static features and methods based on dynamic features.

2.1.1. *Methods based on static features*

Static features are obtained from a piece of the program without requiring its execution. Byte sequences, opcode sequences, Application Programming Interface (API), and Function Call Graph (FCG) are commonly used as static features for malware classification. Drew et al. (2016) proposed a malware classification method based on byte sequences. Their method uses the processed byte sequences of malware as the input for the Strand (Super Threaded Reference-Free Alignment-Free Nsequence Decoder) classifier. A malware classification method based on n-gram sequences of opcodes was proposed by Zhang et al. (2019). In this method, the n-gram Term Frequency (TF) value composition vector is used as the feature vector, combined with the machine learning algorithm

to classify ransomware. Ahmadi et al. (2016) used the top 794 frequent APIs based on an analysis of near 500 K malware samples to build a multimodal system for malware classification. A malware classification method based on extracting a feature vector from FCG representation was proposed by Hassen and Chan (2017). The FCG representation of this method is converted into a feature vector based on function clustering. Besides, some researchers (Manavi and Hamzeh, 2017; Narayanan et al., 2016; Nataraj et al., 2011) used the features extracted from malware images and various traditional machine learning algorithms for malware classification. In their methods, GIST (Manavi and Hamzeh, 2017; Nataraj et al., 2011) or Principal Component Analysis (PCA) (Narayanan et al., 2016) is used to extract features from malware images generated from binary sequences or opcodes. Obtaining the opcode sequences, APIs, or FCG from a PE file involves disassembling this file. Therefore, some methods based on static features would be invalid if they fail to obtain malware's disassembly codes (Yuan et al., 2020).

2.1.2. *Methods based on dynamic features*

Dynamic features are those obtained from the execution of PE files at runtime. Even if malware employs code obfuscation techniques, its behaviors can be monitored and analyzed by dynamic analysis. Instruction traces, network traffic, and API call traces are commonly used as dynamic features for malware classification. Kyoungsoo et al. (2014) proposed a malware classification method based on image matrices. To generate the visualized image in their method, dynamic instruction traces are needed when facing malware using obfuscation or packing techniques. Malware detection and classification system based on Deep Packet Inspection (DPI) and flow packed headers was proposed by Boukhtouta et al. (2016), it executes malware in a sandbox for three minutes to generate malicious traffic. Then, 22 bidirectional flow features are extracted from traffic and provided as input to the classification algorithms of boosted J48, J48, Naive Bayes (NB), Boosted NB, and SVM. Hansen et al. (2016) proposed a malware detection and classification method based on behavioral analysis. Their method use the Cuckoo sandbox to obtain malware's API call sequences to generate feature vectors and then employe a Random Forest (RF) algorithm to detect and classify malware. A malware family classification system for 11 malicious families was proposed by San et al., 2018. This approach uses the proposed malicious feature extraction algorithm to extract API call traces from the reports of the cuckoo sandbox. To classify malware into different families, the authors use RF, KNN, and decision table in their system. In general, the process of extracting dynamic features is time-consuming because much time is needed for the induction of execution and report analysis. Besides, if the induced conditions are not met, some malware's malicious behaviors will not be observed (Nataraj et al., 2011).

2.2. *Malware classification methods based on malware images and deep learning*

Different from the methods that using malware images and traditional machine learning (Manavi and Hamzeh, 2017; Narayanan et al., 2016; Nataraj et al., 2011), malware classification methods based on malware images and deep learning convert malware into an image and then use deep learning algorithms to extract image features or classify malware. This type of method can further reduce the feature engineering cost. Based on the number of types of features or modalities of data, malware classification methods can be divided into two groups: based on one modality of data, based on multiple modalities of data (Gibert et al., 2020). According to this classification criterion, we classify the malware classification methods based on malware images and deep learning into two categories, namely, methods based on one modality of data and methods based on multiple modalities of data. Each type is individually below.

2.2.1. *Methods based on one modality of data*

Malware classification methods based on one modality of data rely on one type of feature or modality of data to classify malware. Binary sequences and opcode sequences are wildly used to convert malware into images. Binary sequences of malware can be obtained easily and efficiently since there is no need to disassemble the malware for extracting them. Kebede et al. (2017) proposed a malware classification method based on deep learning architecture. This method converts the binary sequences of malware into gray images and then employs deep learning to distinguish malware of 8 different categories. Methods (Kalash et al., 2018; Liu et al., 2020; Mourtaji et al., 2019; Rezende et al., 2017) also use the gray images converted directly from the binary sequences of malware as the input for the deep learning architecture. Kim et al. (2017) proposed a transferred generative adversarial network (tGAN) for detecting and classifying malware. In their method, images converted by malware are used to pre-train malware detector. Recently, Yuan et al. (2020) converted the malware's binary sequences into fixed-size Markov images according to the byte transmission probability matrix, and then used a deep convolutional neural network (CNN) to classify malware based on the Markov images without scaling. As for the images generated by the opcode sequences, Zhang et al. (2016) disassembled PE files into opcode sequences to generate images and then used CNN and softmax classifier for recognizing malware variant images. Besides, Ni et al. (2018) proposed a malware classification method using opcode simhash and CNN. Their method converts the disassembled codes of malware into gray images based on simhash and then identifies malware families by CNN. As mentioned above, however, the disassembly technique is required for the methods using opcodes to generate malware images. The gray images converted by the binary sequences of a PE file do not require disassembled code, but these images mainly give the representations of the PE file's binary information and it is difficult to identify the section distribution information such as the number, order, and size of sections from them.

2.2.2. *Methods based on multiple modalities of data*

Malware classification methods based on multiple modalities rely on more than one type of feature or modality of data to classify malware. Yan et al. (2018) proposed a malware detection and classification method named MalNet. MalNet learns features from the gray images generated from

raw binary files and opcode sequences obtained from decompiled files respectively, and then it takes a stacking ensemble to classify malware. An image-based malware classification method using ensemble of CNNs (IMCEC) was proposed by Vasan et al. (2020). IMCEC uses static features and combines malware visualization and an ensemble of CNNs. Narayanan and Davuluru (2020) proposed an approach of fusing both RNN- and CNN-based architectures for malware classification. Both the assembly and compiled files are used as the inputs of this approach, and a simple CNN (Davuluru et al., 2019), AlexNet, ResNet, VGG16, and LSTM are respectively used to extract 9 features (45 features in all) and are then classified using a Logistic Regression (LR) or SVM. In general, the methods combining different modalities or types of information can achieve higher accuracies than the methods based on one modality of data for malware classification, but they usually require much more computation.

Different from the existing work, in this article, we explore the effect of the section distribution information of a PE file on malware classification and propose a novel visualization method that further emphasizes the section distribution information of a PE file in the generated image. Based on CoLab image, VGG16 (Simonyan and Zisserman, 2015), and SVM, an image-based malware classification approach named MalCVS is proposed.

