Chinese Journal of Electronics
Vol.32, No.1, Jan. 2023

MalFSM: Feature Subset Selection Method for Malware Family Classification

KONG Zixiao[1], XUE Jingfeng[1], WANG Yong[1], ZHANG Qian[1], HAN Weijie[2], and ZHU Yufen[2]

(1. *School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*)

(2. *School of Space Information, Space Engineering University, Beijing 101416, China*)

Abstract — Malware detection has been a hot spot in cyberspace security and academic research. We investigate the correlation between the opcode features of malicious samples and perform feature extraction, selection and fusion by filtering redundant features, thus alleviating the dimensional disaster problem and achieving efficient identification of malware families for proper classification. Malware authors use obfuscation technology to generate a large number of malware variants, which imposes a heavy analysis burden on security researchers and consumes a lot of resources in both time and space. To this end, we propose the MalFSM framework. Through the feature selection method, we reduce the 735 opcode features contained in the Kaggle dataset to 16, and then fuse on metadata features (count of file lines and file size) for a total of 18 features, and find that the machine learning classification is efficient and high accuracy. We analyzed the correlation between the opcode features of malicious samples and interpreted the selected features. Our comprehensive experiments show that the highest classification accuracy of MalFSM can reach up to 98.6% and the classification time is only 7.76 s on the Kaggle malware dataset of Microsoft.

Key words — Malware classification, Machine learning, Feature selection, Feature correlation.

I. Introduction

In the era of network space, malware is designed specifically by malware coders to damage computers or access them illegally, which has grown rapidly. An Internet security threat report from McAfee Labs [1] shows that 419 malwares per minute were discovered in the second quarter of 2020, an increase of nearly 12 percent from the first quarter. As computers and networks become more and more indispensable to daily life, the threat of malware is growing. For example, Trickbot, a banking Trojan discovered in September 2016, still topped the global threat index by Checkpoint in October 2021 [2]. With the widespread use of smartphones, not only desktops and laptops but even mobile phones and devices connected to the Internet of things can be attacked by malware. Attackers can make a fortune, and are hard to track. Therefore, the analysis and detection of malware have become a difficult task for anti-malware researchers.

Therefore, this paper proposes a lightweight framework of feature subset selection method for malware family classification (MalFSM). The implementation process of MalFSM is as follows: First of all, opcode sequences are extracted from the disassembled codes of malicious datasets. Take the frequency of opcode as the eigenvalue to construct a simple feature vector. Then the feature selection algorithm is utilized to extract the top-n opcode features. After fusion of selected feature subset and the metadata features, generated characteristic matrix of the original sample for automatic classification. Constructing a lightweight feature set can effectively reduce the classification time and space of a large number of sample sets.

We evaluate the MalFSM on the Kaggle dataset provided by the 2015 Microsoft Malware Classification Challenge with satisfactory results. In addition, compared with other studies, the classification time is shortened obviously and the space occupancy rate is significantly reduced. To sum up, our main contributions to this paper are as follows:

1) We propose a malware feature extraction, selection, and fusion framework named MalFSM. This method can help analysts filter out the most relevant fea-

Manuscript Received Mar. 9, 2022; Accepted May 13, 2022. This work was supported by the National Natural Science Foundation of China (62172042), Major Scientific and Technological Innovation Projects of Shandong Province (2020CXGC010116), and the National Key Research & Development Program of China (2020YFB1712104).

 DOI:10.23919/cje.2022.00.038

tures without the need for high-performance hardware resources, and provides reference and interpretation for malicious code researchers to choose opcode in the future. Due to the large scale and quantity of features in the complete dataset, feature subsets are selected according to the feature selection methods, and a lightweight feature matrix is constructed. The experimental results show that the lightweight eigenmatrix can not only reduce the time and space complexity, and reach the spatio-temporal equilibrium, but also achieve high classification accuracy.

2) Based on this, the superiority of the method in terms of time and space overhead is experimentally verified. The classification accuracy can reach 98.6%. Compared with relevant studies, the classification time is reduced by 69%, and the space occupation is reduced by 93.1%. MalFSM can provide accuracy similar to that of the first prize in the competition, while effectively simplifying the feature space, and is superior to the previous algorithm in terms of efficiency. When using a smaller training set for classification, our accuracy rate is around 94%, with good generalization ability.

3) The extracted opcode features are visualized and transformed into grayscale images. The Transformer model and VGG-16 model are used for classification. Compared with the machine learning classification algorithm, the classification effect is slightly inferior but has excessive time overhead.

The remainder of this paper is organized as follows: Section II introduces the necessary background knowledge and summary of related work. Section III describes the overall implementation of the MalFSM. The experimental details are presented in Section IV. Then evaluate it in Section V. Finally, Section VI concludes the paper and points out the topics to be studied in the future.

II. Background and Related Work

Detecting and classifying malware is a constant battle for researchers in cyberspace security. In recent years, malware analysis and detection techniques have developed continuously. In order to better understand its development, we first briefly introduce the research background and related work.

1. Background

1) Analysis method of malware

Malicious code, also known as malware, plays a major role in most computer breaches. Any software that causes damage to a user, computer, or network in some way can be considered malicious code, including viruses, ransomware, trojans, worms, etc. As malware researchers, we need to understand how to identify malicious code and analyze it. Malicious code analysis is the art of dissecting malicious code. It can be divided into static analysis, dynamic analysis, and hybrid analysis according to whether malicious programs actually run.

Static analysis techniques examine executables without malicious code running. Based on static analysis, we can extract important features directly from binary files, including DLL, API, Opcode, strings, bytes n-gram, and so on. By applying machine learning or deep learning technology to static features, the static method can detect and classify malicious software quickly and accurately. But it is largely ineffective against complex malicious code and might miss some important behavior [3].

In contrast to static analysis, dynamic analysis techniques involve running malicious code and observing behavior on the system [4]. Before running malicious programs, ensure that a secure environment is established to prevent risks to the system and network. Typical dynamic analysis tools include Cuckoo Sandbox, CWSandbox, and Joe Sandbox. In cyber security, the sandbox is the tool used to handle untrusted files or applications in an isolated environment. For instance, Hu *et al.* [5] used the Cuckoo Sandbox to process malware samples. In addition to the sandbox virtual environment, you can also use the debugger to check the behavior and effects of a malicious program when it runs.

Static analysis and dynamic analysis have their own advantages and disadvantages. Static analysis saves the actual running time of malicious code, and some ransomware can detect sandbox virtual environment so that dynamic analysis can not be used. Dynamic analysis is more effective at obfuscating malware, but it consumes a lot of time and resources. In summary, using dynamic and static hybrid analysis technology can better analyze and detect malicious code. Han *et al.* [6] first attempted to study the difference and relationship between static and dynamic API sequences of malicious code. They proposed the MalDAE framework, which fuses the dynamic and static API sequence associations of malicious programs into a hybrid sequence based on semantic mapping to construct a hybrid feature vector space. On the basis of achieving high accuracy and classification rate, comprehensible interpretation is provided.

