Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.XXXXXXX

A New Learning Approach to Malware Classification using Discriminative Feature Extraction

YA-SHU LIU[1,2], YU-KUN LAI[3] (Member, IEEE), ZHI-HAI WANG[1], HAN-BING YAN[4]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044, China
[2]School of Electrical and Information Engineering, Beijing University of Civil Engineering and Architecture, Beijing 100044, China
[3]School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA, UK
[4]National Computer Network Emergency Response Technical Team/Coordination Center of China, 100029, China

Corresponding author: Ya-Shu Liu (liuyashu@bucea.edu.cn)

This work was supported in part by the National Science Foundation of China under Grants U1736218, 61672086, and in part by the National Key Technology Research and Development Program of China under Grants 2015BAK21B01, 2018YFB0803604.

ABSTRACT With the development of Internet, malware has become one of the most significant threats. Recognizing specific types of malware is an important step towards effective removal. Malware visualization is an important branch of malware static analysis techniques, where a piece of malware is turned into an image for visualization and classification. Despite great success, it is still difficult to extract effective texture feature representations for challenging datasets. Existing methods use global image features which are sensitive to relative code locations. In this paper, we present a new learning framework to obtain more discriminative and robust feature descriptors. The proposed method works with existing local descriptors such as LBP (Local Binary Patterns) and dense SIFT (Scale-Invariant Feature Transform), by grouping them into blocks and using a new bag-of-visual-words (BoVW) model to obtain robust features, which are more flexible than global features and more robust than local features. We evaluate the proposed method on three malware databases. Experimental results demonstrate that the obtained descriptors lead to state-of-the-art classification performance.

INDEX TERMS malware visualization, image texture, feature descriptors, malware classification.

I. INTRODUCTION

Malware (e.g. viruses, worms and Trojan horses) has become one of the most significant threats on the Internet. With the help of generation tools, it becomes easy to generate new malware, resulting in a very rapid increase in the number of malware. AV test reported that around 81,598,221 new malware samples were obtained in 2017, a 14% increase compared to the previous year. Among all these malware attacks, over 67% targeted Windows systems [1]. It has caused serious threat. For example, the ransomware "WannaCry" spread over 100 countries in the world and caused damage of 8 billion US dollars. Furthermore, those new variant malicious code files have similar behavior as benign files, making them harder to be detected, which has posed a significant challenge to anti-virus vendors. Although various analysis techniques have been studied to deal with malware variants, they are not sufficient to address increasing avoidance techniques applied in malware. New analysis techniques are still demanded to improve the analysis efficiency. Among different techniques, malware visualization has recently been proposed as an effective approach.

In this paper, we propose a new method that classifies malware families using malware visualization. The method transforms malware binary files to grayscale images. To obtain discriminative features, we present a new learning framework which is formulated as a multi-layered model to characterize and analyze malware images using bag-of-visual-words (BoVW). Starting from existing local descriptors (LBP or dense SIFT), we group them into blocks and build histograms. The extracted features are more flexible than global features (e.g. GIST) and more robust than local features. We evaluate the proposed method on three datasets, which are all from the Windows platform. Experimental results demonstrate that the obtained descriptors are robust and discriminative, which lead to state-of-the-art classification performance, outperforming existing methods.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 gives an overview of our learning framework. Section 4 conducts comparative experiments on three datasets and analyzes the results. Finally, conclusions are drawn in Section 5.

II. RELATED WORK

Various malware analysis and classification methods have been proposed, including signature-based detection [2], [3], behavior-based methods [4], [5], instruction frequency-based methods [6]–[10], opcode-sequence based methods [11]–[13], etc.

Among them, some techniques help analysts analyze malware with feature visualization. Based on the observation that control flow information could be used to identify malware variants, Cesare and Yang [14] developed a control flow graph based malware classification method. Trinius et al. [15] explored two visualization techniques, namely treemaps and thread graphs, to visualize the behavior of malicious software by abstracting in different levels the behavior captured in controlled environments, with an aim to help human analysts. Saxe et al. [16] presented a visual analytic approach to analyze and visualize system calls shared among different malware samples. Their system provides two visualization user interfaces, namely a map-like interface showing the overall similarity among samples, and a linked interface to highlight both the similarities and differences between selected samples. Hu et al. [17] developed a system to handle a large number of malware samples efficiently. Each malware sample is represented using their function-call graph, so that finding similar malware samples from the database to a new malware sample can be formulated as a graph matching problem. They further developed an efficient algorithm for searching in the graph database. These methods often use graphs to represent malware, and/or provide high-level visualization of malware to assist analysts.

Conti et al. [18] put forward a method to classify raw binary data into binary fragments of different primitive types such as text, machine instructions, image data and audio data. Nataraj et al. [19] did the pioneering work of using malware visualization for malware analysis. It allows researchers to understand the structures of malware binary files without disassembling. In the paper, a malware binary is represented as a vector of 8 bit unsigned integers, which is then organized into a 2D array and visualized as a grayscale image by treating integers as pixel intensities. In Nataraj's method, it obtains the image's GIST descriptor, a global texture descriptor, and classifies malware images using machine learning. It obtains a high accuracy (0.98) on the dataset (25 families, 9,458 images). However, when applied to a larger malware dataset (36 families, 12,278 images [20]), the accuracy is not satisfactory (only 0.89). Inspired by Nataraj's method, Han et al. [25] proposed a new method by converting binary files into images and generating entropy graphs. The method obtains malware images using the same approach as Nataraj's method and improves Nataraj's classification method by classifying malware families based on the entropy graph similarity. They claim that their method is as good as Nataraj's method.

