Imbalanced Malware Family Classification Using Multimodal Fusion and Weight Self-Learning

Shudong Li, Yuan Li, Xiaobo Wu, Sattam Al Otaibi, and Zhihong Tian, *Senior Member, IEEE*

***Abstract*— In recent years, the increasing prevalence of Intelligent Transportation Systems with advanced technologies has led to the emergence of many targeted forms of malware such as ransomware, Trojans, viruses, and malicious mining programs. And malware authors use policies like category disguise or family obfuscation in malware components to evade detection, which poses a great security threat to enterprises, government agencies, and Internet users. In this paper, we propose a malware family classification approach based on multimodal fusion and weight self-learning. Firstly, multiple modalities of malware such as byte, format, statistic, and semantic are fused in various ways to generate effective features. And then, we creatively add a weight self-learning mechanism of malware families into the classification model, which works by continuously calculating log-loss based on the family label and the probabilities predicted by each feature. The approach proves to achieve excellent classification performance on highly imbalanced malware family datasets with high efficiency and small resource overhead, which helps to identify and classify malware families and enhance the efficiency of massive malware analysis in Intelligent Transportation Systems.**

***Index Terms*— Malware family classification, multimodal fusion, weight self-learning, imbalanced data.**

I. Introduction

IN RECENT years, the development of information technology such as Artificial Intelligence and the Internet of Things has led to the surging demand and popularity of Intelligent Transportation Systems [1]. While these technologies are gradually being widely used and considered to revolutionize Intelligent Transportation Systems and autonomous vehicles [2], they also make such devices more vulnerable to hackers and lead to a tremendous rise in the amount of malware, thus threatening personal privacy and security. According to AV-TEST [3], in the past five years, the total number of registered malware increased from 719 million in 2017 to 1313 million in 2021, among which, Windows malware can account for up to 75.74%, indicating that Portable Executable (PE) files are still the most common manifestation of malware. At the same time, various malware families are skyrocketing, with ransomware, Trojans, viruses, malicious mining programs [4], and other forms of malware emerging continuously. The number of system intrusions, cryptojacking, and other cyber-attacks launched based on different types of malware keeps increasing year by year [5]. Since attackers continue to write new types of malware and use them to launch large-scale cyberattacks aiming at Intelligent Transportation Systems [6], most enterprises, government agencies, and Internet users are still under security threats from unknown malware.

In this context, PE malware detection has become the most common issue of cybersecurity in Intelligent Transportation Systems, and the task of PE malware family classification has also received a lot of attention from security researchers. On the one hand, malware from different families may use different resources, files, ports, or other components in a computer system, produce different malicious behaviors, and even affect the system to different degrees, so systems usually need to develop specific protection strategies according to the family to which the malware belongs [7]; on the other hand, classifying malware families can help researchers learn the patterns of malware evolution and predict the future evolution trend of malware families, which contributes to better detect malware and its variants [8]. A large number of approaches have been proposed to solve the issue of PE malware classification, but they currently encounter two key challenges. (1) The detection evasion techniques such as code modification and/or obfuscation strategies employed by malware writers, which pose a great challenge to the accuracy of detection and identification. Malware writers have introduced polymorphism in malware components in order to evade detection by traditional security policies. Although malicious files belonging to the same malware family have the same form of malicious behavior, files that originally belong to the same family look different due to the constant modification and/or obfuscation by writers using various strategies. Nowadays, a large amount of malware tends to protect its data, variables,

Manuscript received 18 February 2022; revised 6 August 2022; accepted 13 September 2022. Date of publication 10 October 2022; date of current version 7 July 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62072131 and Grant U20B2046, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515011401, in part by the Key Research and Development Program of Guangdong Province under Grant 2019B010136003, in part by the Science and Technology Projects in Guangzhou under Grant 202102010442, in part by the National Key Research and Development Program of China under Grant 2019QY1406, in part by the Guangdong Higher Education Innovation Group under Grant 2020KCXTD007, in part by the Guangzhou Higher Education Innovation Group under Grant 202032854, and in part by the Taif University Researchers Supporting Project under Grant TURSP-2020/267. The Associate Editor for this article was S. Mumtaz. *(Shudong Li and Yuan Li are co-first authors.) (Corresponding author: Shudong Li.)*

Shudong Li, Yuan Li, and Zhihong Tian are with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: lishudong@gzhu.edu.cn; ly.liyuan@e.gzhu.edu.cn; tianzhihong@gzhu.edu.cn).

Xiaobo Wu is with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China (e-mail: happywxb@gzhu.edu.cn).

Sattam Al Otaibi is with the Head-of-Innovation-and-Entrepreneurship-Center, College of Engineering, Taif University, Taif 21944, Saudi Arabia (e-mail: srotaibi@tu.edu.sa).