2) Methods of feature selection

According to the form of feature selection [7], methods can be divided into three categories: Filter, Wrapper, and Embedded [8]. Common models for each of these three methods are listed below:

a) Filter

i) Pearson correlation coefficient

Pearson correlation coefficient is used to measure

the degree of correlation between two variables, and its value is between −1 and 1 [9]. In general, the correlation strength of variables can be judged by the following range of correlation coefficients:

- 0.8–1.0, very strong correlation;
- 0.6–0.8, strong correlation;
- 0.4–0.6, moderate correlation;
- 0.2–0.4, weak correlation;
- 0.0–0.2, very weak correlation or no correlation.

ii) Chi-square

Chi-square selection is a supervised feature selection method commonly used in statistics. It determines the correlation between the feature and the real label by chi-square test and then determines whether to select it [10].

We can use the Chi-square test to test the independence of features and dependent variables. If the independence is high, it means that there is little relationship between the two, and features can be discarded. If the independence is small and the correlation between the two is high, it indicates that the feature will have a relatively large impact on the corresponding variable and should be selected. When the Chi-square test is applied to feature selection in practice, it is not necessary to know the degree of freedom and chi-square distribution, but only to sort according to the calculated χ^2. The larger the value, the better. Pick the largest pile, and you're done with the Chi-square test for feature extraction. For classification problems, filtering methods generally assume that features independent of labels are irrelevant features, while the Chi-square test can precisely test independence, so it is suitable for feature selection. If the result is that a feature is independent of the label, the feature can be removed.

iii) Mutual information

Mutual information is a useful information measure in information theory. It can be regarded as the amount of information contained in a random variable about another random variable, or the uncertainty reduced by a random variable due to the fact that another random variable is known [11].

iv) Variance threshold

Variance threshold: First calculate the variance of each feature, and then select features whose variance is greater than the threshold based on the threshold. To eliminate the features with variance of 0 first, we will only use variance filtering with threshold of 0 or small threshold to eliminate some obviously unused features first, and then we will choose better feature selection methods to continue to reduce the number of features [12].

v) F-test (ANOVA)

F-test, the most common alias is called joint hypothesis test. Used to test the mean difference significance between two and more samples [13].

b) Wrapper

Recursive feature elimination (RFE): The main idea of RFE is to build the model over and over again, pick out the best (or worst) features (based on coefficients), put those features aside, and repeat the process for the rest of the features until all the features are iterated. The order that is eliminated in this process is the ordering of features.

The stability of RFE depends largely on which model is used at the bottom of the iteration. For example, RFE adopts linear regression (LR), and the regression without regularization is unstable, so RFE is unstable. If lasso/ridge is used and the regularized regression is stable, then RFE is stable [14].

c) Embedded

Embedded is a method to let the algorithm decide which features to use, that is, feature selection is integrated with the training process of the learner. They are completed in the same optimization process, in other words, feature selection is automatically carried out in the training process of the learner. First, some machine learning algorithms (e.g. RF, LR) are used for training to obtain the weight coefficients of each feature [15]. Features are selected from large to small according to the weight coefficients, which often represent some contribution or importance of features to the model. The process is similar to Filter, except that the coefficients are trained.

i) Feature selection method based on penalty term: L_1 norm, L_2 norm, L_1 norm and L_2 norm may also be used simultaneously.

ii) Feature ordering based on machine learning model: decision tree, random forest, etc.

Table 1 summarizes the feature selection methods.

2. Related work

As the research direction of this paper is the feature subset selection technology of malware family classification, this part simply summarizes the work related to malware classification and feature selection (as shown in Table 2).

Modern malware is designed to have mutational characteristics, namely polymorphism, and metamorphosis, which leads to tremendous growth in the number of variants in the malware sample. Classifying malware samples by their behavior is critical to the computer security community because they receive a large number of malwares each day, and the feature extraction process is often based on the malicious portion of the malware family. Ahmadi *et al.* [16] proposed a new technique to extract PE structural features to accurately classify malware. Based on feature extraction, the most

Table 1. Feature selection method

Feature selection method	Filter	Pearson correlation coefficient	Only linear correlations can be captured
		Chi-square	Dedicated to classification algorithms, capture correlation
		Mutual information classification	Can capture any correlation
		F-test (ANOVA)	Only linear correlation can be captured, requiring data to follow normal distribution
		Variance threshold	The variance threshold can be entered to return a new eigenmatrix whose variance is greater than the threshold
	Wrapper	Recursive feature elimination	Iterate over all the features and build the model repeatedly to select the best features
	Embedded	Feature selection method based on penalty	Select features based on penalty terms
		Feature order based on ML	The features are sorted according to machine learning models

Table 2. Summary of the existing malware detection techniques

Author	Analysis method	Dataset	Features	Time consuming	Merits	Demerits	Accuracy
M. Ahmadi *et al.* [16]	Static & Machine learning	Microsoft Malware Challenge dataset	Hex dump-based features, features extracted from disassembled files	Feature extraction time is 5656 s	Providing a trade-off between accuracy and the number of features	Complex feature extraction and classification; Time consuming; No robustness testing	99.77%
S. Ni *et al.* [17]	Static & Machine learning	Microsoft Malware Challenge dataset	Opcode	Identifying a malware sample takes 1.41 s in average. Feature extraction time is NaN	Proposing a novel algorithm to extract features; High accuracy	Lack of high-performance computing	99.26%
W. Han *et al.* [18]	Static & Machine learning	Microsoft Malware Challenge dataset	Top-N opcode list	Feature extraction and generating opcode feature vector time is 3563.42 s	Proposed a parallel process framework	Less characteristic variety	98.53%
A. Darem *et al.*[19]	Static & Deep learning	Microsoft Malware Challenge dataset	Opcode, segment, asm pixed count and other features	Training and detection time of model is 1038 s	High accuracy; Integrates an ensemble deep learning, feature engineering, image transformation and processing techniques	The feature extraction process will take a large scale of time	99.12%
I. Almomani *et al.* [20]	Static & Machine learning	6190 benign apps & 5500 malware apps	Permissions	The fastest execution time of model is 0.1 s	Demonstrate the usefulness of feature selection	Solely on a single feature	About 85%
G. Sun *et al.* [21]	Static & Deep learning	Microsoft Malware Challenge dataset	Opcode	NaN	High accuracy; Excellent generalization	Less characteristic variety	99.5%

effective combination of feature categories is output by feature fusion algorithm to achieve a compromise between accuracy and the number of features. However, this paper did not simplify the features and collect the core common features of each family sample.

Ni *et al.* [17] proposed a malware classification algorithm MCSC (malware classification using SimHash and CNN) utilizing static analysis, which converts disassembled malware into grayscale images based on SimHash. Then the convolutional neural network is used for family classification. In this paper, a malware feature extraction algorithm combining opcode sequence and LSH is proposed, and the feature is converted into fingerprint images using visualization technology. In addition, multi-hash, major block selection, and bilinear interpolation are adopted to improve the classification algorithm. However, the proposed method needs further improvement in terms of performance and robustness.

For large-scale malware datasets, Han *et al.* [18] proposed a parallel classification framework based on

multi-core collaboration and active recommendation. The framework can run on PC without high-performance hardware resources. In addition, this method can reduce processing time and improve efficiency while achieving high classification accuracy. However, the feature types considered in this paper are relatively single, so multidimensional feature analysis can be further adopted.

In order to deal with the major threat brought by malware, Darem *et al.* [19] proposed a semi-supervised approach integrating deep learning, feature engineering, image transformation, and processing techniques for fuzzy malware detection. This method extracts features from assembler files to form images. Then, integrated deep learning is fine-tuned to achieve efficient learning. However, the transformed images in this study are grayscale images, so whether this method can be applied to color images is worth studying.

