applied sciences

MDPI

Article

An Efficient Boosting-Based Windows Malware Family Classification System Using Multi-Features Fusion

Zhiguo Chen [1,2,*] and Xuanyu Ren [1,2]

[1] Engineering Research Center of Digital Forensics, Nanjing University of Information Science and Technology, Ministry of Education, Nanjing 210044, China; renxuanyu@nuist.edu.cn

[2] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

* Correspondence: chenzhiguo@nuist.edu.cn

Abstract: In previous years, cybercriminals have utilized various strategies to evade identification, including obfuscation, confusion, and polymorphism technology, resulting in an exponential increase in the amount of malware that poses a serious threat to computer security. The use of techniques such as code reuse, automation, etc., also makes it more arduous to identify variant software in malware families. To effectively detect the families to which malware belongs, this paper proposed and discussed a new malware fusion feature set and classification system based on the BIG2015 dataset. We used a forward feature stepwise selection technique to combine plausible binary and assembly malware features to produce new and efficient fused features. A number of machine-learning techniques, including extreme gradient boosting (XGBoost), random forest, support vector machine (SVM), K-nearest neighbors (KNN), and adaptive boosting (AdaBoost), are used to confirm the effectiveness of the fusion feature set and malware classification system. The experimental findings demonstrate that the XGBoost algorithm's classification accuracy on the fusion feature set suggested in this paper can reach 99.87%. In addition, we applied tree-boosting-based LightGBM and CatBoost algorithms to the domain of malware classification for the first time. On our fusion feature set, the corresponding classification accuracy can reach 99.84% and 99.76%, respectively, and the F1-scores can achieve 99.66% and 99.28%, respectively.

Keywords: computer security; machine learning; malware classification; feature fusion

check for updates

Citation: Chen, Z.; Ren, X. An Efficient Boosting-Based Windows Malware Family Classification System Using Multi-Features Fusion. *Appl. Sci.* **2023**, *13*, 4060. https://doi.org/10.3390/app13064060

Academic Editor: Luis Javier García Villalba

Received: 10 March 2023
Revised: 20 March 2023
Accepted: 21 March 2023
Published: 22 March 2023

1. Introduction

Malware is software or code that is downloaded and executed on computers or other electronic devices without explicitly prompting or receiving permission to infringe the proper rights and interests of clients. Starting in August 2020, the Lemon Duck [1] crypto mining botnet spread at an unbelievable rate. It used multiple storm breaches, such as SMB storm, RDP storm, and SQL server storm, as well as other vulnerabilities, including the USBLnk vulnerability and the Eternal Blue vulnerability, for propagation. The research of S. Greengard [2] shows that most malware variants contain very small mutations, with less than 2% code variation among them. It shows that the majority of malware samples from the same family have some characteristics. Thus, computer security depends on being able to classify malware variations swiftly and effectively in the face of an infinite influx of them.

There are two main approaches for malicious code analysis: dynamic analysis and static analysis. By watching the program's execution in a controlled environment, dynamic analysis can determine the program's actual behavior when it is in use (e.g., virtual machine, emulator, or the sandbox). During the monitoring process, the operations completed by the program (such as file access, API calls [3–6], communication through networks, etc.) are recorded as reports, and researchers can effectively classify malware by analyzing the characteristics in the reports [7–11]. Alshamrani [12] proposed a machine learning-based malicious program classification system using a dynamic analysis approach. This system efficiently detects PDF malicious files by examining the dynamic execution of a given sample. A novel framework for identifying malicious programs was proposed by Wang et al. [13]. The framework mainly converts the dynamic execution sequence of the original API into a graph of function calls, which preserves majority of the original data and shrinks the input sequence size. The experimental outcomes demonstrate that the framework's F1-score and detection accuracy can reach 97.5% and 97.8%, respectively. Catak et al. [14] analyzed 7107 malicious software applications, including Trojans, worms, and viruses from various families. In a safe sandbox environment, they recorded the Windows operating system's API calls. Then, converted the sequential analysis results into a format that can be used in different classification algorithms. In general, dynamic analysis's main benefit is to try to discover all the actual behaviors of the program, detect malware according to the program's behavior, and not be affected by encryption and obfuscation technology. Thus, it could increase the accuracy of malware classification and identify unknown and variant malware. Nevertheless, because it is difficult to acquire the behavioral features of every execution route and because it uses many resources and much time, this method can only be used to examine the behavior of a single execution path [15]. In addition, recent malware often employs counter-analysis defense mechanisms that can hide malicious behavior in simulated environments [16] and lack of code coverage.

Static analysis finds malicious patterns by decomposing code and exploring the control flow of executable files. Static analysis can achieve full code coverage and typically analyze malware bytecode, assembly code, pixel features, file structure, control flow graphs, syntax library calls, etc. For instance, Du et al. [17] used recursive feature elimination (RFE) and principal component analysis (PCA) algorithms for static feature extraction, thus proposing an intelligent machine-learning algorithm based on KNN and density for detecting ransomware pre-attacks. The suggested technique can effectively increase the accuracy of malware detection, according to experimental data. Gu et al. [18] to extract static features, a convolutional neural network was utilized, deeply analyzed the development status and existing problems of information communication and network security, and adopted a process that uses reinforcement learning to ensure high network security. The experiment's findings demonstrate that this system performs superior to existing algorithms in this field. Ahmadi et al. [19] were the earliest to employ the BIG2015 dataset for malware classification, extracting malware binary and disassembly files' static features and combining multidimensional features to classify nine malware families with a 99.77% classification accuracy. Anderson et al. [20] proposed a supervised static detection model for Windows malware. On a sizable and labeled dataset (EMBER), the model is tested and trained based on the tree structured LightGBM classification algorithm. The experimental results show that using the initialized LightGBM classifier performs better than the unsupervised deep learning classification model on the same feature set. At the same time, the EMBER dataset, as the first standard dataset in the field of detection of malware, is beneficial to the research of malware detection in the field of machine learning.

Although static analysis can achieve full path coverage without polluting the system, low detection rates and a high percentage of false positives are issues, and most malware classification systems use many different types of features, resulting in high system memory overhead and low detection efficiency. Given this, we proposed an efficient boosting-based Windows malware family classification system using multi-features fusion. This system performed feature fusion based on limited feature categories to reduce the time and memory overhead and improve the classification accuracy. Meanwhile, this system adopted an effective approach for classifying malware using tree boosting-based machine-learning algorithms, including XGBoost v1.5.1, LightGBM v3.3.5, and CatBoost v3.3.5. The classification accuracy of the XGBoost algorithm especially can be as high as 99.87%, and the logarithmic loss can be reduced to 0.007.

The following constitutes the paper's original contribution:

(i) Combined plausible binary and assembly malware features using a forward feature stepwise selection method to produce the most efficient fused features. Experimental findings demonstrate that the XGBoost classifier's classification accuracy using the proposed fused features is 99.87%, which is higher than most existing malware classification models.
(ii) Using a limited number of 5530 features makes the system much faster with high accuracy and easier to use for large-scale malware classification tasks.
(iii) The LightGBM and CatBoost-based models were applied for the first time to process malware classification tasks. Both models obtained over 99.7% accuracy.

Here is the remainder of the essay: An overview of the connected works is given in Section 2. The specifics of the suggested methodology are presented in Section 3. The experimental findings are covered in Section 4, and this paper's conclusions and recommendations are provided in Section 5.

2. Related Work

2.1. Portable Executable (PE)

The PE file format describes the executable formats of the Microsoft Windows operating system, including executables, Dynamic Link Libraries (DLLs), and FON font files. A Windows binary file consists of a PE header, code, data, and resource sections. The PE header includes a common object file format (COFF) file header, an optional header, and a section table. Each section has one or more subsections. The COFF header is 24 bytes long and contains crucial details about the file, such as the type of system it is compatible with, its nature (DLL, EXE, OBJ), how many sections and symbols it contains, etc. The linker version, code size, initialized data, uninitialized data, entry point location, code base, etc. are all included in the optional header. Many tables and table sizes, including location and size export tables, import tables, resource tables, exception tables, debug information tables, certificate information tables, and relocation tables, are included in the optional header's data directory. Finally, each section table is 40 bytes long and includes details such as the name, virtual size, address, position, raw data size, number of relocations, number of line numbers, attributes, etc. A binary file can be more than one section, and the section names include .text, .rdata, .data, .idata, .rsro, .rsrc, etc.

