File Entropy Signal Analysis Combined With Wavelet Decomposition for Malware Classification

HUI GUO1, SHUGUANG HUANG1, CHENG HUANG2, (Member, IEEE), ZULIE PAN1, MIN ZHANG1, AND FAN SHI1

1College of Electronic Engineering, National University of Defense Technology, Hefei 230011, China
2College of Cybersecurity, Sichuan University, Chengdu 610065, China

Corresponding author: Zulie Pan (panzulie@126.com)

This work was supported in part by the Laboratory of Network Security, Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation, College of Electronic Engineering, National University of Defense Technology, and in part by the Provincial Natural Science Foundation of Anhui, under Grant 1908085QF291.

ABSTRACT With the rapid development of the Internet, malware variants have increased exponentially, which poses a key threat to cyber security. Persistent efforts have been made to classify malware variants, but there are still many challenges, including the incapacity to deal with various malware variants belonging to similar families, the problem of time and resource consuming, etc. This paper proposes a novel method, called Malware Entropy Sequences Reflect the Family (MESRF), to improve the classification of malware based on the entropy sequences features. In prior research, entropy demonstrated good performance in many areas. First, the global features of the signals were extracted from the entropy sequences by some statistical methods. Next, some local features (i.e. structural entropy features) are extracted based on the discrete wavelet decomposition algorithm and vectorized by the Bag-of-words model, endowing it the high accuracy of malware classification. To evaluate our method, we conducted numerous experiments on the malware datasets with more than 20,000 samples. Through experiments, MESRF showed superiority comparing with other malware classification models, and the accuracy and ROC of the method even could reach 99.83% and 99.98% respectively on the malimg dataset.

INDEX TERMS Malware classification, entropy sequences, signal processing, Haar wavelet transform, support vector machine.

I. INTRODUCTION

With the rapid development of the Internet, malware variants have increased exponentially in recent years, posing a serious threat to cyber security. According to the Symantec report [1], 246,002,762 new malware variants were monitored in 2018, and the number of new malware variants has exceeded a billion in the past three years. Hackers are more inclined to adopt minor changes, packing, encryption and other technologies, combined with the original malicious code, to create new malware variants for propagation. Therefore, the rapid identification and classification of malware variants can effectively assist researchers in grasping their characteristics and have significant research value.

Malware detection methods are mainly composed of three types of methods: signature-based detection, static detection and dynamic detection. Signature-based detection works mainly by extracting signatures from malware and building malware library. Static detection disassembles the malware and analyzes its opcodes, static API sequences and execution logic without performing it. Dynamic detection extracts the behavior feature (e.g. network activity, file operations and system call) of malware by executing the malicious sample in a virtual environment. The Static and dynamic detection methods belong to features-based detection methods. In the process of these methods, the static information or behavioral features of the malware are extracted, and then the malware samples are classified based on these features combined with machine learning algorithms. In recent years, data mining methods are often used to analyze the feature of malware [2]. This approach has become the mainstream method of...
malware detection because of its high efficiency and better performance compared with traditional methods.

A. NEED FOR THIS STUDY

However, signature-based detection often fails to detect new malware variants. And feature-based methods are often disturbed. Obfuscation technology could modify malware, increase the difficulty of code analysis, and reduce the effectiveness of static detection. Dynamic feature detection is more robust, but it still could be challenged by different kinds of countermeasures, producing unreliable results. Furthermore, the time and resources consuming for malware executing are often expensive.

Recently, rather than focusing on non-textured features for malware classification, several scholars [3]–[7] proposed new methods based on binary texture features of malware. They transformed the raw bytes of malware samples into two-dimensional vectors. Then the malware texture features were extracted for classification. Besides, these two different types of feature design strategies are studied comparatively for malware detection [5]. The results showed that the texture-based feature provided performance roughly equivalent to dynamic detection methods, but in less time.

Different malware detection methods all have their pros and cons. The texture-based detection methods can handle some code obfuscation problems, and provide higher efficiency relative to dynamic detection, but it still has some limitations. The malware variant has many homologous parts with its ancestors, which makes it have a strong correlation in structural characteristics. However, the malware samples of similar families also have some correlations with each other, which will cause disruption to the classifier and reduce accuracy. The challenge for building malware classification models is to find an effective means for distinguishing malware samples of similar families.

B. MAJOR CONTRIBUTIONS OF THE STUDY

To address the above challenges, we propose a new method to classify malware based on its another binary structure feature (i.e., entropy distribution). The entropy, as a measurement of uncertainty, has been proven to be effective in the compressed malware and malicious documents detection in recent researches [8]–[11]. Inspired by these studies, we find that the correlation between malware variants may lead to similar entropy distribution of malware samples of the same family. Therefore, we compute the entropy sequences of malware and represent them as a signal. First, the global features of the signals were extracted from the entropy sequences by some statistical methods. Next, some local features (i.e., structural entropy features) are extracted based on the discrete wavelet decomposition algorithm and vectorized by the Bag-of-words model. At last, we train a machine-learning classifier based on the global and local features.

We deployed a prototype system, named MESRF (Malware Entropy Sequences Reflect the Family), to improve the classification of malware variants. To evaluate the proposed framework, we conducted a cross-validation experiment on custom datasets. The experimental results showed that MESRF could better solve the problem of poor classification of similar family malware in existing research. The accuracy and ROC of our method even could reach 99.83% and 99.98% on the maling dataset [3].