Due to the rapid growth of Android malware and the impact of high-quality features on machine learning performance, Jemal Abawajy *et al.* [20] regarded feature selection as a quadratic programming problem and studied the effect of feature selection on malware detection under intelligent IoT platform. In this paper, filter-based feature selection analysis is carried out for Android malware. The usefulness of feature selection is proved by empirical evaluation of the prediction accuracy of feature selection techniques using several learning algorithms that do not perform feature subset selection internally. However, this study is only based on the functions of a single component of the Android system (namely permissions), and it also needs to analyze permissions combination and API features.

Sun *et al.* [21] proposed a new malicious code visualization method, RMVC, combining RNN and CNN technology. The paper considers not only the original information of malware, but also the ability to associate the raw code with temporal characteristics. Then, minhash is adopted to generate feature images from the fusion of the original code and the RNN prediction code. Finally, CNN is trained to classify images with excellent generalization ability. However, the paper does not consider large datasets and dynamic analysis.

III. Overview

1. Motivation

High-quality features are vital for malware classification, this is because the original feature set extracted from malicious samples is very large. For example, Le *et al.* [22] extracted 10,000 features from the Kaggle dataset, and Hu *et al.* [23] extracted 2000 features from the Kaggle dataset, Ahmadi *et al.* [16] extracted 1804 features from the Kaggle data set. In general, some of these features are key to the classification of malware families, while others are not. In addition, too many features can lead to model complexity and dimension disaster problems. Moreover, these features tend to be highly dimensional and redundant, and may not be related to the malware family classification. Furthermore, a large number of redundant features are of little value to the performance of learning machine learning algorithms and cost time and CPU. Therefore, these unwanted features should be removed from the feature set used to train machine learning models.

2. Overall framework

MalFSM mainly consists of four modules, namely, data preprocessing (including feature extraction), feature selection, feature fusion and classification, as shown in the Fig.1.

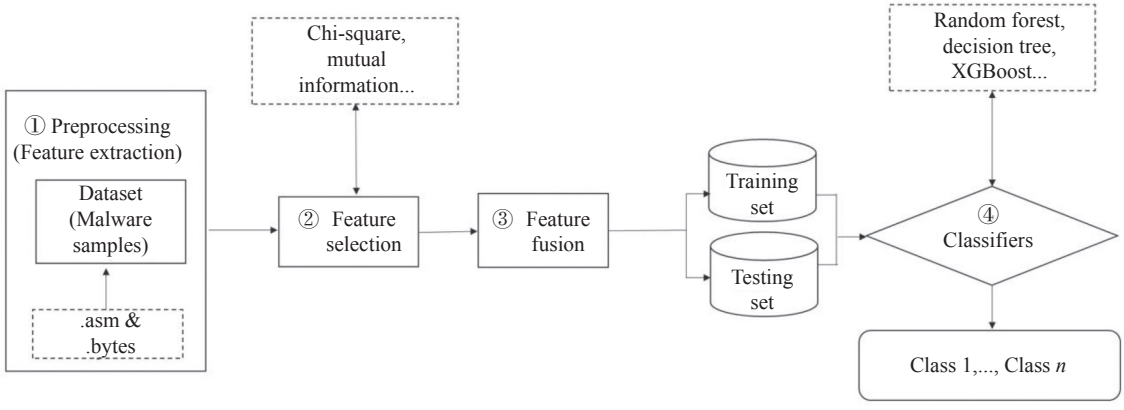


Fig. 1. The framework of MalFSM.

1) Preprocessing

The function of this module is to extract static features from malicious samples. In this paper, we extract the metadata features (namely the size of the file and

the count of lines in the file) and the opcode features (the number of opcode occurrences is taken as the opcode feature).

2) Feature selection

The function of this module is to purify feature vectors by removing redundant opcode features.

3) Feature fusion

Opcode features and metadata features are fused to form a feature matrix.

4) Classification

Based on the above generated feature matrix, we use four different classification models to verify the classification effect of MalFSM through cross-validation.

IV. Implementation

1. Preprocessing

The data source for this article is the Kaggle malware classification dataset released by Microsoft, including ".asm" and ".bytes" samples. The Kaggle dataset is the disassembled files that can extract the required metadata features and opcode features directly from the sample assembler. Save the statistical results in a CSV file.

2. Feature selection

The classification results of different feature selection algorithms are compared. Feature subsets are obtained by sorting different algorithms, and then the feature selection algorithm with the highest classification accuracy is selected to select the optimal feature subset. The final feature subset is the optimal feature set after removing redundant features.

3. Feature fusion

The method of feature fusion is to combine the selected features into a feature vector, and then run the classifier for family classification. We use a $m \times n$ dimensional binary vector to represent sample features. Given a sample set $S = \{s_1, s_2, \ldots, s_m\}$, and the feature set $F = \{OP_1, OP_2, OP_{n-2}, \text{file\_size}, \text{line\_count}\}$ (OP is the opcode), we create a feature vector as (1), where f_{ik} refers to the frequency of opcode k in sample i; L_i and $Size_i$ refer to the line_count and file_size respectively in sample i.

$$v_{ik} = \begin{cases} f_{ik} \\ L_i, Size_i \end{cases} \tag{1}$$

Table 3 shows the eigenvectors of a sample.

Table 3. Feature vectors for malwares

Samples	OP_1	OP_2	⋯	OP_{n-2}	file_size	line_count
s_1	f_{11}	f_{12}	⋯	$f_{1,n-2}$	$Size_1$	L_1
s_2	f_{21}	f_{22}	⋯	$f_{2,n-2}$	$Size_2$	L_2
⋮	⋮	⋮	⋱	⋮	⋮	⋮
s_m	f_{m1}	f_{m2}	⋯	$f_{m,n-2}$	$Size_m$	L_m

4. Malware family classification

Based on the feature matrix generated by the above feature fusion, we use four different classification models to verify the classification effect of MalFSM, including random forest (RF), decision tree (DT), XGBoost (XGB), and deep forest (DF), through cross-validation method. They have been widely used in information security, data mining and natural language processing. More details on classification techniques will be provided in the evaluation section.

V. Evaluation

1. Experimental setup

We implemented our models in python. The experimental environment is configured as follows: 1) Lenovo ThinkStation, Intel® Core™ i7-6700U CPU @3.40 GHz × 24.0 GB RAM, 2) 64 bit Windows 10, and 3) Pycharm 2020.

1) Data

The experimental data in this paper are from the 2015 Kaggle Microsoft Malware Classification Challenge, which aims to classify nine malware families. Since the labels of the test dataset could not be obtained, we directly cross-validated the training dataset. The dataset contains nine malware families, namely Ramnit (R), Lollipop (L), Kelihos ver3 (K3), Vundo (V), Simda (S), Tracur (T), Kelihos_ver1 (K1), Obfuscator.ACY (O), Gatak (G). Table 4 and Fig.2 illustrate the data distribution of this dataset.