2.2. Gradient Boosting Decision Tree (GBDT)

GBDT is an ensemble algorithm based on decision tree implementation, which is crucial for data analysis and prediction. It has many efficient implementations, such as XGBoost, LightGBM, and CatBoost, which use an ensemble strategy to ensemble a more efficient strong classifier from several weak classifiers. Moreover, the algorithm based on the tree structure is more interpretable than other algorithms, which is beneficial for malware analysts to analyze and study malware. These three algorithms are described in detail below.

The gradient boosting framework, which uses the machine-learning algorithm XGBoost, is extremely expandable, adaptable, and versatile. It proposed by this algorithm efficiently implements the GBDT algorithm and makes many improvements in algorithm and engineering, which was proposed by Tianqi Chen in March 2014. It overcomes the limitations of previous gradient lifting algorithms and improves the training speed and accuracy of the model's predictions. Therefore, XGBoost has been applied in various machine-learning competitions, including the Microsoft Malware Challenge, with excellent results. In contrast to existing gradient enhancement algorithms, XGBoost uses a novel regularization strategy to reduce overfitting. When tweaking a model, it is quicker and more reliable. The regularization technique introduces an extra term to the loss function, as Equations (1) and (2) show:

$$\mathrm{L}(\mathrm{f}) = \sum_{i=1}^{n} \mathrm{L}(\overset{\Lambda}{y}_i, y_i) + \sum_{m=1}^{M} \Omega(\delta_m) \tag{1}$$

$$\Omega(\delta) = \alpha|\delta| + 0.5\beta||w||^2 \tag{2}$$

where $|\delta|$ is the number of branches, w is the value of each leaf, and Ω is the regularization function. Compared to the previous gradient enhancement algorithm, XGBoost employs a new gain function as Equations (3)–(7) show:

$$G_j = \sum_{i \in I_j} g_i \tag{3}$$

$$H_j = \sum_{i \in I_j} h_i \tag{4}$$

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \beta} + \frac{G_R^2}{H_R + \beta} - \frac{(G_R + G_L)^2}{H_R + H_L + \beta}\right] - \alpha \tag{5}$$

$$g_i = \partial_{\overset{\Lambda}{y}i} L\left(\overset{\Lambda}{y}_i, y_i\right) \tag{6}$$

$$h_i = \partial^2_{\overset{\Lambda}{y}i} L\left(\overset{\Lambda}{y}_i, y_i\right) \tag{7}$$

G is the cumulative sum of first-order partial derivatives, H is the cumulative sum of second-order partial derivatives, and Gain is the score in the absence of the new child [21].

Due to the high interpretability and robustness of the GBDT algorithm, Microsoft released the first stable version of LightGBM in January 2017 [22]. In XGBoost, the tree is grown by level, called level-wise tree growth. Branching and pruning occur at all nodes at the same level, as shown in Figure 1a. The level-wise indiscriminate treatment of leaves at the same level creates a great deal of pointless overhead. There is no need to hunt for and split many leaves because they have poor splitting gains. By contrast, in addition to the histogram algorithm, Figure 1b illustrates how LightGBM divides after identifying the leaf with the highest splitting gain out of all the current leaves. Therefore, contrasted with level-wise, leaf-wise can reduce the memory overhead of the system with the same number of splits. In addition, LightGBM uses the gradient-based one-side sampling (GOSS) method to delete most small gradient samples according to the data gradient.

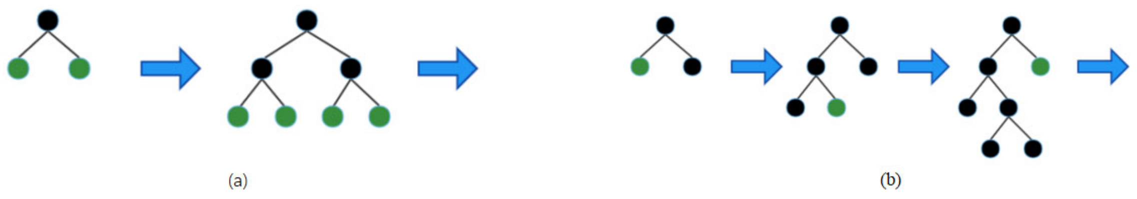

Figure 1. (**a**) XGBoost level-wise tree growth; (**b**) LightGBM leaf-wise tree growth.

As one of the mainstream algorithms of GBDT, CatBoost is also a parallel implementation under the GBDT framework. The Russian company Yandex open-sourced it in 2018. Compared with the traditional machine-learning algorithms, there are two differences: (i) classified features can be processed without feature engineering before training the model; (ii) it lessens model overfitting and improves model prediction effect. CatBoost handles category features differently from LightGBM and converts category features into numeric types as follows:

(I) Dividing the training samples into random subsets.
(II) Converting label values into integers.
(III) Iterating through each sample in turn and converting the category features to the data type according to the following formula.

$$\text{valueNew} = \frac{\text{countlnClass} + \text{prior}}{\text{totalCount} + 1} \tag{8}$$

where valueNew: the converted value; countlnClass: the total number of samples that have been visited to reach the current sample and have the same label value; totalCount: the total number of samples traversed to the current sample; and prior: the smoothing factor (specified by the starting parameter) [23].

2.3. Forward Selection Algorithm

The forward selection procedure is a method for choosing independent variables in regression models that introduces potential independent variables to the regression equation one at a time [24]. Initially, it fits the independent variable with the highest correlation to the dependent variable y to the model, and the inclusion of this independent variable in the model is determined by the significance test of the regression coefficient. Secondly, repeat the above process among the independent variables not introduced into the model. Finally, this process continues until none of the significance tests of the regression coefficients of the unselected independent variables on y are significantly different from zero after excluding the effect of the selected variables on y. The forward selection algorithm uses the idea of the greedy algorithm, which is to reduce the error at each step until the end of the final iteration.

2.4. Visualizing Malware as an Image

In the area of identifying malware, the classification of malware using image visualization techniques is becoming increasingly popular. This approach mainly detects and classifies malware by converting binary sequences into image representations. To visualize a malware sample as an image, a vector of 8-bit unsigned numbers from the provided malware binary file is read and then transformed into a two-dimensional array [25]. This array is visualized as a grayscale image in the range [0, 255] (0: black, 255: white). The height and width of the image can be adjusted according to the file size, as shown in Figure 2.

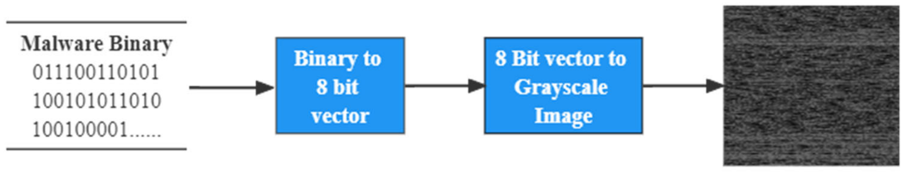

Figure 2. Visualizing malware as an image.

Malware has increasingly been turned into visuals for detection and categorization. A deep learning-based malware categorization method was proposed by Tekerek et al. [26]. The system converts the binary sequences of suspect software into grayscale images. The BIG2015 dataset's classification precision on the CNN classifier is 99.86%. Dai et al. [27] proposed a method to classify malware using hardware features by converting memory dump files into grayscale images and extracting features using a histogram of gradients (HOG). Gibert et al. [28] presented a file-independent deep-learning method for malware classification. This is prompted by the visual resemblance of malware samples from the same series. Using the grayscale graph, this technique extracted a number of discriminative patterns to effectively group malware into the corresponding series.

2.5. Malware Detection and Classification

Traditional signature scanning-based techniques cannot effectively detect malware in real-time malware variants. To overcome these problems, many researchers have established static malware detection models based on techniques for machine learning by extracting static features, such as Windows PE file structure, opcodes, bytecodes, etc., before software execution. The types of columns in Table 1 show whether the study has an application for identifying or categorizing malware. If characteristics were taken from the PE body or header, it is indicated in the features column. All abbreviations are explained below:

BYT: byte code; OP: operation code; STC: structural feature; API: application programming interface. Finally, the model column shows the classification models used by the author.

Table 1. Methods of static analysis for malware on Windows.