Although malware images look like usual grayscale images, they are fundamentally different. When the same code appears in different sections of the malicious file, the overall malware image will be changed. Simply using a global texture descriptor as in [19] does not work well for malware images, especially for those samples containing interference information. Thus in this paper, we develop a new method to obtain more robust descriptors in more challenging cases, for example, when the malware samples are too similar in different families or too different in the same family.

III. METHODOLOGY

In this paper, we propose a new framework for malware classification based on malware visualization, where a piece of malware is first converted into an image.

A. MALWARE VISUALIZATION

To turn a malware executable into an image, we first treat its binary code as a sequence of 8-bit unsigned integers in the range of 0 to 255 inclusive. Each value is directly interpreted as the intensity of the pixel, where 0 is black and 255 is white. The sequence is then structured into an 2D array to form an image. Following Nataraj's method, the width of the image varies according to the binary file size [19]. Some examples of malware images are shown in Figure 1. Those are from Agent.fyi, Instantacess, Dialplatform and Fakerean families from top left to bottom right. It can be seen that images from the same family exhibit similarity. In this paper, we use Nataraj's method to visualize malware binary files and propose a new method to extract effective texture feature representations which achieves higher accuracy.

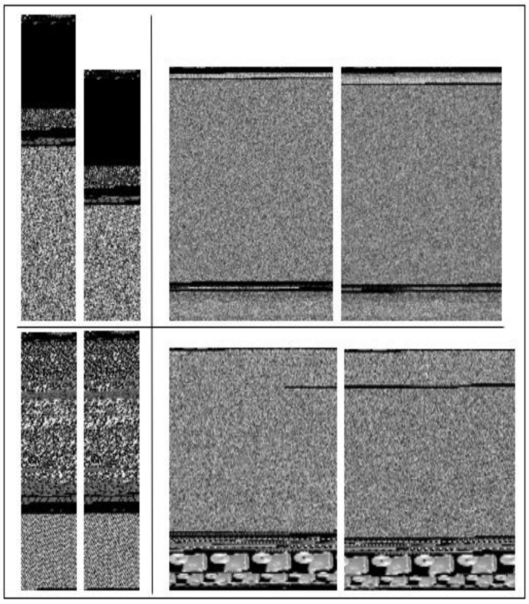


Figure 1: Examples of malware images.

B. MULTI-LAYER LEARNING FRAMEWORK

After converting binary files into images, image-based descriptors are used for classification. We first introduce the symbols used as follows.

Suppose a training malware image set $\mathcal{D} = \{d_{ij}\}$ contains images that belong to n families ($i = 1, \ldots, n, j = 1, 2, \ldots, N_i$ where N_i is the number of malware images in the i^{th} malware family). Given a new malware image, the problem is to classify it into its correct family. Let f_{ij} be the image feature for d_{ij}, which can be global image features (such as GIST descriptors) defined on the image as a whole, or local features (such as LBP or dense SIFT) defined in local neighborhoods around some pixels. Since variants of the same malware family may have similar code appearing in different locations, global features are unable to handle such cases effectively. Local features are more robust in such cases but can be more sensitive to small changes of code.

In this paper, we present a bag-of-visual-words (BoVW) model, with the basic idea originally from the bag-of-words (BoW) model in natural language processing. For the BoW model, a dictionary is first built that contains all the words (excluding stop words) from the collection of documents. Each document is then compactly represented using a vector with each element representing the number of times the corresponding word appears in the document. The BoW model greatly simplifies the representation, as neither the order of words in the document, nor the contextual relationships between words are taken into account. Although such loss of information is most criticized for natural language processing, we argue that in our problem of image-based malware classification, ignoring the order of visual words is indeed more beneficial, since it extracts useful malware image features and handles the differences among variants of the malware with interference information.

Thus we propose to use local features, along with a new multi-layer learning framework based on bag-of-visual-words (BoVW) to improve the robustness of malware image features. The first layer is local feature extraction where existing local texture descriptors can be used. The second layer is local feature descriptor grouping. The third layer extracts representative features from groups in the second layer, and the final layer produces the general feature representation of the image. The pipeline is illustrated in Figure 2.

Layer 1 (Local Feature Extraction): Features represent characteristics of texture images, such as frequent pattern occurrences. Our framework works with different local features. In this paper, we consider two typical features, namely local binary patterns (LBP) and dense SIFT. We briefly describe them below for completeness.

Definition 1: LBP (Local Binary Patterns) turns a local center pixel grayscale value into a binary pattern that encodes the relationship of the pixel with its local neighborhood. Each neighboring pixel is set to 1 or 0 according to whether the grayscale value of the pixel is larger than the value of the central pixel [21]. For a given malware image d_{ij}, f_{ij} is a collection of LBP_c value for each center pixel c. LBP_c is an integer defined as

$$LBP_c = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{1}$$

where g_c and g_p are the grayscale values of the center pixel c and its p^{th} neighbor, P is the number of neighboring pixels and

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases}$$

which provides a binary output (0 or 1). f_{ij} is a matrix, which presents the local features.

$$f_{ij} = \bigcup_{t=1}^{T} LBP_{p_t} \tag{2}$$

where $t = 1, 2, \ldots, T$, T is the number of patch centers and p_t is a patch center.

Our experiments show that the neighborhood radius and number of neighbor pixels have little influence on malware recognition. Thus, we choose radius $R = 1$ (i.e. 3×3 windows), comparing the grayscale value of the central pixel with its 8 neighboring pixels. In this case $P = 8$, so $LBP_c \in [0, 255]$.