Overall, this paper offered the following contributions:

1) A new malware classification method was proposed based on binary entropy features. It converted the raw bytes of malware into entropy sequence and extracted its structural features based on the signal processing technique to classify the malware.

2) We implemented MESRF, a framework extracting features from malware and combined them with a classifier to classify malware variants. This framework can enhance the ability to classify malware variants of similar families.

3) Extensive experimental results demonstrated that MESRF could better solve the problem of poor classification of similar family malware compared with related methods.

The remainder of this paper is structured as follows. In Section II, related work is reviewed. In Section III, we describe the details of the methods for malware classification based on the entropy sequence. In section IV, experimental evaluations of the methods are provided. In section V, we describe some limitations and discussions about the method. The conclusions and future work are given in section VI.

II. RELATED WORK

In this section, the related research about malware is presented, including malware detection based on the signature and feature analysis, malware detection based on texture features and malware detection based on entropy features.

A. MALWARE DETECTION BASED ON SIGNATURE

Malware detection based on the signature is one of the earliest malware detection technologies. It still plays an important role to this day. The signature refers to the unique short code present in the malware, allowing researchers to determine whether the detected file is the target malicious file [12]. When this technology first appeared, the signatures usually needed to be manually generated and updated based on expert experience. When researchers encounter new malware, they need to analyze the malware, generate a signature, and then add it to the signature library. YARA [1] is a widely used framework that can assist users in generating signatures and patterns for specific malware. However, malware detection based on the signature would consume more resources and can only detect existing malware, which lags behind the detection of new malware variants. And the time window for typical malware to appear and be detected is

\[\text{http://virustotal.github.io/yara/}\]
about 54 days [12]. Malware detection based solely on signatures is no longer applicable in the current environment.

B. MALWARE DETECTION BASED ON STATIC FEATURES

The malware static detection technology extracts the static feature of malware, including byte stream, opcode, static API sequence, CPU register features, PE header features, import and export table, etc. to train a classifier for malware classification. Raff and Nicholas [13] trained a malware classifier based on byte stream data, Lempel-Ziv Jaccard Distance algorithm and K-nearest-neighbor (KNN) algorithm. Ahmadi et al. [14] processed malware byte stream, opcodes, PE file features, static API sequence, etc. based on the N-gram algorithm and trained malware classifiers combined with support vector machine (SVM) and random forest (RF) algorithms. Pai et al. [15] utilized k-means and expectation-maximization algorithms to cluster malware of different families based on opcodes. The static detection method is fast and efficient, but it is easily disturbed by code obfuscation techniques.

C. MALWARE DETECTION BASED ON DYNAMIC FEATURES

In recent years, many scholars have conducted malware analysis based on dynamic features. Researchers execute malware in a virtual environment or sandbox and extract behavioral features (e.g. run-time API sequence, network activity, and file operation) to train a classifier. It is more robust to obfuscation techniques. Dai et al. [16] extracted the API sequence, file operations and underlying hardware characteristics, etc., to classify malware based on the ensemble learning algorithm. Mohaisen et al. [17] obtained file operations, CPU register operations, and network communication by executing the malware in a virtual machine, and classify malware based on machine learning algorithms. Mohaisen et al. [17] and Kim et al. [18] extracted file operations, network activities, etc., and performed malware classification based on the similarity measures. These techniques improve the performance of malware detection, but they are still challenged by various countermeasures [5]. Furthermore, dynamic detection is resource-consuming because of the large amount of computation, making it unsuitable for large datasets.

D. MALWARE DETECTION BASED ON TEXTURE FEATURES

In recent years, many studies have utilized malware structure features. Zhang et al. [19] extract structure features from the opcode sequences of malware and classified malware based on machine learning algorithms. The result showed it could achieve good accuracy for a small training set. Han et al. [20] transformed the opcode sequences into images and extract texture features for malware classification. This method has achieved good detection performance on packers and encryption malware.

Different from the above studies, Nataraj et al. [3] proposed a new approach to extract texture features from the raw bytes of malware and classify malware based on machine learning algorithms. They transformed the raw bytes of malware samples into two-dimensional vectors and extracted malware texture features for classification. Compared with the static and dynamic analysis method, their method could provide more improved results [5]. Kosmidis and Kalionis [21] conducted a more in-depth study on the texture features of malware and tested the performance of different classifiers on the dataset based on machine learning algorithms. Based on these studies, Xiaofang et al. [22], Naeem et al. [6], [23] and Hashemi and Hamzeh [24] conducted more researches based on different texture features (e.g. SURF, LBP, DSIFT), and achieved better performance. With the development of deep learning algorithms, Cui et al. [4] proposed a method for malware classification based on malware raw bytes combined with convolutional neural networks. Tang et al. [7] proposed a method to solve the lacking data problem and improve the performance of malware few-shot classification based on the deep learning algorithm. Rezende et al. [25] utilized VGG16 network to extract features from the raw bytes, and classified the malware based transfer learning algorithms, which improved the classification performance. The texture-based detection methods can handle some code obfuscation problems, and provide better performance relative to some traditional methods, but it still has some limitations. At present, the classification accuracy of malware samples for similar families in research is relatively poor. The latest related methods combined with deep learning algorithms have partially alleviated the problem. But the problem is still not completely resolved, and the accuracy of classification needs to be further improved.