Year	Authors	Types		Features		Models
		Det	Class	Header	Body	
2019	Sun et al. [29]	√		-	OP	Machine Learning
2020	Ijaz et al. [30]	√		STC	STC	Gradient Boosting
2021	Loi et al. [5]	√	√	BYT	STC	Automated Pipeline
2021	Hemalatha et al. [31]		√	BYT	BYT	DenseNet
2021	Sun et al. [32]		√	-	OP	RMVC
2022	Li et al. [33]		√	BYT	BYT	SAE
2022	Kumar et al. [34]		√	BYT	BYT	SVM
2022	Jadvani et al. [35]	√		STC	-	Random Forest
2022	Greengard et al. [2]	√		API	-	Machine Learning
2022	Johnson et al. [4]	√		-	OP	Ensemble Learning
2022	Rizvi et al. [3]	√		STC	-	FANN

In 2019, Sun et al. [29] advanced a static detection method to identify Windows malicious programs based on opcode sequences. This method identifies Windows malware of varying digits. Compared to previous systems, it reduces computational complexity and can efficiently detect and classify malware variants. In 2020, Ijaz et al. [30] used APIs, byte entropy, DLLs, and altered registry entries to statically extracted 92 features from binary malware, obtaining 99.36% detection accuracy. In 2021, Loi et al. [5] proposed an automated pipeline for detecting and classifying PE files. The pipeline first detects whether the sample is malicious. In the case of malware, the pipeline further determines the group of samples. Experimental findings indicate that the scheme achieves 96.9% classification accuracy on the EMBER dataset.

An effective malware categorization method built on a deep learning algorithm was proposed by Hemalatha et al. [31]. The suggested approach addresses the issue of unbalanced data by applying a weighted loss function in the neural network model's classification layer. With an accuracy of 98.23% on the MalImg dataset, 98.46% on the BIG2015 dataset, and 98.21% on the MaleVis dataset, the testing findings demonstrate that it may greatly improve the malware classification performance. A technique (RMVC) based on opcode sequences for categorizing malware was suggested by Sun et al. [32]. The system first converts malicious programs into grayscale images, then uses recurrent neural networks (RNN) for feature extraction, and finally uses convolutional neural networks (CNN) for image classification. The outcomes of the trial indicate that the RMVC model achieves 99.5% accuracy on a few samples. In 2022, using the assembly files and binary files of the Kaggle dataset, Li et al. [33] extracted 184 opcode features and 16 probabilistic features. They then fused all the features together using a double-byte feature encoding technique. The CNN algorithm was employed in the experiment to classify the fused samples, and the findings show that the suggested method has an accuracy rate of 98.68% and a logloss of 0.022. Kumar et al. [34] suggested a useful classification scheme based on malware texture features, which does not need to execute malware and effectively avoids terminal infection. The classification accuracy using a machine-learning classifier on the public MalImg dataset was 98.34%.

Jadvani et al. [35] proposed a method to predict malicious executable files using PE file header features. This method uses a limited number of features, which reduces the system's complexity while ensuring high accuracy. Using a dataset with features in PE file format, the model was trained and tested with an average accuracy of roughly 99.5%. A malware detection method based on neural networks and PE file API calls was proposed by Greengard et al. [2]. The experimental results show that the average accuracy rate can reach 97.9%. Johnson et al. [4] presented a static detection method incorporating machine learning to classify ransomware using opcode sequences extracted from assembly files.

The proposed integrated learning model has significantly enhanced performance when testing a dataset consisting of real-time ransomware samples with 99.21%. A malware detection system based on static analysis was proposed by Rizvi et al. [3]. The system is trained on pseudo labels based on unsupervised learning. The experimental findings demonstrate that the deep neural network's accuracy is 98.09% on the 15,457 PE samples, and the time efficiency and detection accuracy of the system are better than most traditional static malware detection systems. At the same time, the system reduces the risk of malware infection to the lowest.

Table 1 summarizes the comparison of feature types and models employed in machine learning-based malware static detection or classification schemes. As can be seen from Table 1, the static malware detection scheme has become more and more perfect, and it has excellent detecting performance or efficiency and accuracy of classification. Researchers mainly use software PE structure information, opcode sequences, bytecode sequences, application program interfaces, and other static features to establish a malware system for static detection and classification.

3. The Proposed System

3.1. Overview of the Proposed System

Figure 3 depicts the architecture of the proposed static malware categorization system. This classification system utilizes the proposed fusion feature set and machine learning-based algorithm to increase malware classification effectiveness. The system essentially consists of four components: (1) Malware samples, (2) Feature extraction, (3) Feature fusion, and (4) Classification model. Malware samples: This study investigates the categorization of malware using the benchmark dataset (BIG2015) made available by the Microsoft Malware Challenge. It contains 10,868 malware, and each malware has two formats: binary language format and assembly language format. Feature extraction: We extracted six groups of representative features based on the BIG2015 dataset and stored them in the feature database for future analysis. Feature fusion: A forward stepwise selection algorithm is used to fuse the extracted features to achieve a balance between model classification accuracy and time efficiency. Classification model: Used tree boosting-based machine-learning algorithm to build an effective malware classification system.

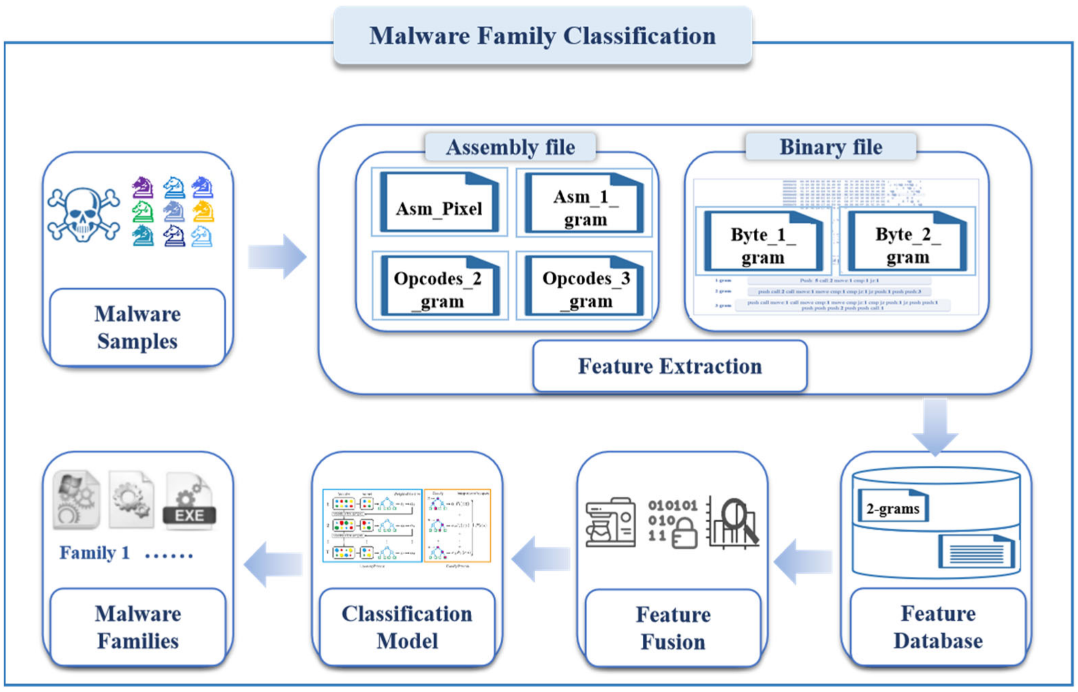

Figure 3. Overview of our proposed architecture.

3.2. Malware Representation

Before detailing the features, we will describe two representations of the BIG2015 dataset's malware samples: the binary form and the assembly language form obtained by decompiling with the Interactive Disassembler (IDA Pro).

The beginning address of these machine codes in memory is indicated by the first value of the hex view, as shown in Table 2, and each value (byte) comprises significant components of the PE, such as instruction code or data. One of the most well-known recursive traversal disassemblers, the IDA Pro tool automatically analyzes binary files' source code by using cross-references between code segments, knowledge of API call parameters, and other data. IDA Pro interprets the byte sequence as an assembly view, as shown in Table 2.

Table 2. Data overview.

Hex view
00401000 00 00 80 40 40 28 00 1C 02 42 00 C4 00 20 00401010 00 00 20 09 2A 02 00 70 00 03 5E 10 31 0A 00401020 40 00 02 01 00 90 21 00 32 40 00 1C 01 40
Assembly view
.text:00401000 56 push esi .text:00401005 50 push eax .text:00401005 50 assume es: nothing

3.3. Feature Extraction

The use of techniques such as obfuscation, confusion, and polymorphism further complicate the malicious patterns of malware and its variants. Therefore, it is difficult to effectively distinguish malware families with single feature information. To make use of the complementary information that these two representations bring, we extracted features from binary and assembly language files for accurate and quick categorization. The experimental results demonstrate that combining these two types of features enhances the classification system's accuracy. The experimental details are introduced in Section 4. This section will detail the features used in this paper.