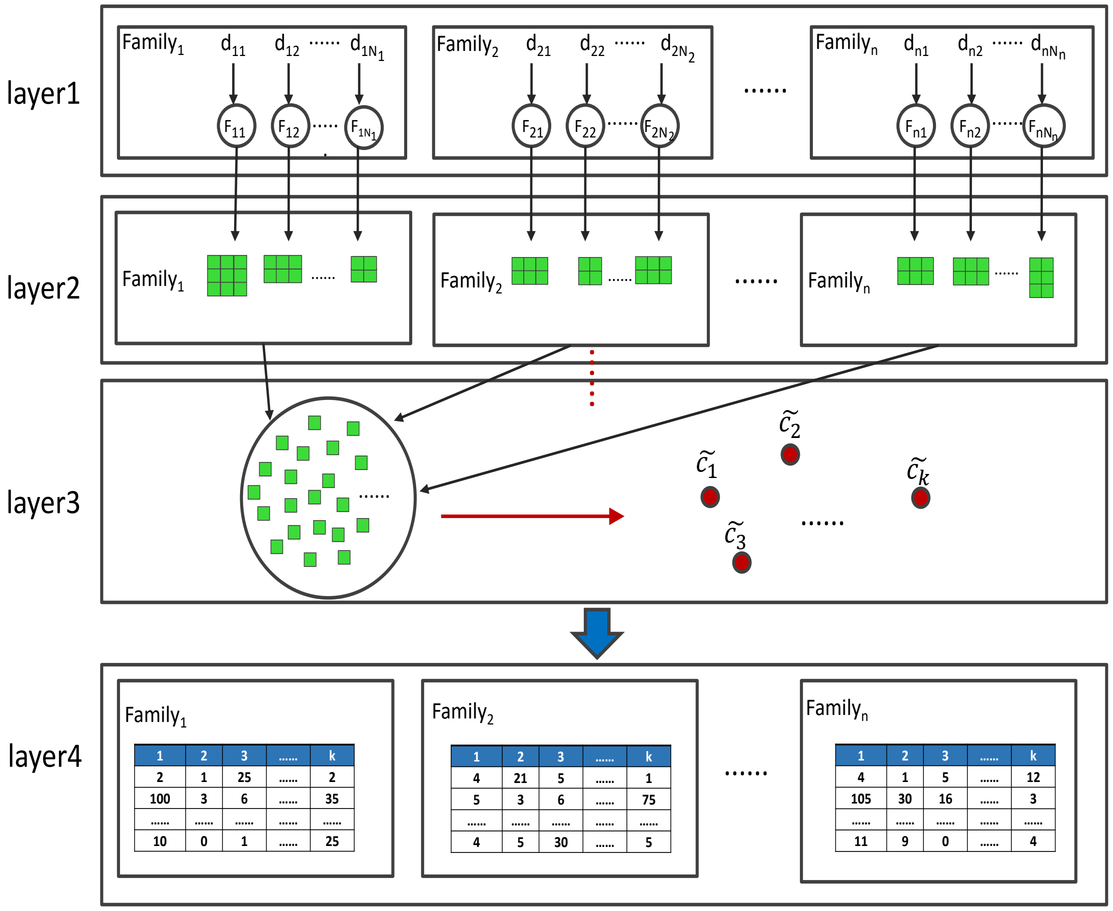
Definition 2: Dense SIFT calculates a SIFT descriptor determined by Lowe's algorithm at every location [22]–[24]. It collects features at each location and scale in an image, which helps increases recognition accuracy. It splits an image into small patches, and each patch is further spilt into smaller bins. The feature is then computed as gradient magnitude histograms in 8 orientations of bins. As the sliding window moves, it computes gradient histograms of each local neighborhood of the image. Finally, it obtains the image feature descriptors using cascaded connection functions.

Layer 2 (Feature Grouping): Although direct use of global features and local features can reflect overall structures of malware images, sometimes, they may not be reliable or robust. In malware code, the position of some code can be changed and some nonsensical code is often added in the files. Therefore, some distinctive local features, not necessarily in the same location, can be essential for classification. To make the features more robust, on the second layer, we split an image into many blocks of features, with each block containing $m \times m$ centers ($m = 16$ is used in our experiments). Denote by $x_{u,v}$ the local feature at relative position (u, v) in the block, the description for each feature block X is given in Equation 3:

$$X = \begin{bmatrix} x_{11} & \ldots & x_{1m} \\ \vdots & \vdots & \vdots \\ x_{m1} & \ldots & x_{mm} \end{bmatrix} \tag{3}$$

Denote by $\mathcal{X} = \{X_i\}$ the set of feature blocks.

Layer 3 (Representative Feature Selection): We cluster all the feature blocks from all the training images into k centers $\tilde{c}_i$ $(i = 1, 2, \ldots, k)$ using k-means clustering, which are considered as visual words. This provides an effective way


layer1
Family$_1$ d_{11} d_{12} …… d_{1N_1}
F_{11} F_{12} …… F_{1N_1}
Family$_2$ d_{21} d_{22} …… d_{2N_2}
F_{21} F_{22} …… F_{2N_2}
……
Family$_n$ d_{n1} d_{n2} …… d_{nN_n}
F_{n1} F_{n2} …… F_{nN_n}
layer2
Family$_1$
Family$_2$
Family$_n$
layer3
$\tilde{c}_1$
$\tilde{c}_2$
$\tilde{c}_3$
$\tilde{c}_k$
layer4
Family$_1$
Family$_2$
Family$_n$

1	2	3	……	k
2	1	25	……	2
100	3	6	……	35
……	……	……	……	……
10	0	1	……	25

1	2	3	……	k
4	21	5	……	1
5	3	6	……	75
……	……	……	……	……
4	5	30	……	5

1	2	3	……	k
4	1	5	……	12
105	30	16	……	3
……	……	……	……	……
11	9	0	……	4

Figure 2: A multi-layer feature extraction model.

to summarize block descriptors and suppress the impact of noise and outliers in the feature space.

$$\left\{\begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \vdots & \vdots \\ x_{m1} & \dots & x_{mm} \end{bmatrix}\right\} \Rightarrow \begin{bmatrix} \tilde{c}_1 \\ \vdots \\ \tilde{c}_k \end{bmatrix}$$

Since we are only concerned with obtaining a representative set of feature blocks, we randomly select a small subset (0.5% in our experiments) of blocks for clustering. This makes the computation much more efficient while maintaining the performance.