Moreover, the entropy distribution has also been proved to be effective in many fields of malware detection. Wojnowicz et al. [8] extracted entropy features of malware to distinguish malware from benign software, achieved efficient detection of parasitic malware. Bat-Erdene et al. [9] detected the packing algorithm of malware based on the entropy features. Liu et al. [10] detected the malicious documents based on entropy sequences. Canfora et al. [11] detected Android malware based on the entropy feature of the file structure, and achieved good results. Therefore, we consider that the entropy sequence of malware may be able to detect malware variants and classify malware.

III. APPROACH

A. OVERVIEW

First, we extract an entropy sequence from the raw bytes of malware. Next, to mine the features in the entropy, we represent the entropy sequence as a signal. Specifically, we obtain the global and local feature (i.e. structural features). For global features, we extract some static characteristics, such as length, mean value, maximum, standard deviation and minimum ratio, etc. For structural characteristics, we process the entropy signals by discrete wavelet decomposition algorithms and vectorize the local features based on the Bag-of-words model. An overview of our method is shown in Figure 1.
We extract the raw bytes of the malware and represent them as the entropy signals. Then, the global and local features are extracted based on the Haar Wavelet Transform algorithm and vectorized by the Bag-of-words model. At last, we train a machine learning-based classifier and identify the families of new malicious samples.

B. THE ENTROPY SEQUENCE OF MALWARE

To compute the structural entropy of the malware, the raw bytes of malware, is divided into continuously and disjoint blocks. In this paper, the value of the block size is set to 256, to ensure that the bytes can take all the values in 00h-FFh when computing the entropy value. If the size of the last block is not equal to 256, we fix it this way: if the size of the block can reach 128, we will fill it with zero. Otherwise, we will abandon it.

The entropy of each block is computed as follows:

\[ H(X) = -\sum_{i=0}^{255} p(x_i) \log_2 p(x_i) \] (1)

where \( x_i \) represents a specific raw byte value and \( p_i \) represents the probability (frequency) of this value in the block. We represent the entropy sequence as \( V = v_1, v_2, \ldots, v_n \), where \( n \) represents the block numbers of malware samples, \( v_i \) represents the entropy value of the block \( i \), and \( v_j \) ranges from zero to eight. When all bytes in the block are equal, the value of \( v_j \) is zero. If all the values in the block are different, the value of \( v_j \) is eight.

Some examples of the entropy distribution are shown in Figure 2. Figure 2(a) and 2(b) show the entropy distribution of the malware samples of family Rbot!gen. Figure 2(c) and 2(d) show the entropy distribution of the malware samples of family Adialer.C. The difference between the entropy sequences of the malicious samples of the two families is mainly as follows: static characteristics and distribution patterns. For each malware sample, we count several statistical characteristics of the entropy sequence. From the statistical characteristics, we can find that there are some similarities between samples of different malware families, but they also show some differences. The length, mean and maximum value of malware entropy sequences of different families are different. Moreover, it can be found from the figure that the malware samples of different families also show some differences in the distribution of the entropy sequence. As shown in Figure 2, at the end of the entropy sequence, the malware samples of family Adialer.C have larger values than the samples of family Rbot!gen. However, the statistical characteristics and distribution patterns exhibited by malware samples from the same family are very similar.

C. GLOBAL FEATURES

In this study, we first extract some static features of the entropy sequence as global features. The extracted static features can provide global information about the entropy sequences and can enhance the classifier’s ability to detect malware variants. The global features we extracted include the following seven items:

1. **Length**: The length of entropy sequence.
2. **Square Root**: Square root of the length of the entropy.
The wavelet function will undergo different translations and dilation. And we use the Wavelet transform to process entropy sequences and extract features. Wavelet transform has many different characteristics from the Fourier transform. Its advantage is that more details of the signal can be obtained through different scales of processing. It represents the power of a signal over different frequencies. We choose the Haar wavelet transform, an important wavelet transform method, to deal with our entropy sequence. This technology projects the signal onto a set of square waves with different heights, widths, and supporting structures (a subset of the non-zero domain of the signal) and could obtain the changes of the malware entropy sequences at different positions and scales. Therefore, we represent the entropy sequence as a signal and utilize Haar Wavelet Transform to extract the structure features.

In the wavelet transform, the father wavelet (scaling function) and the mother wavelet (wavelet function) play a major role in the analysis of the malware signals. The malware signal is processed by these two types of functions in the decomposition. The wavelet function $\psi_{HAAR}(t)$ and the scaling function $\phi_{HAAR}(t)$ the Haar wavelet transform is defined as follows:

$$\psi_{HAAR}(t) = \begin{cases} 1, & 0 \leq t < \frac{1}{2} \\ -1, & \frac{1}{2} \leq t < 1 \\ 0, & t < 0, t > 1 \end{cases}$$

(2)

$$\phi_{HAAR}(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & \text{otherwise} \end{cases}$$

(3)

Before signal decomposition, a series of transforming functions about mother wavelet and father wavelet need to be produced. In this process, the mother function and father function will undergo different translations and dilation. And the transforming functions are produced based on two important arguments (i.e. scaling and translating). The transforming functions are described as following equations.

$$\psi_{a,b}(t) = |a|^{-1/2} \psi\left(\frac{s-b}{a}\right)$$

(4)

$$\phi_{a,b}(t) = |a|^{-1/2} \phi\left(\frac{s-b}{a}\right)$$

(5)

where $a$ is scaling parameter and $b$ is the translating parameter. When $a=2j$ and $b=ia$, this set of function can be transformed into:

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - i)$$

(6)