3.3.1. Binary File

Byte_1_gram: The *n*-gram is a contiguous representation of *n* items of a given sequence, widely used in language modeling, DNA sequence detection, etc. In the field of computer security, malware samples can be efficiently represented as hexadecimal sequences using n-gram analysis to learn important details about different virus types. Based on this, this paper constructed 257-dimensional 1_gram frequency features from binary files.

Byte_2_gram: As a text analysis tool, byte_2_gram is widely used in text mining and analysis in NLP. Numerous studies have shown that byte_2_gram features play an important role in the field of malware classification with their low system overhead and high classification accuracy. Therefore, this paper built a 2562-dimensional 2_gram frequency featured after 1_gram and selected the best 360-dimensional byte_2_gram combination from each binary file by cross-validation.

3.3.2. Assembly File

Asm_Pixel: Assembly files contain much software-related texture information. Therefore, the image representation transformation of the assembly file can effectively utilize its texture information to classify malware [26]. This paper converted all pixelated images to a NumPy array and used cross-validation to find a feature set consisting of 2300 significant pixel features from each malware image.

Asm_1_gram: Numerous studies have found that some features of assembly files, such as segments, opcodes, keywords, and registers, are closely related to malicious activities of malware, which can effectively distinguish malware families. The Microsoft Challenge provided 10,868 assembly language files totaling 150 GB. Due to the huge source data is not conducive to feature extraction, this paper adopts parallel processing technology to randomly divide all assembly files into five folders, and each folder simultaneously extracts these four types of features. Details are as follows:

(a) Segments: The preset sections in PE (.text, .data, .bss, .rdata, .edata, .idata, .rsrc, etc.). We selected 13 commonly used segments, and the experiments prove that these segments benefit the classification results;
(b) Opcodes: The machine code, which represents the assembly instruction, is represented by the opcode as a helper. The full list of $\times 86$ instruction sets is vast and complex, so we chose a subset of 26 opcodes based on how frequently they are used in malware samples. We measured how frequently these 26 opcodes were used in each malware sample;
(c) Keywords: Extracted three keyword features: .dll, std::, and :dword;
(d) Registers: Extracted the features of registers edx, esi, eax, ebx, ecx, edi, ebp, esp, and eip and experimentally demonstrated that the frequency of register usage is a helpful feature for assigning malware samples to a family.

Opcodes_2_gram: The opcode is a command that defines the behavior of malware that does not rely on any external resources or domain knowledge. Shankarapani et al. [36] found that opcodes have higher accuracy by comparing malware classification performed by assembly and application programming interface (API) call sequences. Hence, we constructed 565-dimensional Opcodes_2_gram frequency features from assembly files.

Opcodes_3_gram: Malware has more sophisticated malicious patterns due to the use of techniques such as code reuse and automation. More machine code accompanies common operations of most malware. Therefore, if the value of n in the n-gram is small, more complex malicious operation codes cannot be effectively discovered. Therefore, this paper extracted 2000-dimensional Opcodes_3_gram frequency features after Opcodes_2_gram.

3.4. Feature Fusion

The most straightforward way to combine feature categories is to stack them in a single feature vector and verify their effect by the same classifier. However, irrelevant and redundant features in the stacking process not only increased the need for extra processing complexity but also had the potential to decrease the model's accuracy. Therefore, this paper mainly considers the following feature fusion methods. One approach is the best subset selection technique. This technique starts with a subset comprising only one feature. Retain the subset with the highest objective function value (such as accuracy, loss function, etc.) by assessing the performance of the trained classifier. For each subgroup with f features, the procedure is repeated, with f growing by one at each step. The forward stepwise selection method is one more method we consider. By gradually adding features to the model one at a time, this technique starts with a feature-less model and gradually expands its feature set. Comparing this feature selection method to the best subset selection method, it is more computationally efficient. because the best subset selection method considers only $\sum_{i=1}^{f}(f-k) = \frac{f(f+1)}{2}$ subsets, while the latter uses a greedy algorithm to consider all 2^f possible models. The third approach we consider is early fusion. Multiple layers of features are fused, and then the predictor is trained on the fused features. The concat operation is used to connect the two features directly. If the dimensions of the two input features x and y are p and q, the dimension of the output feature z is $p + q$. At the same time, the parallel strategy is used to combine the two feature vectors into a complex vector. For the input features x and y, $z = x + iy$, where i is an imaginary number unit. The representative algorithms are Inside-Outside Net and HyperNet. The opposite of this approach is late fusion. To enhance the detection performance, it integrates the findings of multiple layers of detection; that is, the characteristics of different dimensions are tested separately, and then the test results are integrated.

Considering the efficiency and practicability of modeling, this paper adopts the forward stepwise selection algorithm for feature fusion. In each iteration, the feature category that yields the smallest logloss is added to our model. Stop the process until the value of logloss does not reduce during the addition process.

3.5. Classification Model

Since various tree-based boosting algorithms have the advantages of strong interpretability, high speed, and high precision, XGBoost, LightGBM, and CatBoost algorithms have received great attention in the field of classification and detection. The XGBoost algorithm, a parallel implementation of a gradient boosting tree classifier that produced better performance than other algorithms in most cases, was mostly used by the top 10 teams in the recent Kaggle contest. Furthermore, as a fast and efficient parallel algorithm, XGBoost has fully adjustable parameters, which is the main motivation for using this algorithm. In this paper, we also utilized random forest, SVM, KNN, AdaBoost, etc., classifiers to assess the viability of the suggested fusion feature set and malware classification system. In the experimental section, categorization models will be covered in more detail.

3.6. Measures for Evaluation

As shown in Formulas (9)–(13), we used the metrics accuracy, precision, recall, F1-score, and logloss to evaluate the effectiveness of the classification system. Precision denotes the quantity of malware instances accurately categorized as a percentage of all malware incidents in the predicted family. Recall denotes the amount of malware instances correctly classified, divided by the total number of malware instances in the actual malware family. The F1-score takes both false positives and false negatives into account. Therefore, the F1-score is a better measure to seek a balance between precision and recall. Accuracy is an important parameter that determines the percentage of the correctly classified instance. Logarithmic loss measures the performance of a classification model by comparing the gap between the predictions and the real data.

$$\text{Precision} = \frac{T_p}{T_p + F_p} \tag{9}$$

$$\text{Recall} = \frac{T_p}{T_{p+F_n}} \tag{10}$$

$$F_1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

$$\text{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{12}$$

where T_p is a true positive, T_n is a true negative, F_p is a false positive, and F_n is a false negative.

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log\left(p_{ij}\right) \tag{13}$$

where N is the number of observations, M is the number of category labels, log is the natural logarithm, y_{ij} is 1 if observation i is in category j and 0 otherwise, and p_{ij} is the predicted probability that observation i is in category j.

4. Experimental Results and Discussion

4.1. Data

Microsoft provided a set of known malware files in a competition held in 2015 (BIG2015) with nearly 0.5 terabytes of data associated with a total of 21,741 malware samples, 10,868 of which were used for training and the remaining ones for testing. There are nine separate families represented by it: Ramnit (R), Lollipop (L), Keli-hos ver3 (K3), Vundo (V), Simda (S), Tracur (T), and Kelihos ver1 (K1), which stands for obfuscator. Table 3 displays ACY (O) and Gatak (G). Each file has a class label that ranges from 1 to 9, with 1 denoting the first malware family in the list above and 9 denoting the final. Two files make up each malware sample, one of which contains hexadecimal code and the other of which contains disassembly code. To make sure the file was clean, Microsoft erased the PE header. Each malware file has an id: a 20-character hash to identify the file uniquely, and a class, an integer representing one of the nine family names to which the malware may belong. Since the test set does not provide the corresponding label, we used the training set to evaluate the efficiency of our proposed system.

Table 3. Malware families in the training dataset.