3. The proposed MalCVS

3.1. Motivation

As mentioned above, the malware image converted by the binary sequences of a PE file is wildly used in the existing malware classification methods based on malware image and deep learning. PE file consists of header and sections. Suppose that there are n samples in a kind of malware family M, $n \geq 2$. For a sample $m_i \in M$, $i \in \{1,2,...,n\}$, its sections S_i can be represented as $\{s_{i_1}, s_{i_2}, ..., s_{i_k}\}$ and its section names are $\{na_{i_1}, na_{i_2}, ..., na_{i_k}\}$, k is the number of sections of m_i. Due to at least one section must be contained if a PE file can run, so $k \geq 1$. In m_i, a section s_{i_x} can be specified by its starting address (expressed as $address_{i_x}$) and the size of this section (expressed as $size_{i_x}$), $x \in \{1,2,...,k\}$. Specifically, $address_{i_1}$ of s_{i_1} is a fixed value of 1000 (in hexadecimal), and the rest of $address_{i_x}$ changes correspondingly according to $size_{i_x}$. From the previous works (Narayanan et al., 2016; Nataraj et al., 2011), we have known that different malware belonging to the same family seen to have similar binary content and image texture. Since the number, order, and size of sections are important attribute characteristics of the PE files, these section distribution information of two different malware samples belonging to the same family could be similar. The section distributions of two malware samples are shown schematically in Fig. 2. As shown in Fig. 2, the number, order, and size of sections of the two malware samples belonging to the same family are similar, even if their binary sequences are something different. However, it is difficult to identify the section distribution information from the current gray images converted by the binary sequences of PE files. Below we explain it through an example.

The binary sequences of a PE file usually contain a varied number of paddings that equal to 0 or other values for the reason of the file alignment mechanism or other. In the current gray image converted by the binary sequences of a PE file, the paddings are exhibited as boundaries of adjacent bytecode blocks and usually divide one section of the PE file into two or more parts. Since the number of padding bits is different for different malware, the boundaries in the current gray image do not correspond well to the actual boundaries of adjacent sections of the original PE file. Fig. 3 depicts a gray image converted by the binary sequences of a malware sample. As shown in Fig. 3, the boundary marked in blue in this gray image is not the actual boundary of the two adjacent sections of this sample, and the actual boundary (mark in red) of these two sections is inconspicuous. Correspondingly, the actual number, order, and size of sections of the PE file are hard to identify from the generated gray image. Therefore, we introduce the colored label boxes into malware images to further emphasize the section distribution information and then construct a malware classification method based on these new images.

3.2. MalCVS framework

The MalCVS framework proposed in this article is shown in Fig. 4. The PE files are first converted to CoLab images, and then a fine-tuned VGG16 model is used for image feature extraction. Subsequently, the extracted features are employed to build a multi-class SVM model. The specific CoLab image generation, image feature extractor, and classification model of MalCVS are given below.

3.3. CoLab image generation

Considering the importance of structural characteristics of PE files in classifying malware families, this article extracts the section distribution information of a PE file to generate a CoLab image. Many of the PE file's features are inherited from the common object file format. The structure of the PE file is shown in Fig. 5, which shows that the PE file generally divides into DOS headers, PE file headers, section tables, sections and debug information. Specifically, the PE file header contains the number of sections; each section header in the section table indicates the start position, size of each section, and section name. As shown in Fig. 6, to generate the CoLab image, the field of "number of sections" in the OptionalHeader, the fields of "pointer to raw data", "size of raw data" and "section name" of each section in the section table need to be extracted, and the little-endian storage inversion is used to restore these values for subsequent CoLab image generation. The n in Fig. 6 represents the number of sections, s_i and p_i, $i \in \{1,2,...,n\}$, respectively represent the starting address and size of the i-th section.

To generate a CoLab image, the section distribution information extracted in the previous step and binary sequences of the PE file are used. In generating a CoLab image, the bytes of a PE file are used as pixel points in the image, image's width is set to 256 and its height changes due to size of the PE file, and then colored label boxes are used to mark

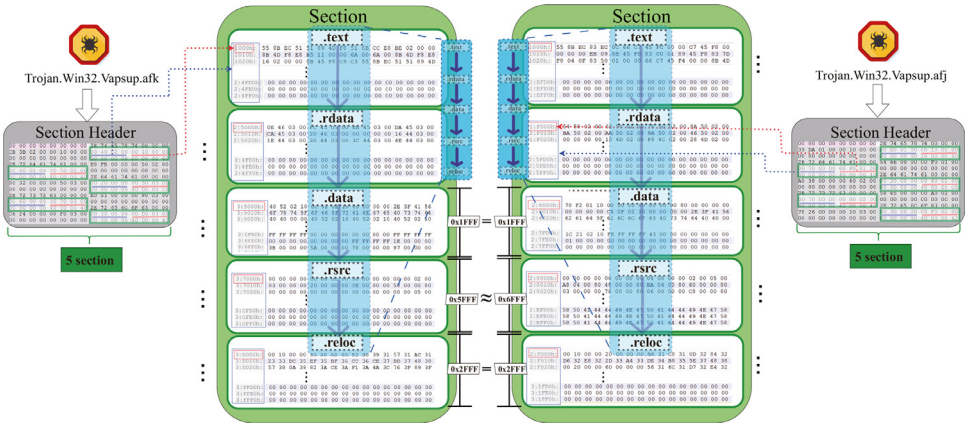


Fig. 2 – The section distributions of two malware samples belonging to the same family.

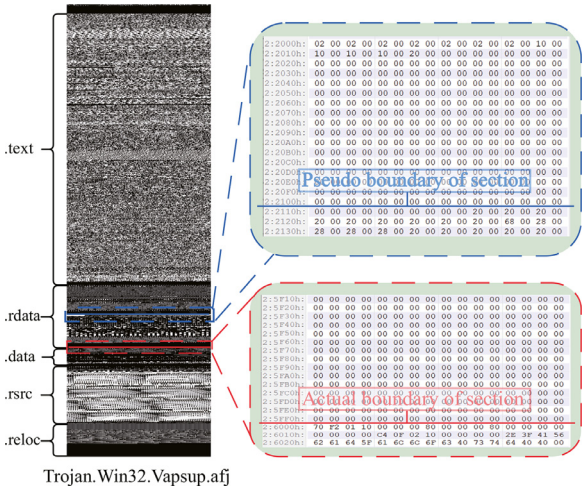


Fig. 3 – A schematic diagram for the pseudo and actual boundaries of a sample.

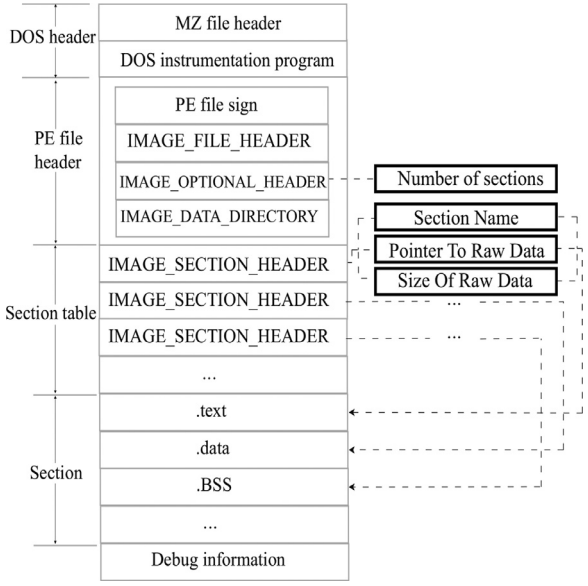


Fig. 5 – PE file structure.

the sections of the PE file in the generated image. When drawing the label boxes, each section having the conventional section name, such as ".text" and ".data", is assigned to a different fixed color, and the section having the unconventional section name is assigned to the color from the preset color sequence according to the section order that the section appears. Note that the value of line thickness t is associated with the size of the PE file, namely, when drawing the CoLab image of different malware, its line thickness may be different. The process of generating CoLab images is given in Algorithm 1.