Table 4. Kaggle dataset

Family name	1(R)	2(L)	3($K3$)	4(V)	5(S)	6(T)	7($K1$)	8(O)	9(G)	Total
Num	1541	2478	2942	475	42	751	398	1228	1013	10868

2) Malware classification models

In order to evaluate the effectiveness of the selected feature subset, we adopted four classification models to conduct experiments in the malware family classi-

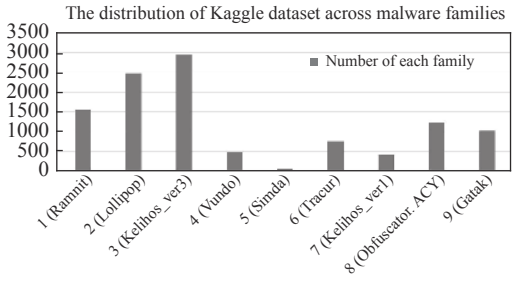


Fig. 2. The distribution of Kaggle dataset across malware families.

fication stage:

• Random forest is a classifier containing multiple decision tree, and its output categories are determined by the mode number of the categories output by individual trees. RF is essentially a branch of machine learning – ensemble learning [24].

• Decision tree is a predictive model. It represents a mapping between object attributes and values [25].

• Extreme gradient boosting (XGBoost, short as XGB) is a kind of promotion tree model, which integrates many tree models together to form a strong classifier. The tree model used is the CART regression tree model [26].

• Deep forest is a deep learning model based on DT. It has better performance than other ensemble learning methods based on DT [27].

2. Experimental results and discussion

In this section, we conduct a comprehensive evaluation of MalFSM through detailed experiments. First, we extract opcode frequency, the size of the file and the count of lines in the file of each malicious sample from the original sample set. Secondly, various feature selection algorithms are applied to the extracted feature set, and the classification effect and model generalization ability are verified. After that, the correlation between features is visually analyzed to explain the reasons for selecting a given subset of features. We also use a malware feature image generation method. After the extracted features are transformed into grayscale images, VGG16, and transformer models are adopted to compare the classification effects. Finally, we compared MalFSM with similar studies.

1) The comparison of the experimental results based on different features

Through the analysis of Microsoft Kaggle dataset, we found that there were 735 opcodes in the sample of the training set. We also found that the data set of malware classification published by Datawhale & iFLYTEK 2021A.I. Developer Contest had the same nine malware families as the Kaggle dataset and extracted 66 opcodes ("fword" and "je" did not exist in the Microsoft dataset) and 2 metadata features [28]. To identify the impact of these features on classification performance, we used RF, DT, XGBoost, and DF. The experimental results are shown in Table 5.

Table 5. The comparison of the experimental results based on different features

Feature	Classifier															
	RF				DT				XGBoost				DF			
Performance metrics	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1
735OPC	97.65%	97.09%	95.62%	96.16%	96.55%	92.71%	92.87%	92.65%	97.50%	96.72%	95.46%	95.91%	97.85%	95.64%	95.84%	95.62%
File_size+Line_count	93.20%	89.27%	84.76%	86.13%	91.55%	85.49%	82.76%	83.77%	92.10%	88.19%	82.95%	84.63%	91.00%	90.13%	80.34%	81.65%
735OPC+File_size+Line_count	98.70%	98.46%	97.13%	97.71%	97.55%	94.36%	93.98%	94.15%	98.35%	97.97%	96.69%	97.25%	98.60%	98.29%	96.96%	97.56%
64OPC+File_size+Line_count	98.48%	98.13%	96.92%	97.45%	96.55%	92.01%	94.88%	93.19%	98.01%	97.17%	95.72%	96.35%	98.50%	98.42%	96.98%	97.62%

Note: A, P, R, and F1 stand for Accuracy, Precision, Recall, and F1-score, respectively.

As can be seen from Table 5 , the feature vector that combines Opcode and metadata features has high classification accuracy, and the similar effect of extracting all Opcode features can be achieved by extracting less than 1/10 opcode features. The accuracy comparison of different features in each classifier is shown in Fig.3. The experimental results in this section show that it is not necessary to select all Opcode as feature vectors for malicious family classification. Due to redundancy in all opcode feature sets, only a few key opcodes can be selected to achieve the desired classifica-

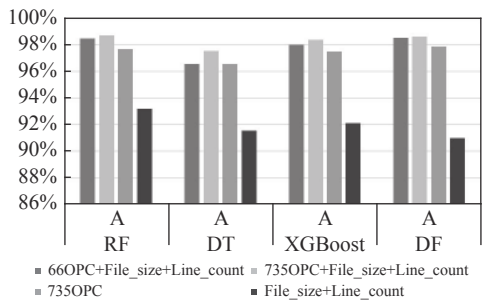


Fig. 3. The comparison of accuracy based on different features in different classifiers.

tion effect.

2) The comparison of the experimental results based on different feature selection methods

In order to select the optimal feature subset, we use six different types of feature selection methods to conduct experiments. Finally, 18 features with the highest ranking were screened out and the classification accuracy reached 98.6%. The experimental results are shown in Table 6. Feature sets are [“retn”; “cmp”; “jz”; “call”; “jnz”; “mov”; “test”; “jmp”; “and”; “or”; “sub”; “pusha”; “lea”; “jl”; “dec”; “shr”; “line_count_asm”; “size_asm”].

Table 6. The comparison of the experimental results based on different feature selection methods

Feature	Classifier															
	RF				DT				XGBoost				DF			
Performance metrics	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1
ANOVA	98.35%	98.11%	96.77%	97.36%	96.70%	91.15%	93.52%	92.12%	97.50%	97.23%	92.73%	94.29%	98.45%	98.30%	96.74%	97.45%
Chi-square	98.05%	97.79%	96.27%	96.96%	97.45%	93.97%	94.28%	94.11%	97.35%	97.06%	92.34%	94.03%	98.00%	93.42%	96.17%	94.53%
Pearson correlation coefficient	97.10%	96.45%	95.09%	95.59%	95.05%	90.74%	91.17%	90.71%	96.60%	96.01%	94.29%	94.95%	97.25%	96.65%	95.21%	95.75%
Mutual information	98.25%	98.08%	96.76%	97.34%	96.85%	96.35%	93.74%	94.77%	97.80%	97.78%	93.01%	94.73%	98.25%	98.10%	96.68%	97.32%
Random forest	98.50%	98.30%	96.97%	97.57%	97.30%	95.67%	95.53%	95.58%	97.90%	95.97%	94.74%	95.29%	98.60%	98.40%	97.04%	97.65%
Logistic regression (L2)	97.40%	97.03%	95.30%	96.08%	95.90%	92.91%	93.52%	93.14%	97.00%	94.64%	91.63%	92.87%	97.70%	95.74%	95.69%	95.66%

Note: A, P, R, and F1 stand for Accuracy, Precision, Recall, and F1-score, respectively.

As can be seen from Table 6, classification effects of different feature selection methods are similar. We first select 6 feature subsets using each feature selection algorithm, and then take their union. Finally, we select these 18 features using the random forest feature selection algorithm with the best classification effect. The accuracy comparison of these 18 features under different classifiers is shown in Fig.4. The experimental results in this section show that the feature selection algorithm is effective in family classification. Moreover, the number of extracted features can be further reduced to 1/50. On this basis, we test the generalization ability of the model, and find that the model's generalization ability is also good after reducing the number of features [29]–[32]. When fewer samples were trained (the ratio of the sample size of the training dataset to the validation dataset was 1:30), approximately 94% accuracy was achieved. The confusion matrix is shown in Fig.5. In addition, we applied these 18 features to Datawhale & iFLYTEK 2021A.I. Developer Contest dataset for experiments, and the accuracy reached 99.88% (confusion matrix is shown in Fig.6), which further verified the generalization ability of the selected feature

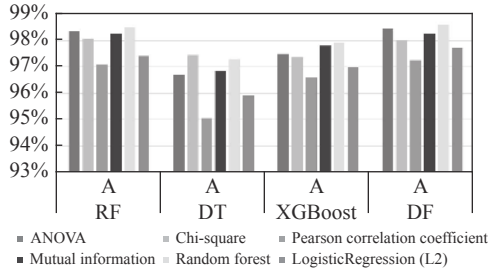


Fig. 4. The comparison of accuracy based on different feature selection methods in different classifiers.