Layer 4 (BoVW Feature Descriptor Representation). The purpose of this layer is to obtain a robust feature representation for malware images, suppressing unreliable features. Following a BoVW model, for each block $X_i \in \mathcal{X}$ in the input image, we compute the Euclidean distance between it and each cluster center $\tilde{c}_j$. The center with the minimum distance is selected and decides the cluster the block belongs to. The histogram of all the chosen clusters forms the feature vector v of the image:

$$v = hist\left(\bigcup_{X_i \in \mathcal{X}} \operatorname{argmin}_j \parallel \tilde{c}_j - X_i \parallel\right) \tag{4}$$

where $hist(\cdot)$ is the histogram operator. v can be viewed as a general feature descriptor and used for classification. For a given test image, we apply Layers 1 and 2 to obtain feature blocks, and Layer 4 to calculate the global feature representation. Layer 3 is only needed in the training stage.

More precisely, taking LBP as local features, the proposed method can be summarized by Algorithm 1 (one-off process only needed for training) and Algorithm 2 (feature extraction for an input image).

IV. EXPERIMENTS

To evaluate the performance of the multi-layer learning framework, comprehensive comparisons are made with well-known methods. Experiments are conducted on three malware image datasets: MalingA dataset used in Nataraj's paper [19], Nataraj's malware album on his personal website [3] (referred to as MalingB), and the malware dataset from the CNCERT Labs. MalingA has 25 families and 9,458 malware images. MalingB is larger, with 36 families and 12,278 malware images. The CNCERT dataset has 10 families and 15,000 malware binary files. Among these datasets, MalingA and MalingB are provided directly as malware images, and we convert malware binary files from the CNCERT dataset to images. In this paper, the block size when building the

Algorithm 1 (Training Stage, involving *Layers* 1-3): Obtain the feature blocks X of training malware images d_{ij} from the training set $\mathcal{D}$, and work out the cluster centers $\{\tilde{c}_k\}$.

```
Input: malware image training set $\mathcal{D}$:
Output: features $X$ with respect to training malware images
    $d_{ij}$, and cluster centers $\{\tilde{c}_k\}$.
  for each $i \in [1, n]$ do
    $//n$ : the number of malware families;
    for each $j \in [1, N_i]$ do
      $//N_i$: the number of training examples for the $i^{\text{th}}$
      family;
      Calculate local LBP features $f_{ij} = \bigcup LBP_{p_t}$;
      Split $f_{ij}$ into blocks using an $m \times m$ grid, each is $X_t$;
      Add $X_t$ to the feature set $\mathcal{X}$;
    end for
  end for
  Take a subset of $\mathcal{X}$ as $\tilde{\mathcal{X}}$;
  Work out cluster centers $\{\tilde{c}_k\} = Kmeans(\tilde{\mathcal{X}})$
```

Algorithm 2 (Feature Extraction Algorithm, involving *Layers* 1, 2 and 4): Calculate global feature vector v for an input image.

```
Input: malware image $i_t$:
Output: the global feature $v$ with respect to the histogram
    of cluster numbers.
  Use Layers 1 and 2 (see Algorithm 1) to obtain feature
  blocks $X$.
  Initialize the histogram $h : h_t = 0, t = 1, 2, \ldots, k$.
  for $X_j \in X$ do
    $j = \text{argmin}_j \parallel \tilde{c}_k - X_j \parallel$
    $h_j \Leftarrow h_j + 1$
  end for
  $v = \frac{h}{\sum_j h_j}$
```

BoVW model is 16×16, i.e. $m = 16$.

A. EXPERIMENTAL RESULTS ON MALINGA

We first performed experiments on the same dataset (MalingA) as used in [19]. Nataraj's method [19] extracts GIST features of malware images, and uses a KNN (K-nearest neighbor) classifier and a random forest (RF) classifier to classify the test data. The GIST feature is a global feature based on Gabor filters. Each malware image is split into a 4×4 grid, and it uses a steerable pyramid with 8 orientations and 4 scales to obtain several filtered images. A feature is obtained by computing the absolute average deviation of transformed values from the mean within a small window of the filtered images. The GIST features are then obtained by cascade connection of those features.

To provide a fair comparison, we perform 10-fold cross validation and randomly run the algorithms for 10 times and report the average accuracy for classification of images in the test set. Table 1 gives the results of GIST [19] and the proposed method (multi-layer LBP using LBP as local descriptors, block size 16×16). In this experiment, we set the number of cluster centers $k = 100$. We also compare with baseline LBP and Dense SIFT without our proposed block-based multi-layer approach.