Fig. 7 shows the traditional gray image and CoLab image of a PE file. As given in Fig. 7, compared to the traditional gray image, the CoLab image not only contains binary information

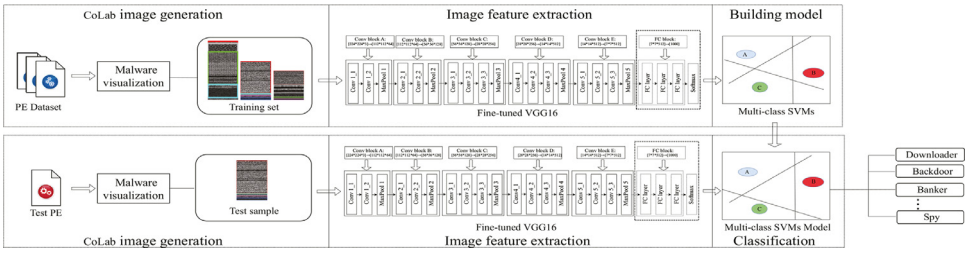


Fig. 4 – MalCVS framework.

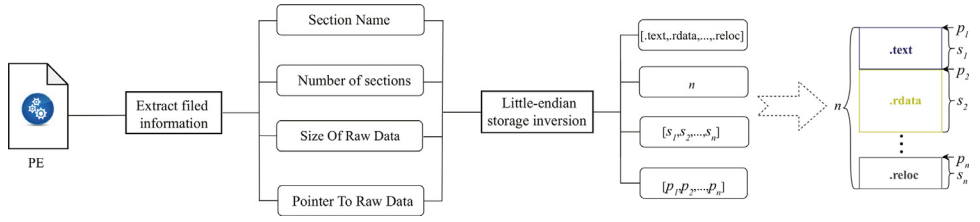


Fig. 6 – Schematic diagram of the section distribution information extraction.

Algorithm 1 CoLab Image Generation

Input: PE software set, *E_Set*;
Output: CoLab image set, *M_Set*;

for $e_i \in E\_Set$ **do**
  /* t_i is the label box thickness, l_i is size of e_i in bytes, k is the thickness interval value for each 50kb file size */
  $t_i \Leftarrow 2 + \lfloor (l_i \div 1024) \div 50) \rfloor \times k$;
  $g_i \Leftarrow$ gray image i generated by sample e_i
  /* S_i is the section size list of e_i, n is the number of sections of e_i */
  $S_i = [s_{i_1}, s_{i_2}, ..., s_{i_n}]$;
  /* $Address_i$ is the starting address list of each section of e_i */
  $Address_i = [address_{i_1}, address_{i_2}, ..., address_{i_n}]$;
  /* Na_i is the section name list of each section of e_i */
  $Na_i = [na_{i_1}, na_{i_2}, ..., na_{i_n}]$;
  for j=0 to n **do**
    $(x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2}) \Leftarrow$ calculate the upper left and lower right coordinates of section i on g_i from $Address_i$ and S_i;
    $C[j] \Leftarrow [(x_{j_1}, y_{j_1}), (x_{j_2}, y_{j_2})]$;
    /* Dic{*na*:c} is the color dictionary corresponding to some conventional section names */
    if $na_{i_j} \in$ Dic{*na*:c} **then**
      $color_{i_j} \Leftarrow$ Dic{na_{i_j}}
    else
      /* R is the preset color sequence according to the section order that the section appears */
      $color_{i_j} \Leftarrow R[j]$
    end if
    $m_i \Leftarrow$ draw a label box with t_i and $color_{i_j}$ on the $C[j]$ of g_i;
  end for
  $M\_Set \Leftarrow m_i$;
end for
return *M_Set*

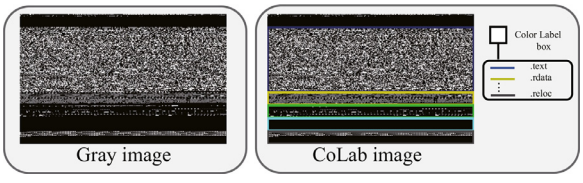


Fig. 7 – The gray image and CoLab image converted by a PE file.

but also emphasizes the section distribution information of the original PE file.

3.4. *Image feature extraction*

Deep learning currently provides good performance in image recognition. The VGG16 model is a deep learning model from ImageNet (Lab, 2016). Owing to VGG16 model has high performance in different scenarios of visualization image classification (Rezende et al., 2018), this article uses a fine-tuned VGG16 as the feature extractor for CoLab image. Since CoLab images must be converted into a uniform size when they are used as inputs to VGG16, the sizes of CoLab images are uniformly scaled to (224, 224, 3). Fig. 8 gives the structure of the fine-tuned VGG16 model used in MalCVS.

As shown in Fig. 8, compared with the original VGG16, the fine-tuning VGG16 used in MalCVS removes the three fully connected layers and one softmax layer, namely, it only retains the previous five convolution modules for extracting image features. The parameters of this structure are less and thus it has much less time and space consumption during training than that of the original VGG16. Each convolution module contains several convolution layers and a maximum pooling layer, where the convolution layer is used to extract the feature from the original CoLab image. The specific conversion rules are as follows.

The original CoLab image m of size $w_{org} * h_{org}$ will be converted into an image m_f of size $w_{new} * h_{new} * n_1$ by passing through n_1 convolution kernels of size (k_c, k_c) with stride s_c and padding p_c, where the w_{new} and h_{new} are calculated by Eqs. (1) and (2), respectively.

$$w_{new} = \left\lfloor \frac{w_{org} - k_c + 2 \times p_c}{s_c} \right\rfloor + 1 \tag{1}$$

$$h_{new} = \left\lfloor \frac{h_{org} - k_c + 2 \times p_c}{s_c} \right\rfloor + 1 \tag{2}$$

To reduce the computational complexity of the neural network, the maximum pooling layer uses n_2 filters of size (k_p, k_p) with stride s_p to extract the maximum value in each step to convert the features of the convolution layer. After this process, image m_f is reduced to the size of $w^*_{new} * h^*_{new} * n_2$, and the w^*_{new} and h^*_{new} are calculated by Eqs. (3) and (4), respectively.

$$w^*_{new} = \left\lfloor \frac{w_{new} - k_p}{s_p} \right\rfloor + 1 \tag{3}$$

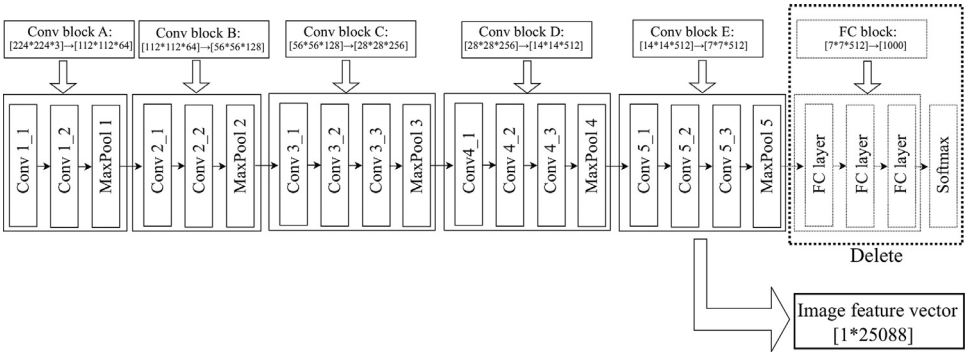


Fig. 8 – The fine-tuned VGG16 used in MalCVS.

$$h^*_{new} = \left\lfloor \frac{h_{new} - k_p}{s_p} \right\rfloor + 1 \tag{4}$$

Through the multiple convolutional layers and maximum pooling layers as well as a flatten operation, a feature vector with 25088 dimensions is extracted. The extracted feature vectors are then used to build a classification model for malware classification.