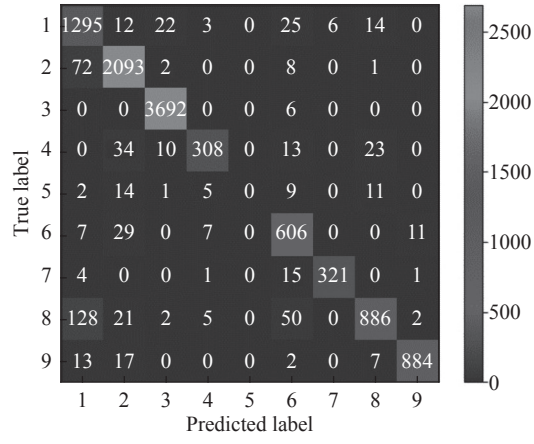


Fig. 5. Confusion matrix (train dataset: test dataset=1:30).

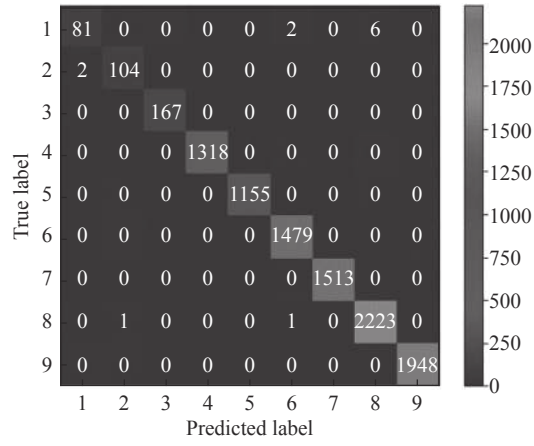


Fig. 6. Confusion matrix (Datawhale & iFLYTEK 2021A.I. Developer Contest datase).

subset [33].

3) The correlation analysis of the different features

Correlation measure between features based on Pearson correlation coefficient: In this section, we investigate the degree of correlation between variables and analyze the correlation between 18 features. The correlation matrix of these 18 features is shown in Table 7.

Table 7. The correlation matrix of different features

	retn	cmp	jz	call	jnz	mov	test	jmp	and	or	sub	pusha	lea	jl	dec	shr	line_count_asm	size_asm
retn	1.00	0.89	0.93	0.94	0.93	0.71	0.94	0.84	0.55	0.53	0.78	0.00	0.89	0.72	0.65	0.37	0.04	0.08
cmp	0.89	1.00	0.96	0.92	0.97	0.75	0.90	0.90	0.62	0.58	0.82	0.02	0.88	0.81	0.68	0.36	0.03	0.06
jz	0.93	0.96	1.00	0.95	0.96	0.72	0.96	0.87	0.57	0.55	0.79	0.00	0.89	0.81	0.69	0.36	0.02	0.05
call	0.94	0.92	0.95	1.00	0.93	0.73	0.92	0.90	0.55	0.50	0.77	0.00	0.95	0.77	0.64	0.32	0.02	0.05
jnz	0.93	0.97	0.96	0.93	1.00	0.74	0.95	0.88	0.57	0.56	0.81	0.00	0.88	0.77	0.69	0.38	0.02	0.05
mov	0.71	0.75	0.72	0.73	0.74	1.00	0.71	0.73	0.53	0.47	0.72	0.00	0.72	0.59	0.53	0.37	0.01	0.03
test	0.94	0.90	0.96	0.92	0.95	0.71	1.00	0.82	0.55	0.54	0.79	0.04	0.86	0.79	0.70	0.42	0.04	0.07
jmp	0.84	0.90	0.87	0.90	0.88	0.73	0.82	1.00	0.63	0.61	0.80	0.00	0.89	0.68	0.62	0.32	0.03	0.06
and	0.55	0.62	0.57	0.55	0.57	0.53	0.55	0.63	1.00	0.90	0.66	0.37	0.58	0.49	0.56	0.48	0.04	0.07
or	0.53	0.58	0.55	0.50	0.56	0.47	0.54	0.61	0.90	1.00	0.64	0.41	0.50	0.39	0.55	0.52	0.05	0.07
sub	0.78	0.82	0.79	0.77	0.81	0.72	0.79	0.80	0.66	0.64	1.00	0.12	0.79	0.64	0.67	0.55	0.05	0.09
pusha	0.00	0.02	0.00	0.00	0.00	0.00	0.04	0.00	0.37	0.41	0.12	1.00	0.00	0.00	0.38	0.61	-0.01	-0.01
lea	0.89	0.88	0.89	0.95	0.88	0.72	0.86	0.89	0.58	0.50	0.79	0.00	1.00	0.72	0.62	0.37	0.02	0.06
jl	0.72	0.81	0.81	0.77	0.77	0.59	0.79	0.68	0.49	0.39	0.64	0.00	0.72	1.00	0.60	0.29	0.03	0.06
dec	0.65	0.68	0.69	0.64	0.69	0.53	0.70	0.62	0.56	0.55	0.67	0.38	0.62	0.60	1.00	0.66	0.03	0.06
shr	0.37	0.36	0.36	0.32	0.38	0.37	0.42	0.32	0.48	0.52	0.55	0.61	0.37	0.29	0.66	1.00	0.02	0.03
line_count_asm	0.04	0.03	0.02	0.02	0.02	0.01	0.04	0.03	0.04	0.05	0.05	-0.01	0.02	0.03	0.03	0.02	1.00	0.99
size_asm	0.08	0.06	0.05	0.05	0.05	0.03	0.07	0.06	0.07	0.07	0.09	-0.01	0.06	0.06	0.06	0.03	0.99	1.00

As shown in Table 7 , the meaning of correlation coefficient is as follows:

- When the correlation coefficient is 0, there is no relationship between the two variables.
- When the correlation coefficient is between 0.00 and 1.00, the two variables are positively correlated.
- When the correlation coefficient is between −1.00 and 0.00, the two variables are negatively correlated.

We may safely draw the conclusion, Pearson correlation coefficient can effectively measure the correlation of features.

Visualization of feature correlation: In this section, we visualize feature correlation based on Pearson correlation coefficient. First, we select the feature subset composed of 18 features. Secondly, we measure the correlation between features. Visualization results using the heat map and scatter plot are shown below. Fig.7 shows the correlation heat map between features. The square graph at the intersection represents the correlation between two features in horizontal and vertical coordinates. The darker the color, the higher the correlation. Fig.8 shows the scatter diagram of correlation between features. A single subgraph in the graph is represented as the correlation between two features in horizontal and vertical coordinates. The graph shows an upward slope, indicating a positive correlation, that is, as one variable increases, the other increases proportionally. A slope of 1 is a perfect positive correlation. And vice versa. A slope equal to 0 means that there is no relationship between the two variables, and the data points are scattered throughout the graph.