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - i)$$

(7)

where $j$ is the scaling parameter, which represents the resolution level at different stages and $i$ is the position parameter. In the process of wavelet transformation, the entropy sequence signal is decomposed into a series of data characterizing the details of the sequence. Specifically, if we get an entropy signal $S(t)$ which consists of $T$ positions, we can get the father wavelet coefficients (i.e. approximate coefficient) of the signal combined with the set of scaling transforming functions, which is defined as follows:

$$a_{j,k}(t) = \{s, \phi_{j,k}\} = \sum_{i=1}^{T} s(t) \phi_{j,k}(t)$$

(8)

The father wavelet coefficients describe the inner character of the signal. Therefore, in this paper, we extract the father wavelet coefficients of the entropy signal as local features. The extracted approximation characteristics can provide some local information about the malware entropy sequences and improve the performance of malware classification. Besides, discrete wavelet decomposition allows for multiple decomposition analysis. At each turn of the decomposition, the previous father wavelet coefficients would be further processed into a more detailed approximate coefficient. Concretely, the process of decomposition is recursive and the signal of the entropy sequence $s(t)$ would be processed into a group of approximate coefficients $s_j(t)$. The process can be defined as follows:

$$s_{j+1}(t) = \{s_j(t), \phi_{j,k}\}$$

(9)

2) FEATURE VECTORIZATION BASED ON THE BAG-OF-WORDS MODEL

We could extract features based on the Haar Wavelet Transform algorithm, but how to vectorize features is a more important issue. In this paper, we convert the features into vectors by using the Bag-of-words model. The main procedure of the method is shown in Figure 3. It consists of two steps: (1) Codebook generation, (2) Histogram computation.

Firstly, we would divide the entropy sequence into many small fragments of the same size, where the size is set to six. And the local feature would be extracted from the segments based on the Haar wavelet transform. The representation of the segment is the combination of the multiple decomposition coefficients. Next, we create the codebook of the malware entropy based on these coefficients. The bag-of-words model quantifies the number of the words in each entropy sequence based on the codebook and then represents the malware samples as the histogram of these words. Besides, the K-means algorithm is implemented to yield the cluster center (i.e. words in the codebook) of the structural features.

D. LOCAL FEATURES EXTRACTION AND VECTORIZATION

1) STRUCTURE FEATURE EXTRACTION BASED ON THE HAAR WAVELET TRANSFORM

To further explore the characteristics of the entropy, we represent each malware as an entropy sequence, which can characterize the trend of entropy changes of the raw bytes in different locations of malware. And we use the Wavelet transform to process entropy sequences and extract features. Wavelet transform has many different characteristics from the Fourier transform. Its advantage is that more details of the signal can be obtained through different scales of processing.

- **Mean**: the mean value of the entropy sequence.
- **Standard Deviation**: the standard deviation of the entropy sequence.
- **Max**: the max value of the entropy sequence.
- **MIN_7**: the proportion of values larger than seven in the entropy sequence.
- **MAX_7**: the proportion of zero values in the entropy sequence.

In the wavelet transform, a series of transforming functions about mother wavelet and father wavelet need to be produced. In this process, the mother function and father function will undergo different translations and dilation. And the transforming functions are produced based on two important arguments (i.e. scaling and translating). The transforming functions are described as following equations.

$$\psi_{a,b}(t) = |a|^{-1/2} \psi\left(\frac{s-b}{a}\right)$$

(4)

$$\phi_{a,b}(t) = |a|^{-1/2} \phi\left(\frac{s-b}{a}\right)$$

(5)

where $a$ is scaling parameter and $b$ is the translating parameter. When $a=2j$ and $b=ia$, this set of function can be transformed into:

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - i)$$

(6)

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - i)$$

(7)

where $j$ is the scaling parameter, which represents the resolution level at different stages and $i$ is the position parameter. In the process of wavelet transformation, the entropy sequence signal is decomposed into a series of data characterizing the details of the sequence. Specifically, if we get an entropy signal $S(t)$ which consists of $T$ positions, we can get the father wavelet coefficients (i.e. approximate coefficient) of the signal combined with the set of scaling transforming functions, which is defined as follows:

$$a_{j,k}(t) = \{s, \phi_{j,k}\} = \sum_{i=1}^{T} s(t) \phi_{j,k}(t)$$

(8)

The father wavelet coefficients describe the inner character of the signal. Therefore, in this paper, we extract the father wavelet coefficients of the entropy signal as local features. The extracted approximation characteristics can provide some local information about the malware entropy sequences and improve the performance of malware classification. Besides, discrete wavelet decomposition allows for multiple decomposition analysis. At each turn of the decomposition, the previous father wavelet coefficients would be further processed into a more detailed approximate coefficient. Concretely, the process of decomposition is recursive and the signal of the entropy sequence $s(t)$ would be processed into a group of approximate coefficients $s_j(t)$. The process can be defined as follows:

$$s_{j+1}(t) = \{s_j(t), \phi_{j,k}\}$$

(9)

2) FEATURE VECTORIZATION BASED ON THE BAG-OF-WORDS MODEL

We could extract features based on the Haar Wavelet Transform algorithm, but how to vectorize features is a more important issue. In this paper, we convert the features into vectors by using the Bag-of-words model. The main procedure of the method is shown in Figure 3. It consists of two steps: (1) Codebook generation, (2) Histogram computation.