3.5. *Classification model building*

In this phase, the feature vectors extracted by VGG16 will be used to train a malware classification model. As mentioned in Section 3.4, the dimension of each feature vector used in this training phase is 25088. These feature vectors belong to the high-dimensional data, thus dealing with them might face the issues of "curse of dimensionality" and "data sparsity". Therefore, it is necessary to adopt a classification algorithm that can handle high-dimensional data.

Among the currently popular classification algorithms, SVM is a good option for us since it is so effective to process high-dimensional data (Wu et al., 2020). The final decision function of SVM is only determined by a few support vectors, thus its computational complexity depends on the number of support vectors rather than space dimension, which avoids the "curse of dimensionality" in a sense. Besides, previous researches have shown that SVM has good performance on large datasets and has proven to be highly effective for malware classification (Mourtaji et al., 2019; Narayanan et al., 2016). Linear SVM is employed in MalCVS to build the malware classification model. After the training phase, the built classification model can be used to determine which malware belongs to which malware family.

4. Experimental evaluation

To evaluate the classification performance of MalCVS, we conducted experiments on two malware datasets. The following is a detailed introduction from four aspects: experimental settings and datasets, evaluation metrics, experimental results, and image similarity assessment.

4.1. *The experimental settings and datasets*

The VGG16 and SVM models of the MalCVS are respectively constructed by python with keras[1] and sklearn[2]. Except for fine-tuning the VGG16 model, the remaining parameters of VGG16 and SVM are set by default. To evaluate the generalization performance of MalCVS, we use 10 fold cross-validation, namely, the dataset is divided into 10 subsets of equal size, the i-th subset is used as the test data in turns, the remaining subsets are used as the training data, and the process is repeated 10 times. The experiments were conducted with an Intel i7-9750H machine with 16 GB memory.

Two malware datasets are used to conduct experiments in this article, which are the VXV and BIG-2015 datasets. The details of these two datasets are shown in Tables 1 and 2. In the VXV dataset, there are 6398 PE files belonging to 16 malware families, each of which is labeled with a family when downloading from VX-heaven or Virusshare. The BIG-2015 dataset (the training dataset provided by Kaggle) contains 10868 malware belong to nine families and each malware has an ASM file and a BYTE file. Note that the section names are missing because the BYTE files in the BIG-2015 dataset do not contain the header information, so their sections are assigned to the colors from the preset color sequence according to the order that each section appears. Besides, we replace the bytes "??" in each BYTE file of BIG-2015 with "FF".

4.2. *Evaluation metrics*

Four widely used evaluation metrics, recall, precision, accuracy, and F1-score, are used to evaluate the classification performance, and confusion matrix is employed to give the detailed classification performance of each family. The metrics of recall, precision, accuracy can be calculated based on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) and they can be calculated by Eqs. (5)–(7).

$$recall = \frac{TP}{TP + FN} \tag{5}$$

$$precision = \frac{TP}{TP + FP} \tag{6}$$

[1] https://keras.io
[2] https://scikit-learn.org

Table 1 – Details of the VXV dataset used in experiment.

FamilyID	Class	Family	Quantity	Average file size (kb)
1	Backdoor	Bifrose	160	70.42
2	Backdoor	Hupigon	1359	496.46
3	Backdoor	Popwin	201	51.99
4	Backdoor	Shark	339	848.08
5	Ransomeware	Ceber	118	160.7
6	Ransomeware	Gandcrab	159	73.64
7	Trojan	Obfuscated	678	438.39
8	Trojan	Vapsup	734	199.53
9	Trojan-Banker	Banker	167	870.49
10	Trojan-Downloader	Hmir	184	136.83
11	Trojan-Downloader	Zlob	877	46.53
12	Trojan-GameThief	Magania	156	135.34
13	Trojan-GameThief	OnLineGames.aro	328	26.26
14	Trojan-GameThief	OnLineGames.blh	213	29.28
15	Trojan-Spy	Pophot	283	111.22
16	Trojan-Spy	Zbot	442	95.8

Table 2 – The sample distribution on the BIG-2015 dataset.

FamilyID	Family	Quantity	Average size (kb) of the BYTE files
1	Ramnit	1541	723.72
2	Lollipop	2478	2846.45
3	Kelihos_ver3	2942	4386.05
4	Vundo	475	547.34
5	Simda	42	2222.82
6	Tracur	751	879.47
7	Kelihos_ver1	398	2466.75
8	Obfuscator.ACY	1228	403.87
9	Gatak	1013	1247.59

Table 3 – Classification results obtained by MalCVS and opcode n-grams based, gray image-based, and CoLab image-based methods.

Method	Accuracy	Recall	Precision	F1-score
Opcode+N-gram+DT	91.97%	93.83%	92.32%	92.93%
Opcode+N-gram+KNN	90.53%	91.45%	91.65%	90.98%
Opcode+N-gram+SVM	92.37%	93.49%	93.11 %	93.11%
Gray+VGG16+DT	79.99%	83.64%	82.00%	82.53%
Gray+VGG16+KNN	92.51%	94.78%	92.14%	93.17%
Gray+VGG16+SVM	93.87%	95.28%	94.42%	94.75%
CoLab+VGG16+DT	87.50%	89.74%	88.84%	89.08%
CoLab+VGG16+KNN	95.01%	96.69%	94.48%	95.42%
MalCVS	**96.59%**	**97.71%**	**96.94%**	**97.28%**

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

F1-score metric is the harmonic average of the precision and recall and is used as the actual scoring criterion for the classifier. It can be calculated in Eq. (8).