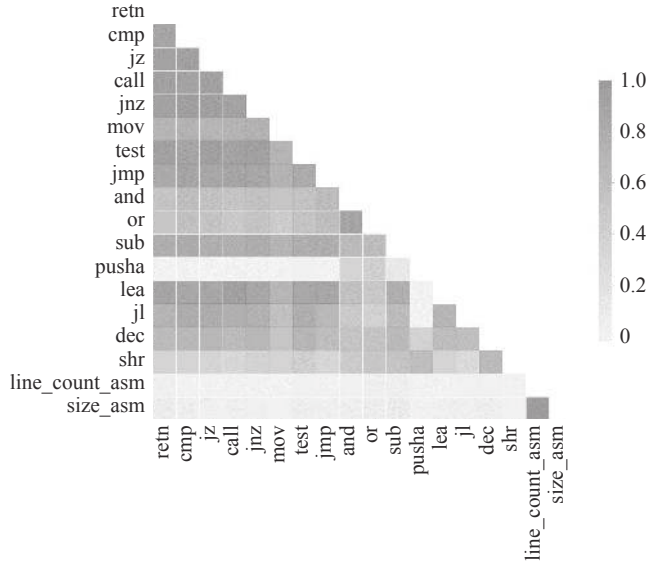


Fig. 7. The heat map of feature correlation.

4) The comparison of the experimental results based on the deep learning models

In this section, we use the deep learning modesl to conduct a family classification experiment on malicious samples. First, the sample feature vectors formed according to the 66 features extracted in Section V.2.1) are transformed into gray scale images. For example, Fig. 9 is the grayscale representation of the malicious

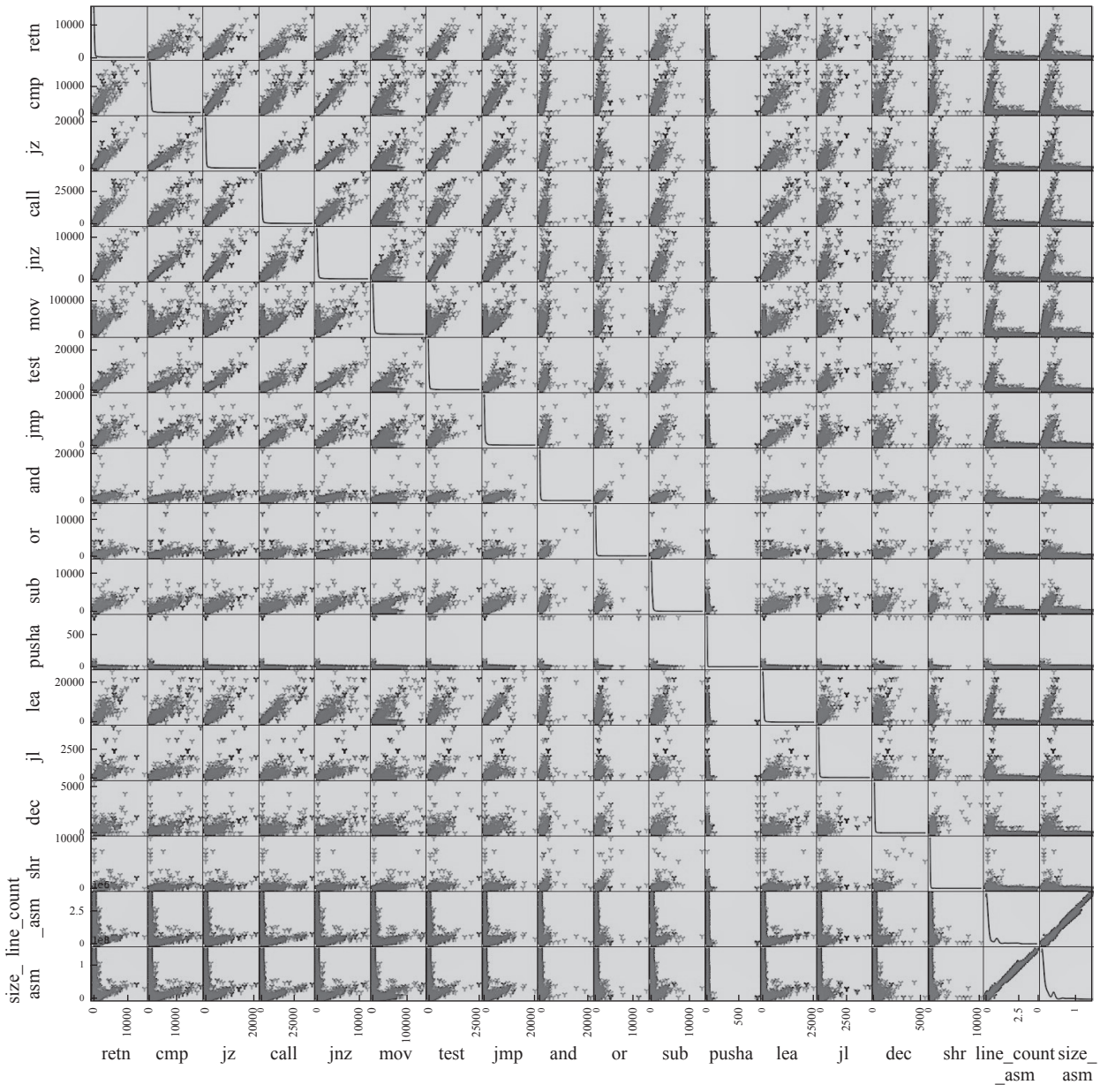


Fig. 8. The scatter plot of feature correlation.

Fig. 9. Grayscale image of "0AnoOZDNbPXIr2MRBSCJ.asm".

sample "0AnoOZDNbPXIr2MRBSCJ.asm." After that, the popular VGG16 and transformer models are adopted for image classification. The experimental results are shown in Table 8. As shown in Table 8, it can be concluded that the common deep learning model consumes a lot of time in the process of model training and detection, and its accuracy is lower than directly using ML classifiers to classify feature vectors of malwares. Possible reasons are as follows:

a) some important features will be lost when malicious samples are transformed into grayscale images;

b) image classification on memory and computational cost is high, long time consuming [34]. The key of

Table 8. The experimental results based on the deep learning models

Classifier	Metrics			
	Validation_loss	Validation_accuracy	Train_accuracy	Consuming time
VGG16	0.1345	96.15%	87.31%	1990.65 s
Transformer	0.1023	97.57%	97.64%	342.95 s

deep learning lies in the ability of learning features and huge models [35]. Compared with deep neural network, deep forest has fewer hyperparameters and higher performance.

Figs. 10 and 11 show the training and validation accuracy graph of VGG16 and transformer respectively. It is worth mentioning that although the transformer model has a higher epochs than VGG16, it still consumes less time than VGG16, probably because the transformer model can be calculated in parallel [36], [37].

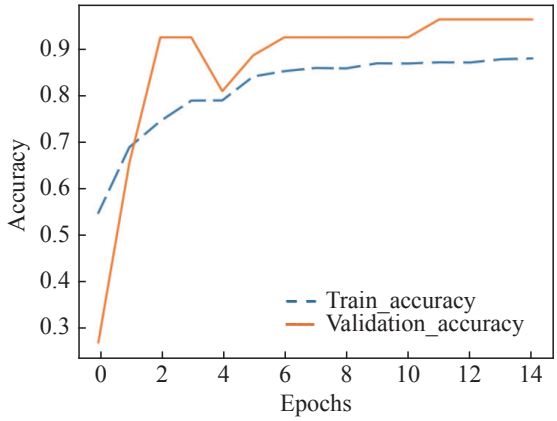


Fig. 10. Training and validation accuracy graph of VGG16.