For KNN, we report the performance using KNN with $K = 2$, and random forest with the number of trees n_{trees} set to 25 as empirically these give better performance. According to Table 1, for MalingA, using GIST [19] features obtains the best accuracy of 0.98, which is identical to Nataraj's reported result [19]. Using the proposed method (multi-layer LBP and multi-layer dense SIFT) in the same dataset, it can get the best accuracy of 0.99 which is a little better than GIST features, but reasonable given the accuracy is already high. These methods also give similar performance with the RF classifier.

B. EXPERIMENTAL RESULTS ON MALINGB

MalingB is a larger dataset with more malware images and families. In order to compare the proposed method with original global and local descriptors, we report the results of the new BoVW model (multi-layer LBP and multi-layer dense SIFT) as well as results using GIST [19], baseline LBP and dense SIFT (as shown in definition 1 and definition 2). The testing results are shown in Table 2. Again we set the number of cluster centers $k = 100$, and the number of trees for the random forest (RF) classifier to 25.

As can be seen, the performance of RF and KNN are similar. The traditional GIST [19] feature in this case only achieves 0.910 accuracy (KNN where $K = 1$) and 0.883 accuracy (RF where $n_{trees} = 25$). When we use local features directly, the performance is similar with dense SIFT producing slightly better result (0.936 for KNN with $K = 1$ and 0.926 for RF). On the contrary, our multi-layer method performs much better, with multi-layer dense SIFT achieving 0.974 and multi-layer LBP achieving 0.970, which are much better than GIST [19], dense SIFT and LBP. In addition, our new method has got 0.966 accuracy using an RF classifier which is also better than alternative methods. Comparing Table 2 with Table 1, the results of GIST [19], dense SIFT and LBP methods are not satisfactory on the MalingB dataset whereas our new method has much more stable performance. We further investigate why the existing global method (GIST [19]) performs well on MalingA but poorly on MalingB. We can find that malware files are sometimes of substantially different file sizes and some malware images in the same family may include different icons (e.g. the Benign family as shown in Figure 3 (left)). Moreover, some malware images in the same family have very different textures (e.g. the Luder family as shown Figure 3 (right)). Those more difficult cases are not included in MalingA but appear in other more challenging datasets. That is why GIST [19], dense SIFT and LBP features are worse than the proposed method. For further analyzing the results, we perform the confusion analysis in the following subsection.

Table 1: Comparison of classification results on the MalingA dataset.

Classifier	KNN ($K = 2$)	RF ($n_{trees} = 25$)
GIST [19]	0.980	0.990
Dense SIFT	0.978	0.982
LBP	0.976	0.986
Multi-layer dense SIFT	0.990	0.987
Multi-layer LBP	0.981	0.989

Table 2: Comparison of classification results on the MalingB dataset.

Classifier	KNN ($K = 1$)	RF ($n_{trees} = 25$)
GIST [19]	0.910	0.883
Dense SIFT	0.936	0.926
LBP	0.877	0.901
Multi-layer dense SIFT	0.974	0.966
Multi-layer LBP	0.970	0.966

Table 3: Details of the MalingB-sub dataset.

family name	image numbers
Autorun.K	95
Benign	365
Fakerean	381
Luder.B	509
Obfuscator.AD	142
Skintrim.N	80
Virut.A	133
Virut.AC	269
Virut.AK	571

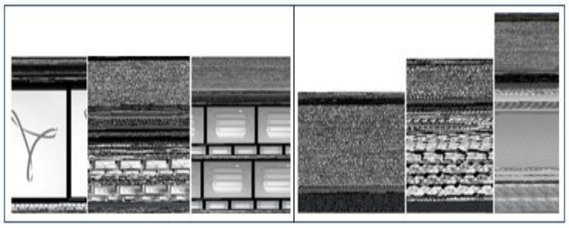


Figure 3: Two examples of challenging malware families in MalingB.

C. CONFUSION ANALYSIS

To better analyze the behavior of different methods, we first select 9 families, 2,545 malware images from MalingB not included in MalingA to form a subset (which we call MalingB-sub), as shown in Table 3. MalingB-sub includes malware families that are more confusing.

We perform experiments using GIST [19] and multi-layer LBP features with an RF classifier. The results with KNN are similar, which are omitted to avoid repetition. In the experiment, we perform 10-fold cross validation and randomly run the algorithms for 10 times and report every result and average accuracy using RF ($n_{trees} = 25$) in Table 4. As can be seen, multi-layer LBP is much better than GIST [19]. In order to show the confusion details, we give the confusion matrix when it achieves the best result 0.914 using the GIST [19] feature, as shown in Table 5. We can find that the most confusing families are the Benign family and the Luder.B family. In addition, 11.8% instances of Virtut.A family are misclassified into Virut.AK. There is also a 2.6% error rate in the classification of the Fakerean family. In comparison, the confusion matrix of the proposed method with multi-layer LBP is shown in Table 6 when it achieves 0.945 accuracy. A comparison of Table 5 and Table 6 shows that Fakerean and Virtut.A families are classified completely correctly with the proposed method. The results of Benign and Luder.B family classification are also improved, which demonstrates the robustness of the proposed method.

D. COMPARISON WITH HAN'S METHOD

We also compare the proposed method with Han's method [25] using a random forest classifier with n_{trees} set to 25. This method is based on Nataraj's image-based approach but uses entry graph based features instead. On the MalingA dataset, it can get the accuracy 0.985 with appropriate threshold value 0.75, which is consistent with their paper and close to the performance of the proposed method (0.99). When applied to more challenging MalingB dataset, as shown in Table 7, the best classification accuracy of Han's method [25] is only 0.908 (with the threshold set to 0.75), which is clearly worse than our results.