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

Pearson Correlation Coefficient (PCC) is the linear- correlation between the vectors of the two images, which can be used as a criterion for measuring the similarity between the images. If the vectors converted by the two images are $X_1 = \{x_{1_1}, x_{1_2}, x_{1_3},..., x_{1_n}\}$, $X_2 = \{x_{2_1}, x_{2_2}, x_{2_3},..., x_{2_n}\}$, $n \in Z^+$, the PCC can be calculated by Eq. (9), where $\bar{x}_i$ is the average of the corresponding X_i. Besides, the average build time (for building the classifier with the feature vectors of malware) and the classification time (for classifying the feature vecters of malware) in seconds are considered to be the metrics for measuring the efficiency of different classification algorithms.

$$PCC(X_1, X_2) = \frac{\sum_{i=1}^{n}\left(x_{1_i} - \bar{x}_1\right)\left(x_{2_i} - \bar{x}_2\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{1_i} - \bar{x}_1\right)^2}\sqrt{\sum_{i=1}^{n}\left(x_{2_i} - \bar{x}_2\right)^2}} \tag{9}$$

4.3. *Experimental results*

In this section, the classification performances of MalCVS on the VXV and BIG-2015 datasets are given. To accomplish a better observation, we take the classification performance of the gray image-based and opcode n-grams based classification methods as our baselines. Specifically, gray image-based classification methods based on gray image, VGG16 and each of three classification algorithms (KNN with K=3, Decision Tree (DT), and SVM); opcode n-grams based classification methods use 3-gram sequences and the same each of three classification algorithms with the gray image-based methods.

4.3.1. *Experiment on VXV dataset*

A.Classification result

Table 3 shows the classification performance of 10 fold cross-validation on the VXV dataset obtained by MalCVS and

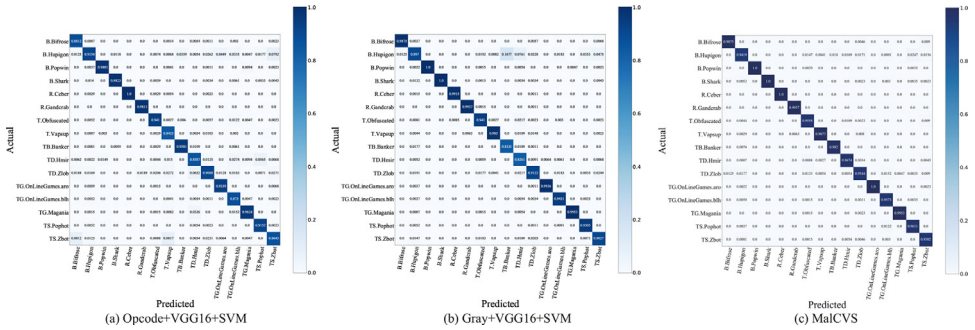


Fig. 9 – Confusion matrices of the three methods based on different feature vector and SVM on VXV dataset.

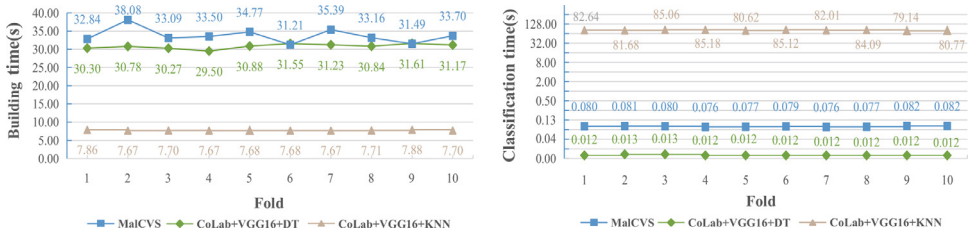


Fig. 10 – Building and classification times of the three methods for each fold of the 10 fold cross-validation on the VXV dataset.

opcode n-grams based, gray image-based, and CoLab image-based methods. As given in Table 3, MalCVS obtains the best classification results, its accuracy reaches 96.59%, which is better than the other opcode n-grams based, gray image-based, and CoLab image-based methods. Moreover, the obtained recall, precision, and F1-score of MalCVS are also higher than the other methods listed in Table 3.

Fig. 9 shows the confusion matrices of MalCVS, "Gray+VGG16+SVM", and "Opcode+N-gram+SVM" on each malware family. In Fig. 9, it can be seen that "Opcode+N-gram+SVM" does not well classify the samples belonging to the families of "B.Bifrose", "T.Vapsup", "TD.Hmir", "TG.OnLineGames.blh" and "TS.Zbot", and the samples belonging to the families of "B.Hupigon", "TB.Banker" and "TD.Hmir" are not well identified by "Gray+VGG16+SVM". As for MalCVS, it outperforms the two methods in all the 16 families and its obtained accuracy for each of the 16 families is higher than 94%.

The building and classification times of the methods of MalCVS, "CoLab+VGG16+DT", and "CoLab+VGG16+KNN" for each fold of 10 fold cross-validation are given in Fig. 10. As shown in Fig. 10, among the three methods, "CoLab+VGG16+KNN" needs the least building time but its classification time is the longest. As for MalCVS and "CoLab+VGG16+DT", they all have high efficiency in classifying malware because their classification times are short. On average, MalCVS only needed about 0.0788s to classify the malware samples from one fold of the 10 fold cross-validation in our experiment.

B.Study of the line thickness and color effect

CoLab image uses several colored label boxes to characterize the position, size, and order of the sections of a PE file. If different line thickness and color are used when drawing the label boxes, different CoLab images will be formed. Fig. 11 gives the CoLab images with line thickness parameter k of 1, 3, 6, and 8 generated by a PE file belonging to the malware family "TS.Pophot", respectively, and Fig. 12 shows the CoLab images of a PE file belonging to the malware family "TS.Pophot" with the same color and the preset colors (both k values are set to 6). As shown in Tables 11 and 12, CoLab image changes accordingly when k value or label colors are different. Therefore, the line thickness and color have a certain impact on the final classification performance of MalCVS. The impacts of the line thickness and color on the classification performance of MalCVS are given below.

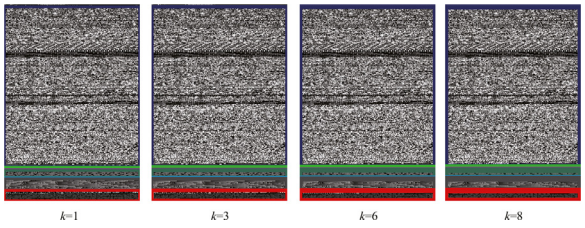


Fig. 11 – CoLab images generated by a PE file with different line thickness parameter.

As mentioned in Section 3.3, the line thickness of the colored label box is associated with the size of PE files and the line thickness parameter k. Table 4 shows the classification results obtained by the CoLab images with k of 1, 3, 6, and 8. As given in Table 4, when using the different k values to generate CoLab images and used for malware classification, there is no significant difference in their obtained results. It also can be seen that when k value was set as 6, MalCVS obtained the best accuracy.

Table 4 – Classification results of MalCVS based on the CoLab images with different line thickness parameter.

Method	Line thickness parameter k	Accuracy	Recall	Precision	F1-score
MalCVS	1	96.45%	97.71%	96.87%	97.24%
	3	96.50%	97.65%	96.88%	97.21%
	6	**96.59%**	97.71%	**96.94%**	**97.28%**
	8	96.39%	**97.73%**	96.54%	97.07%

Table 5 – Classification results obtained by MalCVS with different color settings and other methods.

Method	Color	Accuracy	Recall	Precision	F1-score
Opcode+N-gram+SVM	–	92.37%	93.49%	93.11 %	93.11%
Gray+VGG16+SVM	–	93.87%	95.28%	94.42%	94.75%
	same	95.34%	96.42%	95.40%	95.85%
MalCVS	R*	**96.59%**	**97.71%**	**96.94%**	**97.28%**

* denotes preset color scheme.