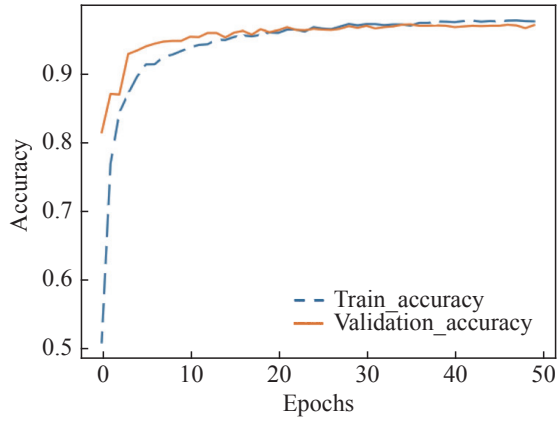


Fig. 11. Training and validation accuracy graph of Transformer.

5) Comparison with similar studies

In this section, we compared MalFSM with similar studies from the aspects of accuracy, features, time cost and occupied space. As the dataset we used was the Kaggle dataset, we selected similar studies using the same dataset for comparison, and the comparison results are shown in Table 9.

Table 9. Comparison with similar methods

	MalFSM	Extract all opcode and metadata features	M. Ahmadi *et al.* [16]	A. Darem *et al.* [19]	E. Raff *et al.* [38]	Q. Le *et al.* [22]
Dataset	The Kaggle malware dataset from the Microsoft Malware Classification Challenge					
Num of features	18	737	1804	4358+Malware images	–	10,000
Classification accuracy (%)	98.60	98.80	99.77	99.12	97.80	98.20
Feature extraction time (s)	5907.27	5889.16	183477	1023 (training time)	32087.4 (training time)	6372 (training time)
Model classification time (s)	7.76	24.9	–	15	804.65	214.32
Occupied space (KB)	1,043	15,120	–	–	–	–

Compared with similar studies, MalFSM has the following advantages: 1) MalFSM achieves an optimal compromise between feature space and classification accuracy. The length of the feature vector constructed by MalFSM is 18, and the classification accuracy is 98.6%. On the one hand, although the classification accuracy is slightly lower than similar studies, it can fully meet the requirements. On the other hand, the feature vectors required by MalFSM are the most concise in other similar studies, so it can provide promising classification results under the condition of reducing the complexity of feature engineering. 2) MalFSM can effectively reduce the time and space occupation of model classification. Compared with extracting all opcode and Metadata features, the time is similar, the efficiency of classification is improved by 69%, and the space occupation is reduced by 93.1%; MalFSM required the shortest processing time compared to similar studies. 3) The hardware platform required by MalFSM has moderate performance and good universality. 4) By extracting and mining the opcode and metadata information of malwares, MalFSM provides the analysis of the correlation between the opcode features of malwares, which can be extended to the homologous malware family classification.

VI. Conclusions and Future Work

Detection and classification of malware is a complex process that requires selecting a subset of discriminatory features from a dataset. In this paper, we filter out relevant feature subset by comparing a series of feature selection algorithms to remove redundant features.

In addition, the correlation between opcode features is analyzed to explain the selection of feature subset. For this purpose, we propose the MalFSM framework, which features the identification of correlations between features, the ranking of selected features subset, and the prediction of the performance of the selected features subset on different classifiers. In industrial applications, the trade-off between complexity and performance can be a key issue. Using a purified subset of features saves time in feature extraction and malware family classification, and it is necessary to collect core common features of malware samples.

The results show that the framework can construct a lightweight feature matrix according to the selected opcodes as well as file_size and line_count features. Experimental results show that the lightweight eigenmatrix can not only reduce the time and space complexity, but also achieve the spatio-temporal equilibrium, and the classification effect is not bad. The main contribution of this paper is to analyze the correlation among opcode features, which provides a reasonable explanation for researchers to select a subset of features. Compared with the extraction process of complete opcode features, the time consumption and space occupation of model classification are reduced and the analysis efficiency is significantly improved.

In the future, we will focus on the following three directions [39]: 1) Research on dynamic analysis and hybrid analysis of malware; 2) Extracting various features of malware for multi-dimensional analysis; 3) Research on adversarial attack and robustness analysis of malicious codes [40], [41].

References

[1] Christiaan Beek, Sandeep Chandana, Taylor Dunton, *et al.*, "McAfee Labs threat report: November 2020," available at: *https://www.mcafee.com/enterprise/zh-cn/assets/reports/rp-quarterly-threats-nov-2020.pdf*, 2020-11-20.

[2] W. He, "The October 2021 malware heinous list," available at: *https://www.easemob.com/news/7467*, 2021-11-23.

[3] H. Zhou, W. Zhang, F. Wei, and Y. Chen, "Analysis of Android malware family characteristic based on isomorphism of sensitive API call graph," in *Proceedings of 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, Shenzhen, China, pp.319–327, 2017.

[4] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variant detection," *IEEE Trans. Dependable and Secure Comput*, vol.11, no.4, pp.307–317, 2014.

[5] W. Hu and Y. Tan, "Black-box attacks against RNN based malware detection algorithms," in *Proceedings of the Workshops of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA, pp.245–251, 2018.

[6] W. Han, J. Xue, Y. Wang, *et al.*, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Computers & Security*, vol.83, pp.208–233, 2019.

[7] C. Wu and W. Li, "Enhancing intrusion detection with feature selection and neural network," *International Journal of Intelligent Systems*, vol.36, no.7, pp.3087–3105, 2021.

[8] Kemal Polat and Salih Güneş, " A new feature selection method on classification of medical datasets: Kernel F-score feature selection," *Expert Systems with Applications*, vol.36, no.7, pp.10367–10373, 2009.

[9] J. Benesty, J. Chen, and Y. Huang, "On the importance of the Pearson correlation coefficient in noise reduction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol.16, no.4, pp.757–765, 2008.

[10] X. Zheng, Y. Wang, L. Jia, *et al.*, "Network intrusion detection model based on Chi-square test and stacking approach," in *Proceedings of 2020 7th International Conference on Information Science and Control Engineering (ICISCE)*, Changsha, China, pp.894–899, 2020.

[11] S. Tan, X. Zhang, Q. Li, and A. Chen, "Information push model-building based on maximum mutual information coefficient," *Journal of Jilin University Engineering and Technology Edition*, vol.48, no.2, pp.558–563, 2018. (in Chinese)

[12] M. Cuturi and A. D'Aspremont, " Mean reversion with a variance threshold," in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, USA, pp.III-271–III-279, 2013.

[13] Moutaz Alazab, " Automated malware detection in mobile app stores based on robust feature generation," *Electronics*, vol.9, no.3, article no.435, 2020.

[14] K. Yan and D. Zhang, " Feature selection and analysis on correlated gas sensor data with recursive feature elimination," *Sensors and Actuators B: Chemical*, vol.212, pp.353–363, 2015.

[15] P. Zhang, "A novel feature selection method based on global sensitivity analysis with application in machine learning-based prediction model," *Applied Soft Computing* , vol.85, article no.105859, 2019.