E. EXPERIMENTAL RESULTS ON THE CNCERT DATASET

The third dataset is from CNCERT Labs, which contains a set of 10 families, 15,000 binary files and corresponding assembly files. We turn the binary files into malware images according to Narataj's method [19]. In the experiment, we compare GIST [19] with multi-layer LBP descriptor using both a KNN classifier and an RF classifier. The results are shown in Table 8. The method [19] achieves better performance with the RF classifier, and the average accuracy obtained is 0.929 ($n_{trees} = 25$). Our multi-layer LBP method achieves better average accuracy of 0.935. The accuracy of Han's method is 0.901 [25] with the threshold value set to 0.75.

F. DISCUSSIONS ABOUT PARAMETER SETTINGS

We perform further experiments to analyze the performance of the proposed method with different parameter settings. In the second layer, the features need to be split into $m \times m$ blocks, where the number of m is very important. If m is too

Table 4: The classification performance on the MalingB-sub dataset.

Classifier	RF ($n_{trees}=25$)										
No.	1	2	3	4	5	6	7	8	9	10	average accuracy
GIST [19]	0.914	0.886	0.910	0.867	0.906	0.882	0.906	0.894	0.906	0.89	0.896
Multi-layer LBP	0.951	0.948	0.95	0.953	0.949	0.938	0.95	0.949	0.952	0.947	0.9488

Table 5: The confusion matrix on MalingB-sub with the GIST [19] feature (accuracy $= 0.914$)

malware family	Autorun.K	Benign	Fakerean	Luder.B	Obfuscator.AD	Skintrim.N	Virut.A	Virut.AC	Virut.AK
Autorun.K	1	0	0	0	0	0	0	0	0
Benign	0	0.667	0	0.303	0	0	0	0	0.03
Fakerean	0	0	0.974	0	0	0	0	0	0.026
Luder.B	0	0.102	0	0.837	0	0	0	0	0.061
Obfuscator.AD	0	0	0	0	1	0	0	0	0
Skintrim.N	0	0	0	0	0	1	0	0	0
Virut.A	0	0	0	0	0	0	0.882	0	0.118
Virut.AC	0	0	0	0	0	0	0	1	0
Virut.AK	0	0	0	0	0	0	0	0	1

Table 6: The confusion matrix on MalingB-sub with our multi-layer LBP feature (accuracy $= 0.945$)

malware family	Autorun.K	Benign	Fakerean	Luder.B	Obfuscator.AD	Skintrim.N	Virut.A	Virut.AC	Virut.AK
Autorun.K	1	0	0	0	0	0	0	0	0
Benign	0	0.73	0	0.27	0	0	0	0.027	0.027
Fakerean	0	0	1	0	0	0	0	0	0
Luder.B	0	0.078	0	0.922	0	0	0	0	0
Obfuscator.AD	0	0	0	0	1	0	0	0	0
Skintrim.N	0	0	0	0	0	1	0	0	0
Virut.A	0	0	0	0	0	0	1	0	0
Virut.AC	0	0	0	0	0	0	0	1	0
Virut.AK	0	0	0	0	0	0	0	0	1

Table 7: Comparison of three methods on MailingB dataset

Method	Han's method (with given threshold)				Proposed method		Nataraj's method (GIST [19])
	0.5	0.75	0.8	0.9	Multi-layer SIFT	Multi-layer LBP	
Accuracy	0.763	0.908	0.882	0.796	0.966	0.966	0.910

Table 8: Classification results on the CNCERT dataset.

Classifier	KNN ($K=2$)	RF ($n_{trees}=25$)
GIST [19]	0.918	0.929
Dense SIFT	0.921	0.926
LBP	0.882	0.901
Multi-layer dense SIFT	0.938	0.930
Multi-layer LBP	0.932	0.935
Han [25]	0.901 (threshold=0.75)	

small, the number of blocks is huge. Therefore it takes long time to obtain the feature descriptors and cluster those blocks. On the contrary if m is too big, the BoVM features are not as robust. In the MailingB dataset, we use 8×8, 16×16 and 32×32 block sizes respectively (see Table 9). 8×8 blocks are too small, and the images are split into many blocks. They consume so much time to extract features that we have to stop it after 5 days. 32×32 blocks are too big, and the results are poor. The results using 16×16 blocks are better and used in all the experiments.

We further analyze the performance of the RF classifier w.r.t. its parameter. Figure 4 presents curves showing the classification accuracies of the original method [19] and our learning framework using different numbers of trees in the Random Forest classifier on MalingB. We test 10, 15, 20 and 25 trees and show the average accuracies in Figure 4. The blue curve presents the GIST method [19] and the other curves present multi-layer LBP and multi-layer dense SIFT respectively. It can be seen that the red and green curves are consistently higher than the blue one, which means that the multi-layer learning framework outperforms the GISTmethod [19]. The performance is generally stable with changing number of trees, and we use 25 trees for fairer comparison as different methods work generally well.

Table 9: Classification results using different window size on the MailingB dataset

classifier	KNN ($K = 2$)		multi-layer LBP ($n_{trees} = 25$)	
	16×16	32×32	16×16	32×32
Multi-layer LBP	0.970	0.9054	0.966	0.9326

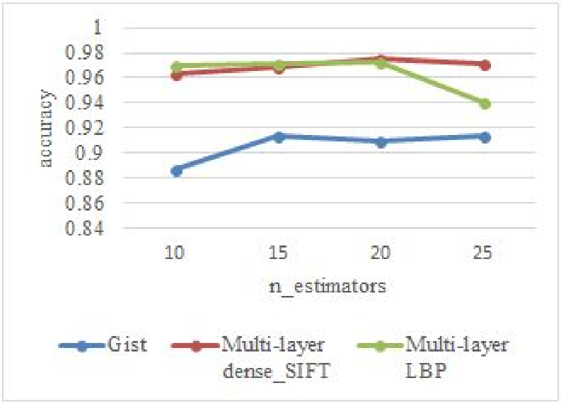


Figure 4: Classification accuracies of GIST [19] and the proposed method with different setting (n_{trees}) using the RF classifier.