Firstly, we would divide the entropy sequence into many small fragments of the same size, where the size is set to six. And the local feature would be extracted from the segments based on the Haar wavelet transform. The representation of the segment is the combination of the multiple decomposition coefficients. Next, we create the codebook of the malware entropy based on these coefficients. The bag-of-words model quantifies the number of the words in each entropy sequence based on the codebook and then represents the malware samples as the histogram of these words. Besides, the K-means algorithm is implemented to yield the cluster center (i.e. words in the codebook) of the structural features.
Formally, the formation process of codebook can be defined as follows:

$$J(s_1, s_2, \ldots, s_n, \mu_1, \mu_2, \ldots, \mu_k) = \frac{1}{n} \sum_{i=1}^{n} ||s(i) - \mu_c(i)||$$

(10)

where $s_i$ is the structural feature extracted from the malware samples and $\mu_i$ is the cluster center (i.e. the word in the codebook). And the length of the words is the same as the structural feature.

In the second step, we covert malware samples into vectors with the codebook. The entropy sequence would be splitted into multiple blocks, then each block is earmarked to the word with the smallest distance. The malware samples would be transformed into the histograms based on the codebook. Figure 4 shows some examples of BOW representation.

The left sub-figure of Figure 4 is the malware entropy signal, and the right sub-figure is the corresponding histogram representation. The size of the codebook is set to 256 in this paper, and every word in the codebook is a structural local feature of the malware samples. The abscissa value represents different clustering centers and the ordinate value corresponds to the number of each local feature in the right sub-figures.

Finally, the histograms of code words are used to be local feature representations for entropy sequences. The machine learning classifier would be trained based on these two types of features.

**IV. EVALUATION**

**A. IMPLEMENTATION AND SETUP**

To evaluate whether the proposed method can effectively classify malware, we implemented a prototype system named MESRF (Malware Entropy Sequences Reflect the Family). All the system is programmed in Python 2.7 with the pywt library and the sklearn library. And the prototype system is deployed on a PC with an Intel (R) Core (TM) i7-4790 CPU (3.6GHz) and 20GB RAM.

1) **DATASET I**

Our experiments are mainly based on the malware dataset from the Malware Research Lab [3]. The data set consists of 9,339 malicious samples from 25 families, which are all collected from the real environment, and the malicious samples are classified by the Microsoft security platform. The details of the dataset are shown in Figure 5.

2) **DATASET II**

We also conducted experiments on the dataset from the Microsoft Malware Classification Challenge (BIG 2015) [26]. The dataset consists of 10,868 samples from
9 families (Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, Gatak) and is provided by Microsoft research and Northeastern university [27].

B. EVALUATION METRIC

The evaluation metric in this experiment includes accuracy, precision, recall, ROC curve (Receiver Operating Characteristic Curve) and AUC (Area Under ROC Curve). The true positive samples involved refer to the samples where the malicious samples are correctly classified into their corresponding families. The false positive samples refer to the samples where the malicious samples are misclassified into the specified family. True negative samples refer to malicious samples that do not belong to the specified family and are indeed classified into other families. False negative samples refer to malicious samples that belong to the specified family but are misclassified into other families. And the evaluation metrics are defined as follows.

1) ACCURACY

The ratio of the number of correct classification of samples to the entire number of samples.

\[ Acc = \frac{TP + TN}{TP + TN + FP + FN} \]  

(11)

2) RECALL

The ratio of true positive samples to entire positive samples in the data set.

\[ Recall = \frac{TP}{TP + FN} \]  

(12)

3) PRECISION

The ratio of true positives samples to the positive samples classified by the model.

\[ Precision = \frac{TP}{TP + FP} \]  

(13)

4) F1-MEASURE

F1-Measure is the weighted harmonic average of Precision and Recall.

\[ \frac{1}{F_1} = \frac{1}{2} \times \left( \frac{1}{P} + \frac{1}{R} \right) \]  

(14)

where \( P \) and \( R \) refer to the accuracy and recall. After conversion, the formula can be converted as following:

\[ F_1 = \frac{2 \times TP}{2 \times TP + FP + FN} \]  

(15)

5) ROC CURVE AND AUC

The ordinate of the ROC curve is the True Positive Rate (or called Recall), and the abscissa is the False Positive Rate. The false positive rate is defined as:

\[ FPR = \frac{FP}{TN + FP} \]  

(16)

When evaluating the generalization performance of two classifiers, the value of AUC is usually compared. A classifier with a large AUC value has a strong generalization ability.

C. CLASSIFIER EVALUATION

In this section, we conduct experiments to evaluate the performance of different classifiers, which are trained based on Random Forest (RF), Multi-layer Perceptron (MLP), k-Nearest Neighbor (KNN), support vector machines (SVM), Decision Tree (DT) algorithms. Each sample has 263 features (7 features from global features and 256 features from local features). In the setting of each classifier, the K value of the KNN classifier is set to 2, the SVM uses a linear kernel function, and the remaining classifiers use the default configuration parameters in sklearn. The results are shown in Figure 6.
than 99.5%. The SVM classifier has the best performance and the accuracy could reach 99.82%. In the process of ten cross-validation experiments, the SVM classifier even appeared to have a 100% accuracy of malware classification in some experiments.

Besides, efficiency and generalization ability are the important evaluation metrics of the classifier. In the experiment, this paper also evaluated the training time, test time and AUC value of each classifier. The results are shown in Table 1.