Class ID	Family Name	Train Samples	Type
1	Ramnit	1541	Worm
2	Lollipop	2478	Adware
3	Kelihos_ver3	2942	Backdoor
4	Vundo	475	Trojan
5	Simda	42	Backdoor
6	Tracur	751	TrojanDownloader
7	Kelihos_ver1	398	Backdoor
8	Obfuscator.ACY	1228	Any kind of obfuscated malware
9	Gatak	1013	Backdoor

4.2. Feature Selection

The malware feature set is typically high-dimensional and sparse, which is not conducive to malware classification. However, most classifiers perform better on low-dimensional dense features. Therefore, this paper mainly uses the chi-square verification algorithm to shrink the size of single-category features and selects the optimal number of features by evaluating the performance of the XGBoost algorithm under cross-validation.

For the Byte_2_gram feature: In all combinations of 2-grams, the chi-square validation at intervals of 50 and the XGBoost classification algorithm are selected to confirm the single-category features that make it easiest to identify the malware family. The outcomes of the trial indicate that the 350 features selected by chi-square verification eliminate extraneous and pointless elements in the feature set and can tend to the optimal solution in terms of classification performance. Therefore, to clarify the quantity of characteristics in the feature range of 300~400, this paper further confirms that the first 360 features can obtain the highest performance of the classifier through chi-square verification at intervals of 10, which is the optimal solution for the single-category feature set of Byte_2_gram.

For the Asm_Pixel feature: Each assembly file image was converted to a NumPy array. The dimension of selected features is reduced based on chi-square validation with an interval of 100, and the XGBoost algorithm is employed to measure the effectiveness of single-category features. The experimental results show that the 2300 features after dimensionality reduction perform better on the classifier. To further determine the number of features, we selected the feature range of 2200~2400 and verified performance based on chi-square validation at intervals of 20. Finally, the first 2300 features were selected to form the optimal feature set of Asm_Pixel.

For the Opcodes_2_gram feature: Like the selection method of the first two categories of features, we conducted chi-square validation in the feature range of all Opcodes 2-gram at intervals of 25 and validated the classification effect using the XGBoost classifier. The experimental findings demonstrate that the classifier's classification effect for 625-dimensional features tends to be the optimal solution. Therefore, to further obtain the number of features with the highest classification accuracy, this paper adopted a feature range of 550 to 625 and continues testing at intervals of 5. Experiments confirm that the first 565 features are the optimal solutions for Opcodes_2_gram single-category features.

For the feature Opcodes_3_gram: Although the vector space induced by the program for Opcodes_2_gram only has 565 different features. A modest value of n will, however, fail to detect complicated malicious blocks of operations, as the majority of typical operations that might be utilized for malicious purposes require more than one or two machine operation codes. Consequently, we also extracted and utilized the Opcodes 3-gram sequences as features. Due to larger values of *n* in the *n*-gram leading to a sparser feature matrix, we performed chi-square validation with intervals of 100 on over 60,000 Opcodes 3-gram. The experimental results show that the 2000 features after dimensionality reduction through chi-square verification remove irrelevant and redundant features in the feature set and achieve high classification accuracy on the XGBoost classifier. To further clarify the number of features, in the feature range of 1900~2100, we further confirmed that the first 2000 features are the optimal solution for the single-category feature of Opcodes_3_gram using chi-square verification with an interval of 20.

As shown in Table 4, chi-square validation streamlines the number of features while reducing feature extraction time. Although the time difference is not large, the number of features that are most conducive to classification is screened through chi-square verification, which reduces system complexity and memory consumption and is conducive to improving model training and classification efficiency.

Table 4. Chi-square verification for feature selection.

Feature	Number	Time (Seconds)	Number (Chi-Square)	Time (Seconds)
Byte_2_gram	2000	1013.15	360	1011.78
Asm_Pixel	5000	966.77	2300	944.75
Opcodes_2_gram	625	0.67	565	0.56
Opcodes_3_gram	10,000	69.21	2000	68.89

Overall, this paper used chi-square validation and the XGBoost classification algorithm to select single-category features that are most beneficial to distinguish malware families. The chi-square verification is used to eliminate redundant and unnecessary characteristics from the dataset, thereby reducing system overhead and improving classification efficiency. The XGBoost classification algorithm is adopted to measure the effectiveness of the selected features. Meanwhile, we also selected the t-SNE (t-Distributed Stochastic Neighbor Embedding) visualization tool to appear the correlation degree of the selected features. As shown in Figure 4, 9 types of malware families are clearly divided, which proves that the correlation among the selected features is small; hence, it is beneficial to distinguish malware families after feature fusion. The details of feature fusion will be introduced in the next section.

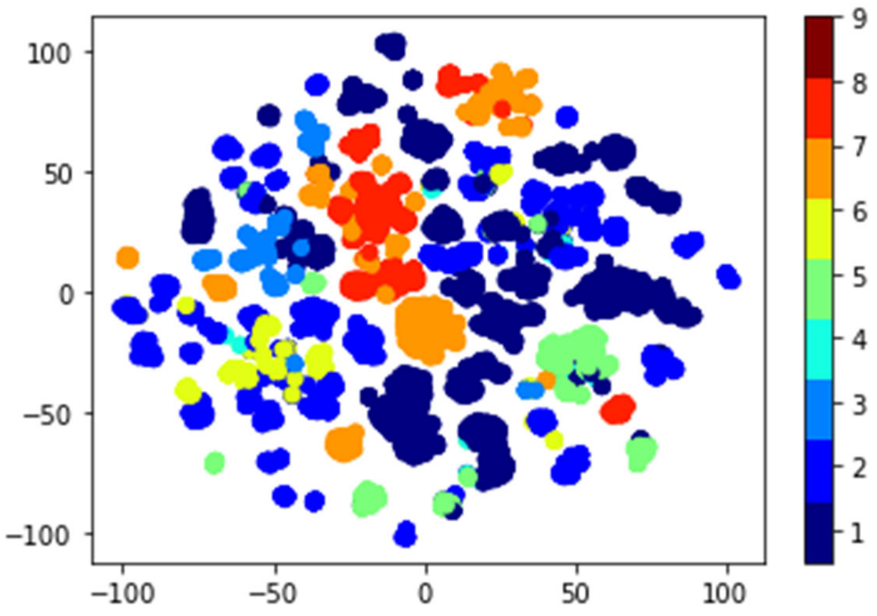

Figure 4. Dataset t-SNE 2D visualization.

4.3. Experimental Results Analysis

In the above, we obtained six sets of categorical features through feature extraction and selection. To verify the effectiveness of single-category features, this paper uses the tree-boosting-based XGBoost classification algorithm to train and test them, as shown in Table 4. Since the Microsoft Challenge does not provide labels for the test dataset, we used five-fold cross-validation on the training set for testing.

As shown in Table 5, the classification accuracy of each proposed six-category feature is over 99.7% and 98.7% on both the test set and the training set. It demonstrates that the selected features can effectively distinguish the families to which the malware belongs. In particular, the Asm_1_gram feature alone achieves the greatest classification accuracy of 99.45% and a low-level logarithmic loss of 0.0207. To significantly increase the suggested system's classification accuracy and decrease memory overhead, we adopted the forward feature stepwise selection algorithm for effective feature fusion to make a trade-off between the number of features and the classification efficiency, as shown in Table 5.

Table 5. List of feature categories and their evaluation with XGBoost.

Category of Feature	Numbers	Train		5-CV	
		Accuracy	Logloss	Accuracy	Logloss
Hex dump file					
Byte_1_gram	257	1.0	0.0007	0.9873	0.0477
Byte_2_gram	360	1.0	0.0005	0.9940	0.0234
Disassembled file					
Asm_Pixel	2300	0.9999	0.0010	0.9902	0.0399
Asm_1_gram	48	0.9982	0.0054	0.9945	0.0207
Opcodes_2_gram	565	0.9971	0.0082	0.9914	0.0335
Opcodes_3_gram	2000	0.9971	0.0083	0.9921	0.0307

As shown in Table 6, according to the Asm_1_gram category features with the lowest logloss, we used a forward feature stepwise selection algorithm to add other category features that make the model produce the lowest logloss to form a fusion feature set. The experimental findings demonstrate that the classification accuracy of all fusion feature sets (C1~C5) based on the XGBoost classification algorithm is 100% on the training data. The logloss values of C1~C5 under the test data continue to decrease, especially C5 achieved a classification accuracy of 99.79% and a log loss of 0.009. It demonstrates that the feature set fused by the forward selection algorithm can gradually improve classification accuracy. In particular, the C5 fusion feature set can help to build a reliable malware classification scheme.

Table 6. Adding features categories gradually based on feature fusion.