Using the proposed method, on the third layer, it needs to cluster the feature blocks. For k-means clustering, the number of clusters needs to be specified. We perform experiments on the MalingB dataset with varying cluster number k (see Figure 5). It shows four curves about multi-layer dense SIFT classification using RF with 25 trees when the number of clusters is chosen as 80, 100, 200 and 500. It clearly demonstrates that the best performance is achieved with $k = 100$, so this is used by default in our experiments.

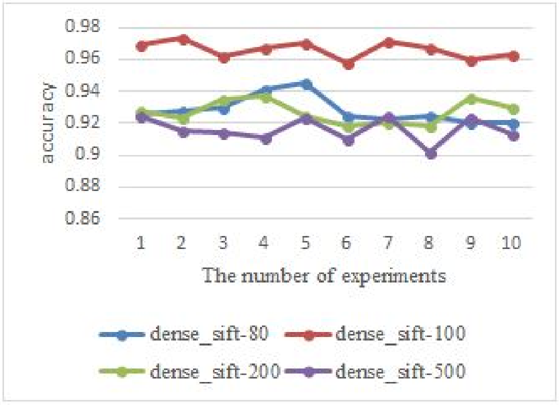


Figure 5: The accuracies of multi-layer dense SIFT with different cluster numbers.

G. DISCUSSION ABOUT SPLITTING STRATEGY

In Section III, the features are extracted firstly on the first layer, which are then split into blocks on the second layer. An alternative approach is to split images into blocks before calculating LBP features. We test this and show comparative results in Figure 6. The blue curve is according to the multi-layer learning model in Section III (labeled as "first_lbp_then_block") and the blue one is the result of classification using the features obtained by switching the order of the first two layers (labeled as "first_block_then_lbp"). It clearly shows the benefit of performing LBP first, followed by putting features into blocks.

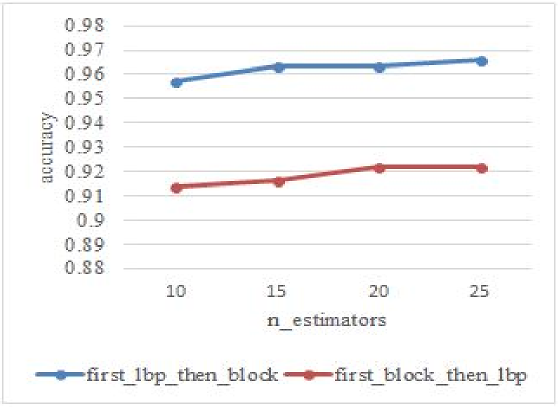


Figure 6: The comparison of two multi-layer models

To investigate why this happens, according to the LBP descriptor, there are some pixels on the border where feature descriptors cannot be calculated. If we use the multi-layer learning framework to obtain the features of each malware image, the computed area is illustrated in Figure 7(left). If we switch the steps of the first two layers, the pixels which can be computed are shown in Figure 7(right), which contain significant gaps between blocks with useful information. This explains the significant performance drop in Figure 6. Since dense SIFT uses a sliding window to compute all pixel features, it does not suffer from this problem.

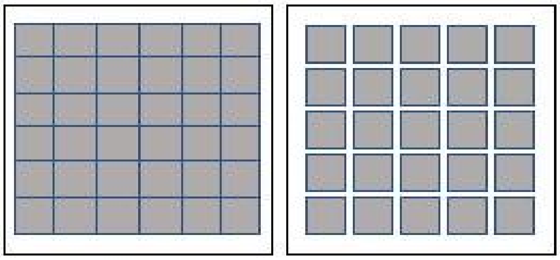


Figure 7: The analysis of two multi-layer models

V. CONCLUSION

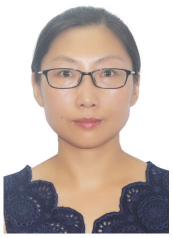
In this paper, we propose a multi-layer learning framework based on a bag-of-visual-words (BoVW) model to obtain feature descriptors of malware images. The model can obtain more robust features and achieve better classification accuracies even for more challenging datasets, compared with other methods. One limitation of our method is its higher computational cost. In terms of time consumption, Han's method is faster than Nataraj's method, which is further faster than our method, because the proposed method needs to split malware images into blocks and further cluster these block features. For example, with our current unoptimized code, it takes 3 days to perform the training and testing on the entire MalingB dataset. In order to avoid malware detection, malware authors may pack, obfuscate or encrypt executables. If the malware execute files are packed, our multi-layer learning framework will still produce consistent results. However, if the malware is obfuscated or encrypted, the proposed method can be interfered. We will resolve these problems in the future.

ACKNOWLEDGMENT

We would like to thank Professor Ralph Martin, who gave helpful advice during the experiments of this paper. He has now retired from Cardiff University.