[16] M. Ahmadi, D. Ulyanov, S. Semenov, *et al.*, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, New Orleans, LA, USA, pp.183–194, 2016.

[17] S. Ni, Q. Qian, and R. Zhang, " Malware identification using visualization images and deep learning," *Computers & Security*, vol.77, pp.871–885, 2018.

[18] W. Han, J. Xue, Y. Wang, *et al.*, "MalInsight: A systematic profiling based malware detection framework," *Journal of Network and Computer Applications*, vol.125, pp.236–250, 2019.

[19] A. Darem, J. Abawajy, A. Makkar, *et al.*., "Visualization and deep-learning-based malware variant detection using OpCode-level features," *Future Generation Computer Systems*, vol.125, pp.314–323, 2021.

[20] I. Almomani, A. AlKhayer, and M. Ahmed, " An efficient machine learning-based approach for Android v.11 ransomware detection," in *Proceedings of 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, Riyadh, Saudi Arabia, pp.240–244, 2021.

[21] G. Sun and Q. Qian, " Deep learning and visualization for identifying malware families," *IEEE Transactions on Dependable and Secure Computing*, vol.18, no.1, pp.283–295, 2021.

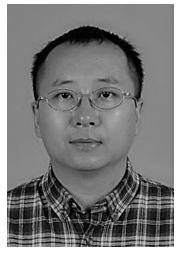
[22] Q. Le, O. Boydell, B. Mac Namee, *et al.*, "Deep learning at the shallow end: Malware classification for non-domain experts," *Digital Investigation*, vol.26, pp.S118–S126, 2018.

[23] X. Hu, J. Jang, T. Wang, *et al.*, "Scalable malware classification with multifaceted content features and threat intelligence," *IBM Journal of Research and Development* , vol.60, no.4, pp.6:1–6:11, 2016.

[24] M. Masum, M.J. Hossain Faruk, H. Shahriar, *et al.*, "Ransomware classification and detection with machine

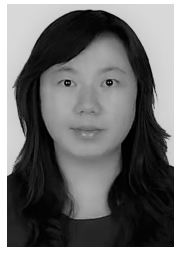
learning algorithms," in *Proceedings of 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, pp.0316–0322, 2022.
[25] S. Jain, T. Khandelwal, Y. Jain, *et al.*, "Android malware analysis using machine learning classifiers," in *Proceedings of International Conference on Computational Intelligence and Emerging Power System*, Singapore, pp.171–179, 2022.
[26] J. Bao, " Multi-features based arrhythmia diagnosis algorithm using Xgboost," in *Proceedings of 2020 International Conference on Computing and Data Science (CDS)*, Stanford, CA, United States, pp.454–457, 2020.
[27] Z. Zhou and J. Feng, "Deep forest," *National Science Review*, vol.6, no.1, pp.74–86, 2019.
[28] iFLYTEK, "Malware classification challenge," available at: *https://challenge.xfyun.cn/topic/info?type=malware-classification*, 2021-08-02.
[29] K. Xu, Y. Li, R. Deng, *et al.*, "DroidEvolver: Self-evolving Android malware detection system," in *Proceedings of IEEE European Symposium on Security and Privacy (EuroS & P)*, Stockholm, Sweden, pp.47–62, 2019.
[30] H. Cai, " Assessing and improving malware detection sustainability through App evolution studies," *ACM Trans. Softw. Eng. Methodol*, vol.29, no.2, pp.1–28, 2020.
[31] X. Fu and H. Cai, "On the deterioration of learning-based malware detectors for Android," in *Proceedings of IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, Montreal, QC, Canada, pp.272–273, 2019.
[32] H. Cai and J. Jenkins, "Towards sustainable Android malware detection," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, Gothenburg, Sweden, pp.350–351, 2018.
[33] H. Cai, " Embracing mobile App evolution via continuous ecosystem mining and characterization," in *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, Seoul, Republic of Korea, pp.31–35, 2020.
[34] T. Han, L. Zhang, and S. Jia, "Bin similarity based domain adaptation for fine-grained image classification," *International Journal of Intelligent Systems*, vol.37, no.3, pp.2319–2334, 2021.
[35] M. R. Minar and J. Naher, "Recent advances in deep learning: An overview," *arXiv preprint*, arXiv: 1807.08169, 2018.
[36] E. Rezende, G. Ruppert, T. Carvalho, *et al.*, "Malicious software classification using VGG16 deep neural network's bottleneck features," in *Information Technology - New Generations, Advances in Intelligent Systems and Computing, vol.738*, Springer, Cham, pp.51–59, 2018.
[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *The Ninth International Conference on Learning Representations (ICLR 2021 Oral)*, Virtual Event, article no.1909, 2021.
[38] E. Raff, R. Zak, R. Cox, *et al.*, "An investigation of byte n-gram features for malware classification," *J Comput Virol Hack Tech*, vol.14, no.1, pp.1–20, 2018.
[39] G. Suarez-Tangil and G. Stringhini, " Eight years of rider measurement in the Android malware ecosystem," *IEEE Transactions on Dependable and Secure Computing*, vol.19, no.1, pp.107–118, 2022.
[40] A. Al-Dujaili, A. Huang, E. Hemberg, *et al.*, "Adversarial deep learning for robust detection of binary encoded malware," in *Proceedings of 2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, pp.76–82, 2018.
[41] C. Agarwal, A.M. Nguyen, and D. Schonfeld, "Improving robustness to adversarial examples by encouraging discriminative features," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Taipei, China, pp.3801–3505, 2019.

KONG Zixiao was born in 1996. She takes a successive postgraduate and doctoral program at Beijing Institute of Technology, majored in Cyberspace Security. Her research interests include cyber security and machine learning. She has a B.S. degree in software engineering.
(Email: 3120185534@bit.edu.cn)

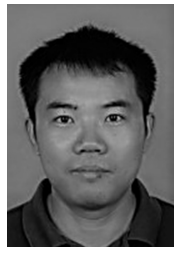


XUE Jingfeng was born in 1975. He is a Professor and Ph.D. Supervisor in Beijing Institute of Technology. His main research interests focus on network security, data security, and software security.
(Email: xuejf@bit.edu.cn)

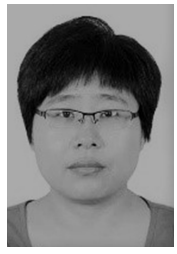
WANG Yong was born in 1975. She is an Associate Professor of Beijing Institute of Technology. Her main research interests focus on cyber security and machine learning.
(Email: wangyong@bit.edu.cn)



ZHANG Qian (corresponding author) was born in Inner Mongolia, China, in 1986. She graduated from the School of Software, Beijing Institute of Technology (BIT) in 2012. She is an Assistant Experimentalist of BIT now, and her research interests include software security and software testing.
(Email: zhangqian16@bit.edu.cn)



HAN Weijie received the B.E. and M.E. degrees from Space Engineering University in 2003 and 2006, respectively, and received the Ph.D. degree from BIT in 2020. He is currently a Lecturer in Space Engineering University. His current research interest includes malware detection and APT detection.
(Email: bit_hwj2016@126.com)



ZHU Yufen received the B.E. degree in 2006. She is currently an Engineer in the Software Evaluation Center of Beijing Institute of Technology. Her research interests include malware analysis and network anomalies detection.
(Email: visc_hwj@126.com)