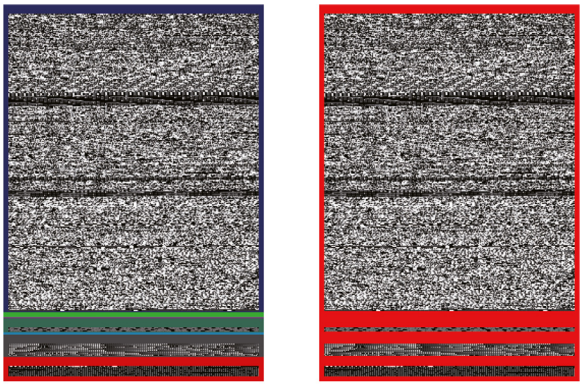


Fig. 12 – CoLab images with the same or different label color.

Table 5 compares the classification performance of MalCVS when drawing the label boxes with the same color and the preset colors. As shown in Table 5, when all the colored label boxes are drawn with the same color (red), the accuracy obtained by MalCVS is 95.34%, this value dropped by 1.25% compared to that obtained by MalCVS with different label colors. This is because the colored label boxes with different colors are beneficial to represent the different sections of a PE file. Note that the classification result obtained by MalCVS with the same label box color is still better than that of "Gray+VGG16+SVM", which indicates that the section distribution information of a PE file is useful for malware classification once again.

4.3.2. *Experiment on BIG-2015 dataset*

Table 6 shows the classification results of 10 fold cross-validation on the BIG-2015 dataset obtained by MalCVS and opcode n-grams based, gray image-based, and CoLab image-based methods. As shown in Table 6, among the nine classification methods, MalCVS obtains the best performances in terms of accuracy, recall, precision, and F1-score.

Table 6 – Classification results obtained by MalCVS and opcode n-grams based, gray image-based, and CoLab image-based methods on BIG-2015 dataset.

Method	Accuracy	Recall	Precision	F1-score
Opcode+N-gram+DT	97.48%	91.69%	95.46%	92.49%
Opcode+N-gram+KNN	92.80%	86.47%	94.76%	88.53%
Opcode+N-gram+SVM	97.37%	93.95%	96.46%	94.67%
Gray+VGG16+DT	92.28%	85.45%	84.33%	84.71%
Gray+VGG16+KNN	97.08%	92.54%	93.51%	92.69%
Gray+VGG16+SVM	98.05%	95.46%	96.40%	95.56%
CoLab+VGG16+DT	93.58%	87.92%	87.78%	87.61%
CoLab+VGG16+KNN	98.05%	95.32%	96.03%	95.40%
MalCVS	**98.94%**	**97.72%**	**98.26%**	**97.91%**

The confusion matrices of the methods of MalCVS, "Gray+VGG16+SVM", and "Opcode+N-gram+SVM" are given in Fig. 13. Fig. 13 shows that the samples belonging to families of "Vundo", "Obfuscator_ACY" and "Simda" are not well classified by the "Opcode+N-gram+SVM". From Fig. 13, we can see that MalCVS exhibited the best performance for all the nine malware families, but the samples belonging to "Simda" family are also not well classified. This is because the number of samples of "Simda" family in BIG-2015 is only 42. Overall, the classification accuracy of MalCVS reaches 98.94%.

Fig. 14 gives the building and classification times of MalCVS, "CoLab+VGG16+DT", and "CoLab+VGG16+KNN" for each fold of 10 fold cross-validation on the BIG-2015 dataset. From Fig. 14, we can see that "CoLab+VGG16+KNN" needs the least building time among the three methods, and the building time of MalCVS is better than that of "CoLab+VGG16+DT". As for the classification time, "CoLab+VGG16+DT" and MalCVS exhibit better performances than "CoLab+VGG16+KNN". On av-

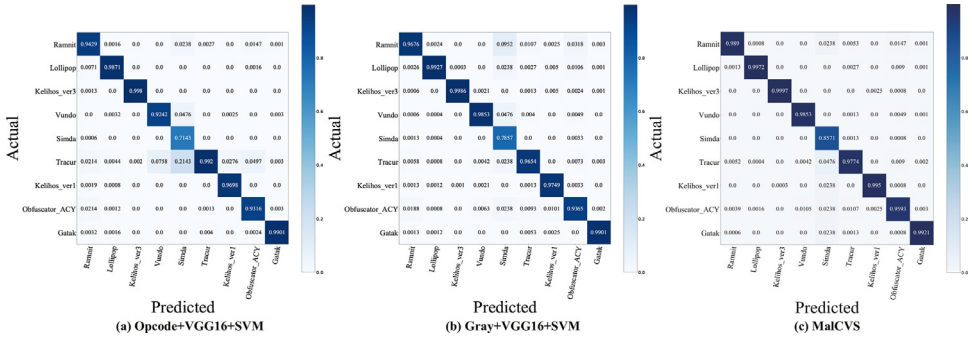


Fig. 13 – Confusion matrices of three methods based on different feature vector and SVM.

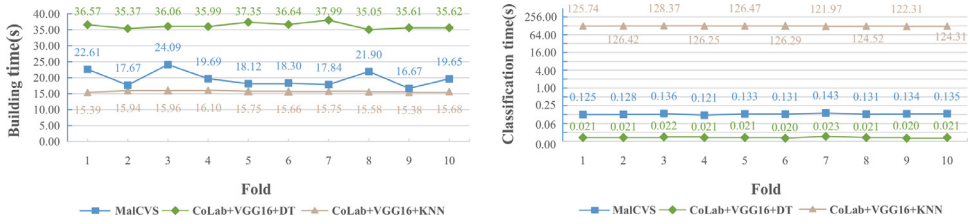


Fig. 14 – Building and classification times of the three methods for each fold of the 10 fold cross-validation on BIG-2015 dataset.

erage, MalCVS takes 0.1314s to classify the malware samples from one fold of the 10 fold cross-validation in our experiment.

The performance of MalCVS is compared with some other malware classification methods reported in the literature, these methods can be divided into different categories and all used the whole BIG-2015 training dataset containing 10868 samples. More specifically, "IMG+PCA+KNN" (Narayanan et al., 2016), "BYTE+Strand" (Drew et al., 2016) and "ASM+Strand" (Drew et al., 2017) can be classified as methods based on static features and traditional machine learning; "IMG+tGAN" (Kim et al., 2017) and "IMG+CNN" (Gibert et al., 2019) are the methods based on malware images and deep learning; "BYTE+Entropy+CNN" (Gibert et al., 2018b) and "BYTE+Autoencoder+DRN" (Gibert et al., 2018a) are the methods based on static features and deep learning. Besides, three deep learning methods based on multiple modalities of data ("MalNet" (Yan et al., 2018), "Ensemble+LR" (Narayanan and Davuluru, 2020), and "Ensemble+SVM" (Narayanan and Davuluru, 2020) are provided for reference. In the above methods, "Byte+Strand", "IMG+PCA+KNN", "IMG+tGAN", "BYTE+Autoencoder+DRN", "IMG+CNN" and MalCVS are based on only malware's binary sequences and therefore do not involve disassembling PE files. Table 7 presents the 10 fold cross-validation accuracy achieved by these methods with the same dataset. On the one hand, the three methods based on multiple modalities of data get higher accuracies than the methods based on one modality of data, and "Ensemble+SVM" achieves the best accuracy. On the other hand, among the methods based on one modality of data listed in Table 7, MalCVS obtains higher accuracy than the other methods, it could be a competitive candidate for malware classification.