### TABLE 1. Generalization ability and efficiency of classifiers.

| Classifier | Training time (s) | Testing time (s) | P1 (%) | AUC   |
|------------|-------------------|-----------------|--------|-------|
| RF         | 0.15              | 0.007           | 99.46  | 0.9996|
| MLP        | 19.21             | 0.003           | 99.33  | 0.9991|
| KNN        | 0.13              | 0.27            | 98.48  | 0.9864|
| SVM        | 7.02              | 0.16            | 99.83  | 0.9998|
| DT         | 0.25              | 0.0001          | 99.23  | 0.9902|

From the results in Table 1, we can see that the SVM classifier has the best F1-score and AUC values in the experiment, but the training time is relatively long, it has reached about 7 seconds on the training dataset with more than 9,000 samples. Although the classification performance of the RF classifier is slightly worse than that of the SVM classifier, it has achieved almost the best performance in training and testing time. In terms of efficiency evaluation, the RF classifier is the classifier with the best overall performance in the experiment. However, the test time of the SVM classifier is only 0.16 seconds. This efficiency performance is also not bad. Combined with its performance on the accuracy, if the efficiency requirements are not very strict, the SVM classifier is still the best choice for malware classification. Therefore, we deployed our prototype system based on the SVM classifier.

### D. THE IMPACT OF FEATURES

We mainly extracted two major types of features from the malware signal: global features and local features. To evaluate the impact of different features on the classification results, and whether the combined features could improve the experimental performance, we trained the classifier based on the two types of features separately, and the results are shown in Table 2.

### TABLE 2. The performance of the classifier with different features.

| Features     | Acc (%) | Pre (%) | Recall (%) | P1 (%) |
|--------------|---------|---------|------------|--------|
| Global Features | 88.585  | 91.106  | 88.275     | 88.509 |
| Local Features   | 99.753  | 99.768  | 99.753     | 99.754 |
| Global + Local   | 99.828  | 99.841  | 99.829     | 99.829 |

From the results, we can see that both types of features can be used to classify the malware, and the accuracy of the classifier based on the local features extracted by DWT and BOW could reach 99.75%. Moreover, we combined two types of features and trained the classifier based on the combined features. The accuracy of the classifier has been further improved, reaching 99.83%. We can know that both global features and local features contribute to the final classification performance.

To further analyze why the method proposed in this paper can give better performance and provide a reference for the following research, we analyzed the impact of 263 features on the classification results in the experiment. As shown in Figure 7, it is the impact of the features of one experiment in ten cross-validation experiments. The features shown in Figure 7 are the serial numbers of the top 15 most impactful features in the experiment. These features are 15 codewords in the codebook, which correspond to different clustering centers. The top five word values among these fifteen features are shown in Table 3.

### FIGURE 7. The impact of features.

### TABLE 3. Top 5 impactful features.

| NUM | Word in Codebook |
|-----|------------------|
| 164 | [0.29591626 9.9236417 10.0676709 7.22631873 14.3396808] |
| 40  | [10.131403 10.1329987 8.68861259 14.32909607 11.266218] |
| 52  | [2.61138929 0.2788261 9.59590078 2.04369092 20.9224950] |
| 79  | [7.50917074 6.77499974 6.57281398 10.1004338 9.1523957] |
| 173 | [6.68683308 6.50131741 6.08563977 9.32543064 8.3124658] |

From Figure 7, we can see that the first-ranked feature has a significantly higher impact on the classification results than other features and in the ten cross-validation experiments, the values of first-ranked features are all close to the word of the feature 164 in Table 3. It indicates that in the experiment, this clustering center in the codebook plays an important role in good classification performance. However, further research is still needed on how this feature works.

### E. COMPARED WITH OTHER MALWARE CLASSIFICATION METHODS

To validate the effectiveness and efficiency of the proposed method, we compared our method with four other malware classification methods [3], [6], [24], [25] that use the structural features of malware. Each of the methods extracts the structural features (texture features) from the raw bytes of the malware, and then applied machine learning algorithms
(e.g., RF, KNN) to classify the malware. Table 4 shows the results of our method compared with the four combination models: GIST+KNN, LBP+KNN, GIST+DSIFT+KNN, VGG16 (fine-tune). We can see that our method has achieved higher accuracy.

**TABLE 4. Proposed method compared with other malware classification methods on dataset I.**

| Methods                  | Acc (%) | Recall (%) | F1 (%) | RT (s) |
|--------------------------|---------|------------|--------|--------|
| GIST + KNN               | 97.23   | 97.23      | 96.61  | 0.423  |
| LBP + KNN                | 97.87   | 97.88      | 98.6   | 0.052  |
| GIST + DSIFT + RF        | 98.53   | 98.53      | 98.49  | 0.026  |
| VGG16 (fine-tune)        | 98.84   | 98.84      | 98.82  | 59.50  |
| **MESRF**                | **99.83** | **99.82**  | **99.83** | **0.165** |

1 RT means running time.

The method proposed by Naeem et al. (GIST+DSIFT) [6] extracts more complex features based on malicious samples and improved classification accuracy. And the method proposed by Rezende et al. (VGG16 fine-tune) [25] combined deep learning and transfer learning algorithms to achieve a higher classification accuracy of malware classification. However, our method (MESRF) still improved the classification accuracy rate by 1% and achieved a better classification performance. In terms of efficiency, the method proposed in this paper performs slightly worse than the methods proposed by Hashemi and Hamzeh [24] and Naeem et al. [6]. However, it shows superiority to the methods proposed by Nataraj et al. [3] and Rezende et al. [25]. To sum up, the method proposed in this paper completes the classification task in a relatively short time, proving that this method is also applicable to large-scale malware classification problems.