Digital Object Identifier 10.1109/TITS.2022.3208891

and network communication [9] by randomizing variable or function names and using encoding techniques such as XOR to evade malware detection strategies. As for this type of malware and its variants, traditional heuristic and feature-code-based classification approaches are no longer effective and need to be adjusted at any time, which is time-consuming and labor-intensive. At the same time, machine learning, however, gradually plays a significant role in the automated identification and classification of malware with its powerful mass data analysis and computing power. (2) The rapid development and imbalanced distribution phenomenon of malware families in real environments have caused concept drift (i.e., statistical properties of target objects change in an unpredictable style over time) [10], which is an unavoidable problem in the task of analyzing massive malicious samples, which in turn leads to significant performance degradation of machine learning models. Malware belonging to different families distributes unevenly in the current network environment, which makes it difficult to obtain malware samples from certain families as well. Also, as time advances, variation in the distribution of sample classes can affect the actual classification performance of the classification model as its accuracy-oriented designation usually renders the classes that contain few samples to be ignored. Nevertheless, previous studies rarely mention the negative impact of this issue on malware family classification. By constructing classification models for the specific characteristics of malware families, the advantages of machine learning techniques in this field can be better exploited.

Therefore, this paper proposes a malware family classification approach based on multimodal fusion and weight self-learning, aiming to solve the above-mentioned problems encountered in the malware family classification task by constructing machine learning classification models according to the characteristics of malware and its family development. This paper makes the following contributions:

- We focus on multiple modalities such as the byte, format, statistic, and semantic for PE malware and its disassembled files, and perform feature-based multimodal fusion in the feature engineering phase; we also apply artificial intelligence strategies such as weighted soft voting and multi-model integration in the model building phase to achieve decision-based multimodal fusion, which effectively solves the overfitting problem.
- We incorporate a weight self-learning mechanism for malware families in the classification model, which is used to automatically learn the weights of different features for each family during the training process, and the weights are self-learning and migratable to alleviate the concept drift problem.
- We conducted comparative experiments on highly imbalanced malware family datasets. The experimental results show that the proposed method can achieve excellent classification performance with high efficiency and small resource overhead.

The remainder of the paper is arranged as follows. In Section II, we review the related works of malware detection and classification. Section III is devoted to the approach of malware family classification using multimodal fusion and weight self-learning. Section IV provides the experimental results and analysis to validate the approach. Finally, conclusions and future works are drawn in Section V.

II. Related Works

The existing malware is large and diverse, and in the course of long-term malware analysis, security analysts have tried to consider a group of programs with enough "code overlap" to be the same type of malware, i.e., a malware family [11]. The emergence of this concept obviously broadens the scope of the original malware, because, over time, malware authors often modify or reconstruct existing malware to generate malware variants, which have a certain "code overlap" and then constitute the "malware family" together with existing malware.

Considering that machine learning techniques develop rapidly and obtain excellent results in malware detection [12], domestic and international research has gradually tended to focus on the application of machine learning techniques in malware family classification. Approaches based on machine learning for PE malware family classification can be classified into image-based, binary-based, and disassembly-based types according to the format of the malware files analyzed (i.e., the input format used by the approach) [13]. Image-based approaches perform malware family classification by converting PE malware into image files and then applying image classification techniques. Vasan *et al.* [14] proposed a novel classifier, which converted the raw malware binaries into color images that were used by the fine-tuned CNN architecture to detect variants of malware families. Çayır *et al*. [7] proposed an ensemble capsule network model based on the bootstrap aggregating technique, which was less complex and less sensitive to data. Binary-based approaches directly treat PE malware as binary raw files and obtain the byte sequences or behavioral attributes of the files through static or dynamic analysis, based on which new classification features are constructed. Refs. [15], [16] released labeled benchmark datasets and corresponding open-source codes for extracting features from the binaries for training machine learning models to statically detect malicious Windows portable executable files. Raff *et al.* [17] introduced malware detection from raw byte sequences and built a neural network to tackle this problem. Yang *et al.* [18] performed feature engineering on the dynamic behaviors generated from PE malware and introduced the ensemble model to integrate the recognition results from trained multiple classifiers. Fan *et al.* [19] combined weighted heterograph with GNN into malware detection tasks, which constructed a malicious behavior graph to represent the association of malware and its behavior associated entities and learned the semantic information carried in it. Disassembly-based approaches need to first disassemble the PE malware file to get the assembly file, and then construct the classification features based on the assembly code statements in the assembly file. Ni *et al.* [20] extracted the opcode sequences from malware and encoded them with SimHash to an equal length, which was converted to gray images and trained by CNN to identify malware families. Chen *et al*. [21] extracted the opcodes of the disassembled file of malware based on

the label of basic block and in the next generated SimHash value vectors of basic blocks through these opcodes and a hash algorithm while using CNN to train the classification model finally. Ding *et al.* [22] proposed a self-attention-based malware classification method by analyzing malware ASM files to solve the malware classification on imbalanced datasets.