Table 7 – Accuracies obtained by MalCVS and other malware classification approaches on the BIG-2015 dataset.

Approaches	Accuracy
Based on one modality of data	
BYTE+Strand (Drew et al., 2016)	97.41%
IMG+PCA+KNN (Narayanan et al., 2016)	96.6%
IMG+tGAN (Kim et al., 2017)	96.39%
ASM+Strand (Drew et al., 2017)	98.59%
BYTE+Autoencoder+DRN (Gibert et al., 2018a)	98.61%
BYTE+Entropy+CNN (Gibert et al., 2018b)	97.08%
IMG+CNN (Gibert et al., 2019)	97.5%
MalCVS	98.94%
Based on multiple of data	
MalNet* (Yan et al., 2018)	99.3%
Ensemble+LR** (Narayanan and Davuluru, 2020)	99.5%
Ensemble+SVM** (Narayanan and Davuluru, 2020)	99.8%

* denotes the partition of training and test datasets didn't make public. ** denotes the dataset is split into groups of 72%, 8%, and 20% for training, validation, and test, respectively.

4.4. *Image similarity assessment*

To further analyze the experimental results, this section quantifies image similarities of CoLab images or gray images belonging to the same malware family using the PCC. As mentioned in Section 4.3.1, the samples belonging to the families of "B.Hupigon", "TB.Banker" and "TD.Hmir" from the VXV dataset were not well classified by "Gray+VGG16+SVM". Therefore, we randomly select 10 samples from each of the three malware families and convert them into gray and CoLab images. The PCCs of each of the two types of images within these three families are calculated, and the average values for the

Table 8 – Image similarities for CoLab and gray images within the same family.

Family	Image	Average PCC
B.Hupigon	Gray	0.0190
	CoLab	0.3539
TB.Banker	Gray	0.3041
	CoLab	0.5000
TD.Hmir	Gray	0.0295
	CoLab	0.2159

images belonging to each malware family are used to measure their similarity. The PCCs of the two types of malware images within the three families are given in Table 8. As shown in Table 8, for all the three families of "B.Hupigon", "TB.Banker" and "TD.Hmir", the CoLab images belonging to the same family have a higher average PCC than gray images. This result shows that, compared to the gray images, the CoLab images belonging to the same malware family are more similar, which is beneficial to the better classification result obtained by MalCVS for the three malware families.

5. Conclusion

Gray images are wildly used in the current malware classification methods based on malware images and deep learning. This article found that the section distribution information of a PE file is very useful for classifying malware and this information in the current gray image converted by the binary sequences is difficult to identify. To improve the classification accuracy, this article proposes a visualization method to further emphasize the section distribution information of a PE file in the generated CoLab images. Besides, based on CoLab image, VGG16, and SVM, a malware classification method called MalCVS is constructed. The experimental results on the VXV and BIG-2015 datasets show that MalCVS can obtain good classification performance, which is better than the gray image-based and opcode n-grams based classification methods.

Since malware will rewrite or confuse the header field of PE files after packing, the proposed visualization method can not well characterize packed malware. In future work, we will continue to look for the malware image that has a stronger ability to characterize various malware including packed malware.

Declaration of Competing Interest

We declare that we have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the manuscript entitled " Image-based malware classification using section distribution information".

CRediT authorship contribution statement

Mao Xiao: Methodology, Software, Investigation, Writing – original draft. **Chun Guo:** Conceptualization, Methodology, Funding acquisition, Writing – review & editing. **Guowei Shen:** Formal analysis, Funding acquisition, Resources, Writing – review & editing. **Yunhe Cui:** Data curation, Validation, Writing – review & editing. **Chaohui Jiang:** Data curation, Validation, Writing – review & editing.

Acknowledgments

The authors thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the article. This work is supported by the National Natural Science Foundation of China under Grant 62062022, the Science and Technology Foundation of Guizhou Province No. [2020]1Y268, the Open Project of Guizhou Provincial Key Laboratory of Public Big Data No. 2017BDKFJJ025.

REFERENCES

Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G. Novel feature extraction, selection and fusion for effective malware family classification. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy; 2016. p. 183–94. doi:10.1145/2857705.2857713.

Alazab M. Profiling and classifying the behavior of malicious codes. J. Syst. Softw. 2015;100:91–102. doi:10.1016/j.jss.2014.10.031.

AV-TEST. Av-test: the av-test malware statistics 2020. https://www.av-test.org/en/statistics/malware/; 2020. Online.Accessed: 7 July 2020.

Boukhtouta A, Mokhov SA, Lakhdari NE, Debbabi M, Paquet J. Network malware classification comparison using DPI and flow packet headers. J. Comput. Virol. Hack. Tech. 2016;12(2):69–100.

Damodaran A, Troia FD, Visaggio CA, Austin TH, Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hack.Tech. 2017;13(1):1–12. doi:10.1007/s11416-015-0261-z.

Davuluru VSP, Narayanan BN, Balster EJ. Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs. In: Proceedings of the IEEE National Aerospace Electronics Conference, NAECON; 2019. p. 273–8. doi:10.1109/NAECON46414.2019.9058025.

Drew J, Hahsler M, Moore T. Polymorphic malware detection using sequence classification methods and ensembles. Eurasip J. Inf. Secur. 2017;2017(1). doi:10.1186/s13635-017-0055-6.

Drew J, Moore T, Hahsler M. Polymorphic malware detection using sequence classification methods. In: Proceedings - 2016 IEEE Symposium on Security and Privacy Workshops, SPW 2016; 2016. p. 81–7. doi:10.1109/SPW.2016.30.

Gandotra E, Bansal D, Sofat S. Malware analysis and classification: a survey. J. Inf. Secur. 2014;5(2):56–64.

Gibert D, Mateu C, Planes J. An end-to-end deep learning architecture for classification of malware's binary content. In: Kurková V, Manolopoulos Y, Hammer B, Iliadis L, Maglogiannis I, editors. In: Artificial Neural Networks and Machine Learning – ICANN 2018. Cham: Springer International Publishing; 2018a. p. 383–91.

Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. 2020;153:102526. doi:10.1016/j.jnca.2019.102526.

Gibert D, Mateu C, Planes J, Vicens R. Classification of malware by using structural entropy on convolutional neural networks. In: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018; 2018b. p. 7759–64. http://web.udl.cat/usuaris/m4372594/IAAI18.pdf

Gibert D, Mateu C, Planes J, Vicens R. Using convolutional neural networks for classification of malware represented as images. J. Comput. Virol. Hack.Tech. 2019;15:15–28. doi:10.1007/s11416-018-0323-0.

Hansen SS, Larsen TMT, Stevanovic M, Pedersen JM. An approach for detection and family classification of malware based on behavioral analysis. In: 2016 International Conference on Computing, Networking and Communications, ICNC 2016; 2016. p. 1–5. doi:10.1109/ICCNC.2016.7440587.

Hassen M, Chan PK. Scalable function call graph-based malware classification. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy; 2017. p. 239–48. doi:10.1145/3029806.3029824.