Second, we also compared our method with other four methods on dataset II. Table 5 shows the results of our method compared with the four combination models. We can see that VGG16 (fine-tune) and our method have achieved higher accuracy. The accuracy of our method is almost the same as VGG16 (fine-tune) (i.e., only 0.04% lower), but in terms of efficiency, our method performs far better than VGG16. So, in terms of both accuracy and efficiency, our method still has the best performance.

**TABLE 5. Proposed method compared with other malware classification methods on dataset II.**

| Methods                  | Acc (%) | Recall (%) | F1 (%) | RT (s) |
|--------------------------|---------|------------|--------|--------|
| GIST + KNN               | 94.82   | 94.81      | 94.75  | 0.621  |
| LBP + KNN                | 91.87   | 91.88      | 91.87  | 0.045  |
| GIST + DSIFT + RF        | 95.09   | 95.10      | 95.06  | 0.004  |
| VGG16 (fine-tune)        | 97.92   | 97.95      | 97.92  | 62.59  |
| **MESRF**                | **97.88** | **97.91**  | **97.88** | **0.281** |

To further evaluate the classification performance of the method proposed in this paper, we deeply compared the detection performance with the methods proposed by Cui et al. [4] and Rezende et al. [25] on each malware family of dataset 1. The results are shown in Figure 8. Both the two methods proposed by Cui et al. [4] and Rezende et al. [25] are based on deep learning algorithms to classify malware, and the performance is better than other researches in this field.

In the malware dataset, numerous research methods cannot effectively solve the malware classification problem of the Swizzor.gen!E family and Swizzor.gen!I family, C2LOP.P family and C2LOP.gen!g family. As shown in Figure 8, we can see that even though these two methods have achieved good classification results on the dataset, the malware classification problem of similar families still has not been completely solved. VGG16 (fine-tune) almost solves the problem of confusion between the C2LOP.P family and C2LOP.gen!g family samples, but the classification accuracy of the samples in the Swizzor.gen!E family and Swizzor.gen!I family still could not reach 80%. However, the method proposed in this paper can better solve the classification problem of similar families of malware. The classification accuracy of samples from the Swizzor.gen!E family and Swizzor.gen!I family, C2LOP.P family and C2LOP.gen!g family could reach 90%.

**FIGURE 8. Comparison with other methods on each malware family of dataset 1.**

**V. LIMITATION AND DISCUSSION**

This paper conducted a 10-fold cross-validation experiment on the experimental dataset, which can objectively prove that the method proposed in this paper is true and effective on the experimental data set. It can be seen from the experimental results that the method proposed in this paper can effectively solve the problem that the classification accuracy of samples of similar families is not high in the existing methods, indicating that the method proposed in this paper has a relatively good performance. However, in the actual environment, the types of malicious samples are complex, and new malware is also emerging. Different methods have their pros and cons in different situations. The method proposed in this paper cannot solve all the problems of malware classification and may be limited in some cases. In future research and practical application, the method proposed in this paper can be used as an effective means of malware classification, combined with other excellent methods, to better solve various problems in the actual environment.

In the experiment of this paper, the most impactful element of the classifier is analyzed, and the most important word in the codebook is located, but the specific attributes of the word and the reason why it can effectively distinguish samples of similar malware families have not been fully analyzed.
The analysis of this important feature attribute still needs further study.

VI. CONCLUSION AND FUTURE WORK

This paper proposed a new method to improve the classification of malware samples based on the entropy signal features. The statistical characteristics and the internal detail features were extracted to provide the global and local information of the malware samples. At last, a machine learning-based classifier was trained to identify and classify the malware samples. We implement the method and evaluate it on custom datasets with more than 20,000 malware samples. The experiments showed that the method proposed in this paper could effectively facilitate us in distinguishing malware variants belonging to similar families. And MESRF even could achieve 99.83% accuracy with good detection speed on the maling dataset, showing its superiority comparing with the related methods. This paper also serves as proof that the entropy feature of the raw bytes can play an important role in the malware classification, and is of great significance for future research to further improve the accuracy and efficiency of classification.

In future work, we would further explore the features contained in the entropy sequence of malware and conduct in-depth research on how to propose better methods for malware classification combined with other features.

ACKNOWLEDGMENT

The authors thank Zhi-Hao Yan, Xuan-Zhen Guo, Rui-Peng Wang, and Yuan-Chao Chen for their help in preparation of experiment and paper review. They also thank the anonymous reviewers for their detailed comments which helped to improve the quality of the paper.