Category of Feature	Numbers	Train		5-CV	
		Accuracy	Logloss	Accuracy	Logloss
C1: Asm_1_gram + Asm_Pixel	2348	1.0	0.00042	0.9959	0.0163
C2: C1 + Byte_1_gram	2605	1.0	0.00036	0.9966	0.0129
C3: C2 + Opcodes_2_gram	3170	1.0	0.00035	0.9976	0.0097
C4: C3 + Opcodes_3_gram	5170	1.0	0.00035	0.9978	0.0092
C5: C4 + Byte_2_gram	**5530**	**1.0**	**0.00035**	**0.9979**	**0.0090**

The previous experiments show that the fusion feature set (C5) can achieve 99.79% classification accuracy on the original XGBoost classifier. The XGBoost algorithm based on tree structure has the advantage of adjustable parameters. We used a grid search algorithm to adjust the parameters of the XGBoost to obtain the optimal parameter combination, which is most conducive to distinguishing malware families. The optimization results show in Table 7. After parameter adjustment, the XGBoost classification algorithm has a 99.87% classification accuracy on the fusion feature set, an increase of 0.08%, and a logarithmic loss of 0.007, a decrease of 0.002. It further illustrates that the tree-boosting-based malware classification system can effectively distinguish the malware families.

Table 7. XGBoost parameter optimization is used.

Category of Feature	Numbers	5-CV	
		Accuracy	Logloss
C5 (Parameter optimization)	**5530**	**0.9987**	**0.007**
C5 (No parameter optimization)	5530	0.9979	0.009

The categorization model's log-normalized confusion matrix is illustrated in this publication to assess the viability of the suggested system. As can be seen from Figure 5, the true positive rate of the system for each type of malware family is above 97%, especially for the R and K3 families, which can achieve 100%. Among them, the G family has a true positive rate of 99.21%, which may have a similar malicious pattern to the O family. Overall, the malware classification system proposed in this paper can effectively classify malware and its variants.

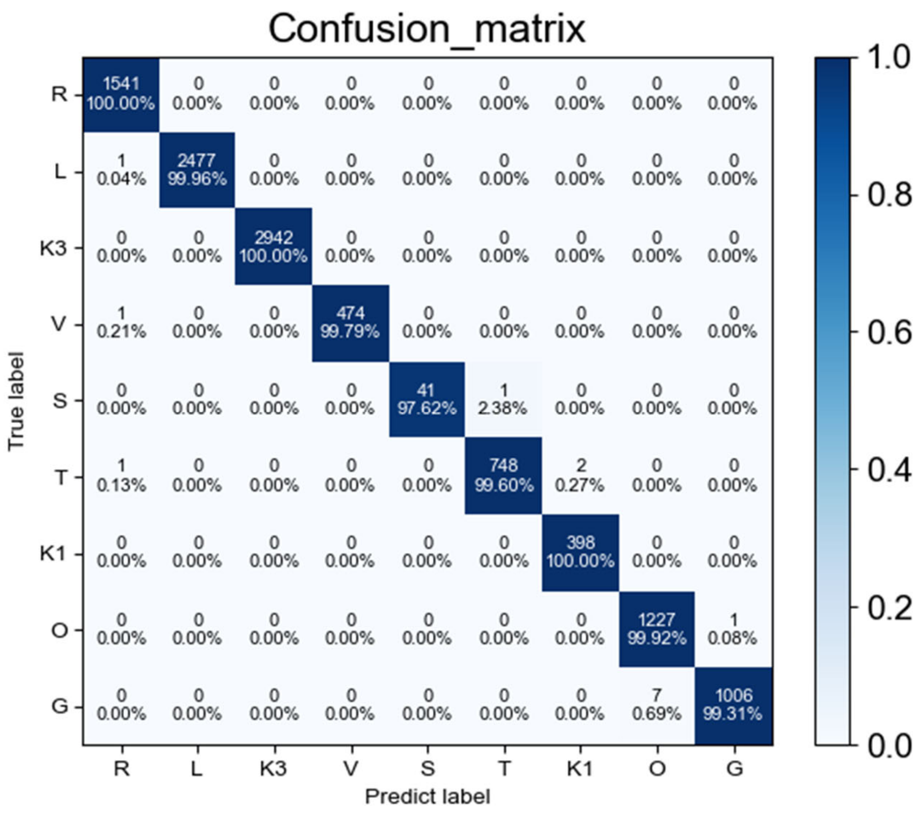

Figure 5. Confusion matrix of fused features based on XGBoost classifier.

4.4. Comparison and Discussion

In this section, to evaluate the efficacy of the combined feature set, we applied five common machine-learning techniques. The accuracy, precision, recall, and F1-score of the system suggested in this research are compared to those of the prior work.

Figure 6 shows that compared with the feature set used in the existing work, the fusion feature set proposed in this paper performs better in classification methods for machine learning, such as KNN, SVM, random forest, AdaBoost, and XGBoost. The following is a detailed analysis and comparison.

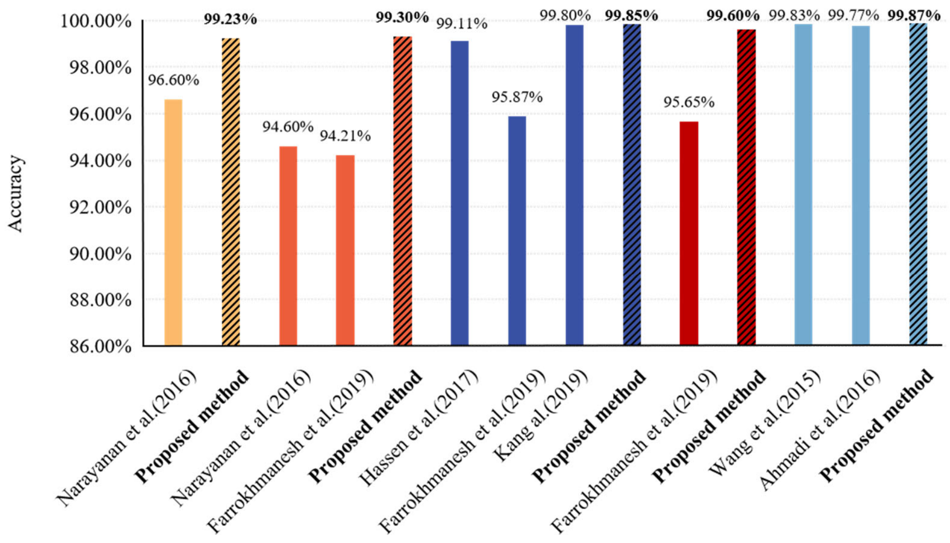

Figure 6. Our model is compared to recent research on the BIG2015 dataset [19,37–41].

Narayanan et al. [37] utilized PCA to reduce the dimension of gray image representation converted from malware and used KNN to classify malicious software. For the BIG2015 dataset, the experimental findings demonstrate that the system has a classification accuracy of 96.6%. In contrast, our fusion feature set not only includes image information (Asm_Pixel), but also inconsistent opcodes, assembly codes, and other information that are more beneficial to classify malware. It proves that the fusion feature set containing multiple category features is more conducive to classifying malware families. Therefore, under the same KNN classification algorithm, this paper has achieved a classification accuracy of 99.23%.

Narayanan et al. [37] also used an SVM classifier to classify the BIG2015 dataset with an accuracy of 94.6%. Farrokhmanesh et al. [38] proposed a new method to detect malware using digital signal processing strategies. To create a machine-learning music classification model from audio signals, Farrokhmanesh et al. [38] converted program bytes into meaningful audio signals and used music information retrieval (MIR) methods. The model's classification accuracy on the BIG2015 dataset using the SVM classifier was 94.21%. However, the fused feature set proposed in this paper achieves 99.3% classification accuracy on the SVM classifier, which is higher than the above two methods.

Hassen et al. [39] extracted a new control statement overlap-based feature to classify the BIG2015 dataset using a random forest classifier with an accuracy of 99.11%. Kang et al. [40] employed malware parameters, such as resource size, virtual size, number of sections, and image version, to classify the BIG2015 dataset using the random forest classifier with an accuracy of 99.8%. Farrokhmanesh et al. [38] achieved 95.87% and 95.65% classification accuracy, respectively, on the BIG2015 dataset using random forest and AdaBoost. By contrast, the random forest and AdaBoost classifier separately achieved 99.85% and 99.6% accuracy on the proposed fusion feature set. Wang et al. [41] and Ahmadi et al. [19] applied the XGBoost classifier, which utilized features extracted from malware binaries and assembly files, to classify malware into their families with 99.83% and 99.77% accuracy. The proposed system used the XGBoost classification algorithm to achieve a higher accuracy of 99.87% on the fusion feature set.