Jamalpur S, Navya YS, Raja P, Tagore G, Rao GRK. Dynamic malware analysis using cuckoo sandbox. In: Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018; 2018. p. 1056–60. doi:10.1109/ICICCT.2018.8473346.

Kalash M, Rochan M, Mohammed N, Bruce NDB, Wang Y, Iqbal F. Malware classification with deep convolutional neural networks. In: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS); 2018. p. 1–5. doi:10.1109/NTMS.2018.8328749.

Kebede TM, Djaneye-Boundjou O, Narayanan BN, Ralescu A, Kapp D. Classification of malware programs using autoencoders based deep learning architecture and its application to the microsoft malware classification challenge (BIG 2015) dataset. In: 2017 IEEE National Aerospace and Electronics Conference (NAECON); 2017. p. 70–5. doi:10.1109/NAECON.2017.8268747.

Kim JY, Bu SJ, Cho SB. Malware detection using deep transferred generative adversarial networks. Lecture Notes in Computer Science, 10634; 2017. p. 556–64.

Kyoungsoo H, Boojoong K, Gyu IE. Malware analysis using visualized image matrices. Sci. World J. 2014;2014:132713.

Lab S.V.. Inagenet. 2016. http://www.image-net.org/Online.Accessed: 7 May 2020.

Lindorfer M, Neugschwandtner M, Platzer C. Marvin: efficient and comprehensive mobile app classification through static and dynamic analysis, 2; 2015. p. 422–33. doi:10.1109/COMPSAC.2015.103.

Liu X, Lin Y, Li H, Zhang J. A novel method for malware detection on ML-based visualization technique. Comput. Secur. 2020;89(Feb.) 101682.1–101682.12.

Manavi F, Hamzeh A. A new method for malware detection using opcode visualization. In: 2017 Artificial Intelligence and Signal Processing Conference (AISP); 2017. p. 96–102. doi:10.1109/AISP.2017.8324117.

Microsoft. Microsoft malware classification challenge (BIG 2015). https://www.kaggle.com/c/malware-classification; 2015. Online.Accessed: 10 February 2020.

Mourtaji Y, Bouhorma M, Alghazzawi D. Intelligent framework for malware detection with convolutional neural network, Part F148154; 2019. p. 1–6. doi:10.1145/3320326.3320333.

Narayanan BN, Davuluru VSP. Ensemble malware classification system using deep neural networks. Electronics 2020;9(5):721.

Narayanan BN, Djaneye-Boundjou O, Kebede TM. Performance analysis of machine learning and pattern recognition algorithms for malware classification. In: Proceedings of the IEEE National Aerospace Electronics Conference, NAECON. IEEE; 2016. p. 338–42.

Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: visualization and automatic classification. In: ACM International Conference Proceeding Series; 2011. p. 1–7. doi:10.1145/2016904.2016908.

Ni S, Qian Q, Zhang R. Malware identification using visualization images and deep learning. Comput. Secur. 2018;77:871–85. doi:10.1016/j.cose.2018.04.005.

Rezende E, Ruppert G, Carvalho T, Ramos F, de Geus P. Malicious software classification using transfer learning of resnet-50 deep neural network. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA); 2017. p. 1011–14. doi:10.1109/ICMLA.2017.00-19.

Rezende E, Ruppert G, Carvalho T, Theophilo A, Ramos F, de Geus P. Malicious software classification using VGG16 deep neural networks bottleneck features. Adv. Intell. Syst. Comput. 2018;738:51–9. doi:10.1007/978-3-319-77028-4_9.

San CC, Thwin MMS, Htun NL. Malicious software family classification using machine learning multi-class classifiers. In: International conference on computational science and technology. springer; 2018. p. 423–33.

Vasan D, Alazab M, Wassan S, Safaei B, Zheng Q. Image-based malware classification using ensemble of CNN architectures (IMCEC). Comput. Secur. 2020;92:101748. doi:10.1016/j.cose.2020.101748.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. The International Conference on Learning Representations, ICLR 2015.

Virusshare. Virusshare.com – because sharing is caring. https://virusshare.com/; 2010. Online.Accessed: 12 April 2020.

Vxheaven. Vxheavens. https://archive.org/download/vxheavens-2010-05-18; 2010. Online.Accessed: 12 April 2020.

Wang R, Su PR, Yang Y, Feng DG. An anti-obfuscation malware variants identification system. Tien Tzu Hsueh Pao/Acta Electronica Sinica 2011;39(10):2322–30. doi:10.1631/jzus.C0910717.

Wu TD, Yen Y, Wang JH, Huang RJ, Lee HW, Wang HF. Automatic target recognition in SAR images based on a combination of CNN and SVM. In: Proceedings - 2020 International Workshop on Electromagnetics: Applications and Student Innovation Competition, iWEM 2020; 2020. p. 1–2. doi:10.1109/iWEM49354.2020.9237422.

Yan J, Qi Y, Rao Q. Detecting malware with an ensemble method based on deep neural network. Sec. Commun. Netw. 2018;2018:17. doi:10.1155/2018/7247095.

Ye Y, Li T, Adjeroh D, Iyengar SS. A survey on malware detection using data mining techniques. ACM Comput. Surv. 2017;50(3):1–40.

Yuan B, Wang J, Liu D, Guo W, Wu P, Bao X. Byte-level malware classification based on Markov images and deep learning. Comput. Secur. 2020;92:101740. doi:10.1016/j.cose.2020.101740.

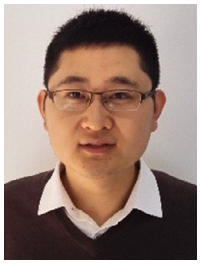
Zhang H, Xiao X, Mercaldo F, Ni S, Martinelli F, Sangaiah AK. Classification of ransomware families with machine learning based on n-gram of opcodes. Future. Gener. Comp. Sy. 2019;90:211–21. doi:10.1016/j.future.2018.07.052.

Zhang J, Qin Z, Yin H, Ou L, Hu Y. Irmd: Malware variant detection using opcode image recognition. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS); 2016. p. 1175–80. doi:10.1109/ICPADS.2016.0155.

Mao Xiao received B.S. degree in information security from Guizhou University in PR China in 2018. He is currently pursuing the M.S. degree in computer science and technology from Guizhou University. His recent research interests include information security and malware classification.

Chun Guo received Ph.D. in information security from Beijing University of Posts and Telecommunications in July 2014. He is currently an associate professor in the College of Computer Science and Technology, Guizhou University, PR China. His research interests include data mining, intrusion detection and malware Detection.

Guowei Shen received his Ph.D. degree from Harbin Engineering University. He is currently an associate professor of Guizhou University, PR China. His main research interests include big data, computer network and cybersecurity.



Yunhe Cui received his Ph.D. degree from the Southwest Jiaotong University, Chengdu, Sichuan, PR China. He is currently a lecturer of Guizhou University, PR China. His research interests include software-defined networking, network security, traffic engineering, swarm intelligence algorithm, data centers, edge computing and cloud computing.

Chaohui Jiang received B.S. in Precision measuring instrument from Chengdu university of science and technology (Sichuan university) in July 1987, received M.S. in machine manufacturing from Chengdu university of science and technology (Sichuan university) in July 1990. He is currently a professor in the College of Computer Science and Technology, GuiZhou University, PR China. His research interests include network security and operating system security.