REFERENCES

[1] SYMANTEC. (May 16, 2020). Internet Security Threat Report. [Online]. Available: https://www.broadcom.com/support/security-center/publications/threat-report
[2] D. Ucci, L. Aniello, and R. Baldoni, “Survey of machine learning techniques for malware analysis,” Comput. Secur., vol. 81, pp. 123–147, Mar. 2019.
[3] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in Proc. 8th Int. Symp. Vis. Cyber Secur. (VisSec), 2011, pp. 1–7.
[4] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, “Detection of malicious code variants based on deep learning,” IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 3187–3196, Jul. 2018.
[5] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, “A comparative assessment of malware classification using binary texture analysis and dynamic analysis,” in Proc. 4th ACM Workshop Secur. Artif. Intell. (AISec), 2011, pp. 21–30.
[6] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabas, and M. S. Javed, “Identification of malicious code variants based on image visualization,” Comput. Electr. Eng., vol. 76, pp. 225–237, Jun. 2019.
[7] Z. Yang, P. Wang, and J. Wang, “ConvProtoNet: Deep prototype induction towards better class representation for few-shot malware classification,” Appl. Sci., vol. 10, no. 8, p. 2847, Apr. 2020.
[8] M. Wojnowicz, G. Chisholm, M. Wolff, and X. Zhao, “Wavelet decomposition of software entropy reveals symptoms of malicious code,” J. Innov. Digit. Ecosyst., vol. 3, no. 2, pp. 130–140, Dec. 2016.
[9] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M.-S. Choi, “Entropy analysis to classify unknown packing algorithms for malware detection,” Int. J. Inf. Secur., vol. 16, no. 3, pp. 227–248, Jun. 2017.
[10] L. Liu, X. He, L. Liu, L. Qing, Y. Fang, and J. Liu, “Capturing the symptoms of malicious code in electronic documents by file's entropy signal combined with machine learning,” Appl. Soft Comput., vol. 82, Sep. 2019, Art. no. 105598.
[11] G. Canfora, F. Mencaldo, and C. A. Visaggio, “An HMM and structural entropy based detector for Android malware: An empirical study,” Comput. Secur., vol. 61, pp. 1–18, Aug. 2016.
[12] R. Moskovitch, C. Feher, and Y. El-Yovi, “A chronological evaluation of unknown malcode detection,” in Proc. Pacific-Asia Workshop Intell. Secur. Inform. Berlin, Germany: Springer, 2009, pp. 112–117.
[13] E. Ruff and C. Nicholas, “An alternative to NCD for large sequences, lempel-ziv jaccard distance,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2017, pp. 1007–1015.
[14] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, “Novel feature extraction, selection and fusion for effective malware family classification,” in Proc. 6th ACM Conf. Data Appl. Secur. Privacy (CODASPY), 2016, pp. 183–194.
[15] S. Pai, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, “Clustering for malware classification,” J. Comput. Virol. Hacking Techn., vol. 13, no. 2, pp. 95–107, May 2017.
[16] Y. Dai, H. Li, Y. Qian, R. Yang, and M. Zheng, “SMASH: A malware detection method based on multi-feature ensemble learning,” IEEE Access, vol. 7, pp. 112588–112597, 2019.
[17] A. Mohaisen, O. Alrawi, and M. Mohaisen, “AMAL: high-fidelity, behavior-based automated malware analysis and classification,” Comput. Secur., vol. 52, pp. 251–266, Jul. 2015.
[18] H. Kim, J. Kim, Y. Kim, J. K. Kim, and H. Kim, “Improvement of malware detection and classification using API call sequence alignment and visualization,” Cluster Comput., vol. 22, no. S1, pp. 921–929, Jan. 2019.
[19] J. Zhang, Z. Qin, H. Yin, L. Ou, S. Xiao, and Y. Hu, “Malware variant detection using opcode image recognition with small training sets,” in Proc. 25th Int. Conf. Comput. Commun. Netw. (ICCCN), Aug. 2016, pp. 1–9.
[20] K. Han, B. Kang, and E. G. Im, “Malware analysis using visualized image matrices,” Sci. World J., vol. 2014, pp. 1–15, 2014.
[21] K. Kosmidis and C. Kalloniatis, “Machine learning and images for malware detection and classification,” in Proc. 21st Pan-Hellenic Conf. Inform. (PCI), 2017, pp. 1–6.
[22] B. Xiaofang, C. Li, H. Weihua, and W. Qu, “Malware variant detection using similarity search over content fingerprint,” in Proc. 26th Chin. Control Decis. Conf. (CCDC), May 2014, pp. 5334–5339.
[23] H. Naeem, B. Guo, F. Ullah, and M. R. Naeem, “A cross-platform malware variant classification based on image representation,” KSII Trans. Internet Inf. Syst., vol. 13, no. 7, pp. 3756–3777, 2019.
[24] H. Hashemi and A. Hamzeh, “Visual malware detection using local malicious pattern,” J. Comput. Virol. Hacking Techn., vol. 15, no. 1, pp. 1–14, Mar. 2019.
[25] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. D. Geus, “Malicious software classification using VGG16 deep neural network’s bottleneck features,” in Information Technology-New Generations (ITNG), 2017, pp. 51–59.
[26] Microsoft. (Jun. 28, 2020). Microsoft Malware Classification Challenge (Big 2015). https://www.kaggle.com/c/malware-classification
[27] R. Ronen, M. Rudu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, “Microsoft malware classification challenge,” 2018, arXiv:1802.10135. [Online]. Available: http://arxiv.org/abs/1802.10135
SHUGUANG HUANG received the Ph.D. degree from the University of Science and Technology of China, Hefei, China. He is currently a Professor with the College of Electronic Engineering, National University of Defense Technology. His research interests include computer science, computer engineering, and network security.

CHENG HUANG (Member, IEEE) received the Ph.D. degree from Sichuan University. He is currently an Assistant Research Professor with Sichuan University. His research interests include cyber security, attack detection, and threat intelligence analysis.

ZULIE PAN received the Ph.D. degree from the Electronic Engineering Institute, Hefei, China. He is currently a Professor with the College of Electronic Engineering, National University of Defense Technology. His research interests include vulnerability discovery, network security, and computer science.

MIN ZHANG received the Ph.D. degree from Anhui University. He is currently a Professor with the National University of Defense Technology. His research interests include machine learning, data mining, and computer security.

FAN SHI received the master’s degree from the Electronic Engineering Institute, Hefei, China. He is currently an Associate Professor with the College of Electronic Engineering, National University of Defense Technology. His research interests include data analysis, network security, and networking protocol analysis.