In summary, using our fusion feature set can vastly improve the classification accuracy of the classifiers. Compared to the current malware classification models, the amount, kind, and complexity of features and classification approaches are used. It is less complex in terms of the feature number, category, and classification algorithm. Therefore, our fusion feature set can be used to design a high-speed, high-accuracy, and efficient malware classification system.

Table 8 compares the proposed system's accuracy, precision, recall, and F1 score to that of the previous work. A malware classification technique based on opcode sequences was proposed by Sun et al. [32]. The system converted malicious programs into grayscale images, used RNN for feature extraction, employed CNN for image classification, and achieved 99.5% accuracy on the BIG2015 dataset. Due to the high complexity of malware itself, we considered not only opcode sequences but also bytecodes and features of image texture. The suggested method constructed based on the XGBoost classification algorithm achieves 99.87% accuracy on the BIG2015 dataset. Therefore, it is more suitable for complex malware classification tasks.

Table 8. Comparison with other methods.

Approaches	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RMVC [32]	99.50	-	-	-
MCFT and CNN [42]	98.63	98.56	96.00	97.22
RCNF [43]	99.56	-	-	98.20
SERLA [44]	98.31	98.68	97.93	98.30
MDMC [45]	99.26	-	-	-
ML and VT [46]	97.73	-	-	-
HYDRA [47]	99.75	-	-	99.51
XGBoost and RNN [48]	96.90	-	-	-
CNN and BILSTM [49]	98.20	-	-	96.05
ACO-DT [50]	99.37			
CatBoost	**99.76**	**99.66**	**98.93**	**99.28**
LightGBM	**99.84**	**99.77**	**98.55**	**99.66**
XGBoost	**99.87**	**99.87**	**99.58**	**99.70**

A deep learning-based malicious software classification system (MCFT-CNN) was proposed by Sudhakra et al. [42] that does not require manual feature analysis or the actual execution of malware. The system achieved 98.63% accuracy, 98.56% precision, 96% recall, and 97.22% F1-score on the BIG2015 dataset. We used a tree-boosting-based machine-learning classification system, improved the classification accuracy by 1.24%, and the model is interpretable, which is helpful for further research and analysis of malicious. A novel methodology was suggested by Gibert et al. [47] to efficiently identify and categorize malware by combining different types of features, including API sequences, bytecodes, and file structures. On the BIG2015 dataset, CNN is 99.75% accurate, and the F1-score is 99.51%. Our proposed fusion feature set achieves greater F1-score and accuracy on the lightweight classifier-XGBoost, separately increased by 0.12% and 0.19%.

Çayır et al. [43] proposed a lightweight network model based on static analysis. It requires less feature analysis and lower system complexity, with an F1-score of 98.2% on the BIG2015 dataset and an accuracy of 99.56%. Jian et al. [44] proposed a deep neural network-based visualization approach for classifying malware with 98.31% accuracy, 98.68% precision, 97.93% recall, and 98.3% F1-score on the BIG2015 dataset. A malware classification system based on data mining and visualization of PE files was proposed by Yuan et al. [45]. The system converts the malicious file binary into a gray image as the deep convolution neural network's input. We not only considered the image texture features of malware but also fused features from other categories, which is more beneficial for malicious family classification. We adopted the more interpretive XGBoost classification algorithm, and the classification accuracy reached 99.87%.

A new cloud-based, three-component malware classification model with semi-supervised migratory learning was proposed by Gao et al. [48]: training, prediction, and migration. The experimental outcomes on the BIG2015 dataset demonstrate that semi-supervised migration learning has a 96.9% accuracy rate. They used a hybrid classification model composed of XGBoost and RNN, while we only used a separate XGBoost classifier, which greatly reduces the model complexity and memory consumption while ensuring high classification accuracy. Kattamuri et al. [50] proposed a new data set (SOMLAP) based on PE file header information for malware research and analysis. At the same time, based on three feature selection algorithms (ACO, CSO, GWO), the features of the data set are streamlined to improve the accuracy and time efficiency of the detection system. Experiments have proved that the SOMLAP dataset makes up for the lack of benchmark datasets in malware feature representation, contains more new features that are conducive to detection, and effectively improves detection accuracy. Moreover, the combination of ACO and decision tree screened out the 12 feature categories that are most conducive to classification, achieving a classification accuracy of 99.37%. Inspired by this, we can then perform feature dimensionality reduction on the fusion feature set to reduce time consumption while ensuring accuracy.

In conclusion, the fusion feature set suggested in this paper produced highly accurate classification results and F1-score on the tree boosting-based algorithms, including XGBoost, LightGBM, and CatBoost. Among them, the XGBoost algorithm achieved the highest accuracy of 99.87%, which is better than most current classification systems.

5. Conclusions

The explosive growth of malware types has made the classification of malware a prominent topic. Existing malware classification schemes are mainly in accordance with a variety of malware static features for classification, which may contain many irrelevant and redundant features, resulting in high feature dimension, computational complexity, time, and memory overhead. Meanwhile, most of the existing classification models are not highly interpretable, which is not conducive to the analysis and research of malware. Therefore, we proposed a tree boosting-based malware classification system with model interpretability, high speed, and high accuracy to efficiently classify malware variants into their actual families based on a fusion feature set consisting of a limited, finite number of malware features.

This paper constructed a new fusion feature set based on the BIG2015 dataset by exploiting the complementary information brought by malware content and structure. We used a forward feature stepwise selection technique for effective feature fusion. Various machine-learning algorithms, including random forest, SVM, KNN, and AdaBoost, are used to confirm the effectiveness of the fusion feature set. The suggested system beats most of the existing research works in terms of accuracy, precision, recall, and F1-score, according to experimental data. We developed an effective malware classification system employing tree-boosting-based machine-learning methods. Furthermore, we applied LightGBM and CatBoost algorithms to the domain of malware classification for the first time, and both achieved excellent results on the proposed fusion feature set. The malware classification system should be suitable for large-scale real-time classification tasks. Future work will continue to emphasize striking a balance between the challenges of time and accuracy.