References

[1] AV-Test: http://www.av-test.org.
[2] V.S. Sathyanarayan, P. Kohli, B. Bruhadeshwar, "Signature generation and detection of malware families," *Proceedings of Australasian Conference on Information Security and Privacy*, July.2008, pp. 336–349.
[3] M.F.B. Abbas, T. Srikanthan, "Low-complexity signature-based malware detection for IoT devices," *Proceedings of Applications and Techniques in Information Security*, pp. 181–189, June. 2017.
[4] A. Mohaisen, O. Alrawi, M. Mohaisen, "AMAL: high-fidelity, behavior-based automated malware analysis and classification," *Computers and Security*, vol. 52, pp. 251–266, July. 2015.
[5] H.S. Galal, Y.B. Mahdy, M.A. Atiea, "Behavior-based features model for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 59–67, May. 2016.
[6] K.S. Han, B.J. Kang, E.G. Im, "Malware classification using instruction frequencies," *Proceedings of ACM Research in Applied Computation Symposium*, pp. 298–300, January. 2011.
[7] P. Natani, D. Vidyarthi, "Malware detection using API function frequency with ensemble based classifier," *Proceedings of Security in Computing and Communications*, vol. 377, pp. 378–388, November, 2013.
[8] Y. Ye, T. Li, Y. Chen, et al., "Automatic malware categorization using cluster ensemble," *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining*, July, 2010, pp. 95–104,
[9] E. Kirda, C. Kruegel, G. Banks, et al., "Behavior-based spyware detection," *Proceedings of the 15th Conference on USENIX Security Symposium*, July, 2006, pp. 273–288.
[10] S. Cesare, X. Yang, "A fast flow graph based classification system for packed and polymorphic malware on the end host," *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*, April, 2010, pp. 721–728.
[11] I. Santos, F. Brezo, J. Nieves, et al., "Idea: opcode-sequence based malware detection," *Proceedings of Engineering Secure Software and Systems*, vol. 5965, no. 4, pp. 35–43, 2010.
[12] I. Santos, F. Brezo, B.Sanz, et al., "Using opcode sequences in single-class learning to detect unknown malware," *IET Information Security*, vol. 5, no. 4, pp. 220–227, December. 2011.
[13] P. O'kane, S. Sezer, K. Mclanghlin, "Detecting obfuscated malware using reduced opcode set and optimized runtime trace," *Security Informatics*, vol. 5, no. 1, pp. 2–13, May. 2016.
[14] S. Cesare, X. Yang, "Classification of malware using structured control flow," *Proceedings of the 8th Australasian Symposium on Parallel and Distributed Computing*, vol. 107, pp. 61–67, January. 2010.
[15] P. Trinius, T Holz, J Gobel, et al., "Visual analysis of malware behavior using treemaps and thread graphs," *Proceedings of the 6th International Workshop on Visualization for Cyber Security*, January, 2010, pp. 33–38.
[16] J. Saxe, D. Mentis, C. Greamo, "Visualization of shared system call sequence relationships in large malware corpora," *Proceedings of the 9th International Symposium on Visualization for Cyber Security*, October, 2012, pp. 33–40.
[17] X. Hu, T.C. Chiueh, G.S. Kang, "Large-scale malware indexing using function-call graphs," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, November, 2009, pp. 611–620.
[18] G. Conti, S. Bratus, A. Shubina, et al., "Automated mapping of large binary objects using primitive fragment type classification," *Digital Investigation*, vol. 7, pp. S3–S12, August. 2010.
[19] L. Nataraj, S. Karthikeyan, G. Jacob, et al., "Malware images: visualization and automatic classification," *Proceedings of International Symposium on Visualization for Cyber Security*,July, 2011, pp. 4:1–7.
[20] L. Nataraj, "Malware Images," http://vision.ece.ucsb.edu/~lakshman/malware_images/album.
[21] T. Ojala, M. Pietikinen, D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, November, 1994, pp. 582–585.
[22] D.G. Lowe, "Object recognition from local scale-invariant features," *IEEE International Conference on Computer Vision*, September, 1999, vol. 2, pp. 1150–1157.
[23] D.G. Lowe, "Distinctive image features from scale-invariant key points," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–111, November. 2004.
[24] A. Oliva, A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, May. 2001.
[25] K.S. Han, J.H. Lim, B.J. Kang, et al., "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, no. 1, pp. 1–14, February. 2015.
[26] K.S. Han, B.J. Kang, E.G. Im, "Malware Analysis Using Visualized Image Matrices," *The Scientific World Journal*, pp. 1–15, July. 2014.
[27] B. Kolosnjaji, A. Zarras, G. Webster, et al., "Deep learning for classification of malware system call sequences," *Proceedings of AI 2016: Advances in Artificial Intelligence. Australasian Joint Conference on Artificial Intelligence*, November, 2016, vol 9992, pp. 137–149.
[28] E. David, N.S.Netanyahu, "DeepSign: deep learning for automatic malware signature generation and classification," *Proceedings of International Joint Conference on Neural Networks*, October, 2015, pp. 1–8.
[29] T. Shibahara, T. Yagi, M. Akiyama, et al., "Efficient dynamic malware analysis based on network behavior using deep learning," *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, February, 2017, pp. 18–7.



Yashu Liu is a Ph.D student in Beijing Jiaotong University, Beijing, China. Her research interests include data ming. Now, she majors in detection and classification of malware.

Yu-Kun Lai received the bachelor's and Ph.D. degrees in computer science from Tsinghua University, China in 2003 and 2008, respectively. He is currently a Reader in the School of Computer Science and Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing, and computer vision. He is on the Editorial Board of *The Visual Computer*.

Zhihai Wang is a professor in Beijing Jiaotong University, Beijing, China. He is Yashu's supervisor. His research interests include data mining and business intelligence, database and data warehouse, machine learning and computational intelligence.

Hanbing Yan obtained the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China in 2006. He is now working in the National Computer Network Emergency Response Technical Team/Coordination Center of China. His research interests include cyber security, image-spam analysis, computer graphics and image processing.