Author Contributions: Conceptualization, Z.C.; methodology, Z.C.; software, X.R.; validation, X.R.; formal analysis, X.R.; investigation, X.R.; resources, Z.C.; data curation, X.R.; writing—original draft preparation, Z.C.; writing—review and editing, X.R.; visualization, X.R.; supervision, Z.C.; project administration, Z.C.; funding acquisition, Z.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (grant no. 62102190).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data are available from https://www.kaggle.com/competitions/malware-classification/data, accessed on 8 April 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Said, V.; Eelly, E.; Zag, E.; Murat, O. The Dangerous Combo: Fileless Malware and Cryptojacking. *J. SoutheastCon* **2022**, 125–132. [CrossRef]
2. Greengard, S. Cybersecurity gets smart. *Commun. ACM* **2016**, *59*, 29–31. [CrossRef]
3. Rizvi, S.K.J.; Aslam, W.; Shahzad, M.; Saleem, S.; Fraz, M. PROUD-MAL: Static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. *Complex Intell. Syst.* **2022**, *8*, 673–685. [CrossRef]
4. Johnson, S.; Gowtham, R.; Nair, A.R. Ensemble Model Ransomware Classification: A Static Analysis-based Approach. *Inventive Comput. Inf. Technol.* **2022**, *336*, 153–167.
5. Loi, N.; Borile, C.; Ucci, D. Towards an Automated Pipeline for Detecting and Classifying Malware through Machine Learning. 2021. Available online: https://arxiv.org/abs/2106.05625 (accessed on 5 December 2022).
6. Jeon, J.; Kim, J.; Jeon, S.; Lee, S.; Jeong, Y.S. Static Analysis for Malware Detection with Tensorflow and GPU. In *Advances in Computer Science and Ubiquitous Computing*; Springer: Singapore, 2021; Volume 715, pp. 537–546.
7. Barbi, S.; Barbieri, F.; Marinelli, S.; Rimini, B.; Merchiori, S.; Larwa, B.; Bottarelli, M.; Montorsi, M. Phase change materialsand mixtures for distributed latent heat thermal energy storage: Interaction and performance analysis. *Renew. Energy* **2021**, *169*, 1066–1076. [CrossRef]
8. Chanajitt, R.; Pfahringer, B.; Gomes, H.M.; Yogarajan, V. Multiclass Malware Classification Using Either Static Opcodes or Dynamic API Calls. In Proceedings of the AI 2022: Advances in Artificial Intelligence, Perth, WA, Australia, 3 December 2022; pp. 427–441.
9. Jing, C.; Wu, Y.; Cui, C. Ensemble dynamic behavior detection method for adversarial malware. *Future Gener. Comput. Syst.* **2022**, *30*, 193–206. [CrossRef]
10. Li, C.; Lv, Q.; Li, N.; Wang, Y.; Sun, D. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. *Comput. Secur.* **2022**, *116*, 102686. [CrossRef]
11. Anderson, B.; Quist, D.; Neil, J.; Storlie, C.; Lane, T. Graph-based malware detection using dynamic analysis. *J. Comput. Virol.* **2020**, *52*, 247–258. [CrossRef]
12. Alshamrani, S.S. Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files. *Secur. Commun. Netw.* **2022**, *2022*, 7611741. [CrossRef]
13. Wang, W.; Ren, C.; Song, H.; Zhang, S.; Liu, P. FGL_Droid: An Efficient Android Malware Detection Method Based on Hybrid Analysis. *Secur. Commun. Netw.* **2022**, *2022*, 8398591. [CrossRef]
14. Catak, F.O.; Yazı, A.F. A Benchmark API Call Dataset for Windows PE Malware Classification. 2019. Available online: https://arxiv.org/abs/1905.01999 (accessed on 11 December 2022).
15. Afianian, A.; Niksefat, S.; Sadeghiyan, B.; Baptiste, D. Malware dynamic analysis evasion techniques: A survey. *ACM Comput. Surv. (CSUR)* **2020**, *52*, 1–28. [CrossRef]
16. Lebbie, M.; Prabhu, S.; Agrawal, A.K. Comparative Analysis of Dynamic Malware Analysis Tools. In Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences, Kurukshetra, India, 1 January 2022; pp. 359–368.
17. Du, J.; Raza, S.H.; Ahmad, M.; Alam, I.; Hanif, S.; Habib, M.A. Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection. *Secur. Commun. Netw.* **2022**, *2022*, 1424638. [CrossRef]
18. Gu, Z.; Nazir, S.; Hong, C.; Khan, S. Convolution Neural Network-Based Higher Accurate Intrusion Identification System for the Network Security and Communication. *Secur. Commun. Netw.* **2020**, *2020*, 8830903. [CrossRef]
19. Ahmadi, M.; Ulynaov, D.; Semenov, S.; Trofimov, M.; Giacinto, G. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; pp. 183–194.
20. Anderson, H.S.; Roth, P. Ember: An Open Dataset for Training Static pe Malware Machine Learning Models. 2018. Available online: https://arxiv.org/abs/1804.04637 (accessed on 12 December 2022).
21. Zhang, Y.; Haghani, A. A gradient boosting method to improve travel time prediction. *Transp. Res. Part C Emerg. Technol.* **2015**, *58*, 308–324. [CrossRef]
22. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 3149–3157.
23. Dorogu, A.V.; Ershov, V.; Gulin, A. CatBoost: Gradient Boosting with Categorical Features Support. 2018. Available online: https://arxiv.org/abs/1810.11363 (accessed on 16 December 2022).
24. Mao, K.Z. Orthogonal forward selection and backward elimination algorithms for feature subset selection. *IEEE Trans. Syst. Man Cybern.* **2004**, *34*, 629–634. [CrossRef] [PubMed]
25. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
26. Tekerek, A.; Yapici, M. A novel malware classification and augmentation model based on convolutional neural network. *Comput. Secur.* **2022**, *112*, 102515. [CrossRef]
27. Dai, Y.; Li, H.; Qian, Y.; Lu, X. A malware classification method based on memory dump grayscale image. *Digit. Investig.* **2018**, *27*, 30–37. [CrossRef]
28. Gibert, D.; Mateu, C.; Planes, J.; Vicens, R. Using convolutional neural networks for classification of malware represented as images. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 15–28. [CrossRef]

29. Sun, Z.; Rao, Z.; Chen, J.; Xu, R.; He, D.; Yang, H.; Liu, J. An Opcode Sequences Analysis Method For Unknown Malware Detection. In Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis, Prague, Czech Republic, 15–17 March 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 15–19.
30. Ijaz, M.; Durad, M.H.; Ismail, M. Static and Dynamic Malware Analysis Using Machine Learning. In Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 8–12 January 2019; pp. 687–691.
31. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Damasevicius, R. An Efficient DenseNet-Based Deep Learning Model for Malware Detection. *Entropy* **2021**, *23*, 3. [CrossRef] [PubMed]
32. Sun, G.; Qian, Q. Deep Learning and Visualization for Identifying Malware Families. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 283–295. [CrossRef]
33. Li, L.; Ding, Y.; Li, B.; Qiao, M.; Ye, B. Malware classification based on double byte feature encoding. *Alex. Eng. J.* **2022**, *61*, 91–99. [CrossRef]
34. Kumar, S.; Janet, B.; Neelakantan, S. Identification of malware families using stacking of textural features and machine learning. *Expert Syst. Appl.* **2022**, *208*, 118073. [CrossRef]
35. Jadvani, N.; Agarwal, M.; Leelasankar, K. Malware Detection Based on Portable Executable File Features. In Proceedings of the International Conference on Computing, Communication, Electrical and Biomedical Systems, Cham, Switzerland, 28 February 2022; pp. 377–384.
36. Shankarapani, M.K.; Ramamoorthy, S.; Movva, R.S.; Mukkamala, S. Malware detection using assembly and API call sequences. *J. Comput. Virol.* **2011**, *7*, 107–119. [CrossRef]
37. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. Performance analysis of machine learning and pattern recognition algorithms for Malware classification. In Proceedings of the 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, USA, 25–29 July 2016; pp. 338–342.
38. Farrokhmanesh, M.; Hamzeh, A. Music classification as a new approach for malware detection. *J. Comput. Virol. Hacking Tech.* **2019**, *5*, 77–96. [CrossRef]
39. Hassen, M.; Carvalho, M.M.; Chan, P.K. Malware classification using static analysis-based features. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November 2017; pp. 1–7.
40. Kang, J.; Won, Y. Malware Classification Using Machine Learning. *Adv. Comput. Sci. Ubiquitous Comput.* **2019**, *536*, 279–284.
41. Wang, X.; Liu, J.; Chen, X. First Place Team: Say No to Overfitting. 2015. Available online: https://github.com/xiaozhouwang/kaggle_Microsoft_Malware/blob/master/Saynotooverfitting.pdf (accessed on 21 December 2022).
42. Sudhakra; Kumar, S. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things. *Future Gener. Comput. Syst.* **2021**, *125*, 334–351. [CrossRef]
43. Çayır, A.; Ünal, U.; Dağ, H. Random CapsNet forest model for imbalanced malware type classification task. *Comput. Secur.* **2021**, *102*, 102133. [CrossRef]
44. Jian, Y.; Kuang, H.; Ren, C.; Ma, Z.; Wang, H. A novel framework for image-based malware detection with a deep neural network. *Comput. Secur.* **2021**, *109*, 102400. [CrossRef]
45. Yuan, B.; Wang, J.; Liu, D.; Guo, W.; Wu, P.; Bao, X. Byte-level malware classification based on markov images and deep earning. *Comput. Secur.* **2020**, *92*, 101740. [CrossRef]
46. Liu, X.; Lin, Y.; Li, H.; Zhang, J. A novel method for malware detection on ML-based visualization technique. *Comput. Secur.* **2020**, *89*, 101682. [CrossRef]
47. Gibert, D.; Mateu, C.; Planes, J. HYDRA: A multimodal deep learning framework for malware classification. *Comput. Secur.* **2020**, *95*, 101873. [CrossRef]
48. Gao, X.W.; Hu, C.; Shan, C.; Liu, B.; Niu, Z.; Xie, H. Malware classification for the cloud via semi-supervised transfer learning. *J. Inf. Secur. Appl.* **2020**, *55*, 102661. [CrossRef]
49. Le, Q.; Boydell, O.; Namee, B.M.; Scanlon, M. Deep learning at the shallow end: Malware classification for non-domain experts. *Digit. Investig.* **2018**, *26*, S118–S126. [CrossRef]
50. Kattamuri, S.J.; Penmatsa, R.K.V.; Chakravarty, S.; Madabathula, V.S.P. Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence. *Electron* **2023**, *12*, 342. [CrossRef]