While image-based approaches are simple and straightforward, two issues impede their application: the loss of information caused by converting malware to image files and the interpretability of migrating image classification models to malware analysis. Binary-based approaches have an obvious advantage over image-based approaches in that by refining or adjusting the input patterns and content (APIs, etc.) of the text model, the model can better learn key information about the malware and thus improve the classification accuracy. However, they also encounter several challenges like dealing with malware with large byte sequences. Disassembly-based approaches try to restore the malware code structure and operation process as realistically as possible to obtain better results, but they require a large amount of prior knowledge and processing time to conduct.

A modality is a way something happens or exists, and every source or form of information can be called a modality. Clearly, the application of machine learning techniques to malware family classification is multimodal. On the basis of our analysis of PE malware, either by transforming it into an image, inputting it directly into a binary file, or disassembling it, the multidimensional features made up are multimodal. Different modalities have different manifestations, and there may be many different information interactions between modalities. If the multimodal information can be reasonably processed, correlated, and integrated, rich feature information can be obtained, and then a more robust prediction result can be also obtained, which is exactly multimodal fusion [23]. In order to make the AI-enabled malware classification solutions more effective, more and more studies have started to introduce the concept of multimodal fusion into the malware family classification tasks currently. Gibert *et al.* [24] introduced a bimodal approach to categorize malware into families based on deep learning by combining the byte sequence representing the malware's binary content and the assembly language instructions extracted from the assembly language source code of malware and performing automatic feature learning and classification with a convolutional neural network. Snow *et al.* [25] proposed an end-to-end deep learning framework for multimodel, which utilized three different deep neural network architectures to jointly learn meaningful features from different attributes of the malware data to improve the performance of the model on the malware classification issue. Li *et al.* [26] presented an approach based on feature fusion and machine learning to automatically detect malicious mining code. However, most of the current studies only explore new data modalities aiming to improve classification accuracy or attempt to conduct simple experiments on imbalanced datasets [27], [28] but do not work to eliminate the negative effects caused by inherent problems such as malware variants and concept drift of malware families.

III. Approach

In this paper, we focus on various modalities such as the byte, format, statistic, and semantic for PE malware and its disassembly files, based on which feature engineering (including feature extraction, feature fusion, feature selection, etc.) for multidimensional features is carried out through static analysis. Meanwhile, a malware family classification model is constructed based on machine learning techniques such as gradient boosting, weighted soft voting, and multi-model integration, and a weight self-learning mechanism is added in the voting decision stage for alleviating the negative impacts of imbalanced data and concept drift and further improving the generalization ability of the model. The general framework of the proposed approach is shown in Fig. 1.

A. Feature Engineering

The feature engineering part consists of three steps: multimodal (the byte, format, statistic, semantic, etc.) based feature extraction, feature-based multimodal fusion, and feature selection.

As for byte features, we do not need to parse the PE file but read the PE file directly as a binary byte sequence, on which static byte features such as byte histogram, byte entropy histogram, and string information are extracted.

1) Byte Histogram: The byte histogram counts the number of occurrences of values from 0 to 255 (256 integers) in the file and is normalized by calculating the ratio of these counts to the total number of bytes in the file, so as to obtain the final byte feature vector.

2) Byte Entropy Histogram: The byte entropy histogram needs initially to be set in the file with a sliding window of 2048 bytes and a step size of 1024 bytes. First, for each window, 2048 value pairs (H, X) of file entropy H and byte values X can be obtained by pairing the entropy value with each byte value (2048 non-unique values) within the window. Its entropy can be calculated as Eq. 1. Then, based on the obtained value pairs, we can draw a joint histogram about the file entropy H and byte values X, which is used to approximately represent their joint probability distribution $p(H, X)$. The procedure of generating the byte entropy histogram is shown in Fig. 2. Finally, the distribution matrix of the histogram is spliced into row vectors by rows and normalized similarly to the byte histogram to obtain the final byte entropy feature vector.

$$H = -\sum_{i=1}^{n} P(X_i)\log(P(X_i)) \tag{1}$$

where n is the number of bins that divide entropy values. For the ith bin, X_i is the number of the byte values X contained in the bin.

3) String Information: String information is expressed as the statistical features of printable strings in the file, and here we define printable strings as the strings consisting of at least five consecutive characters in the range of 0x20 to 0x7f. For such printable strings, on the one hand, we pay attention to their number, average length, and the total number, entropy value, and histogram distribution (in the range of 0x20

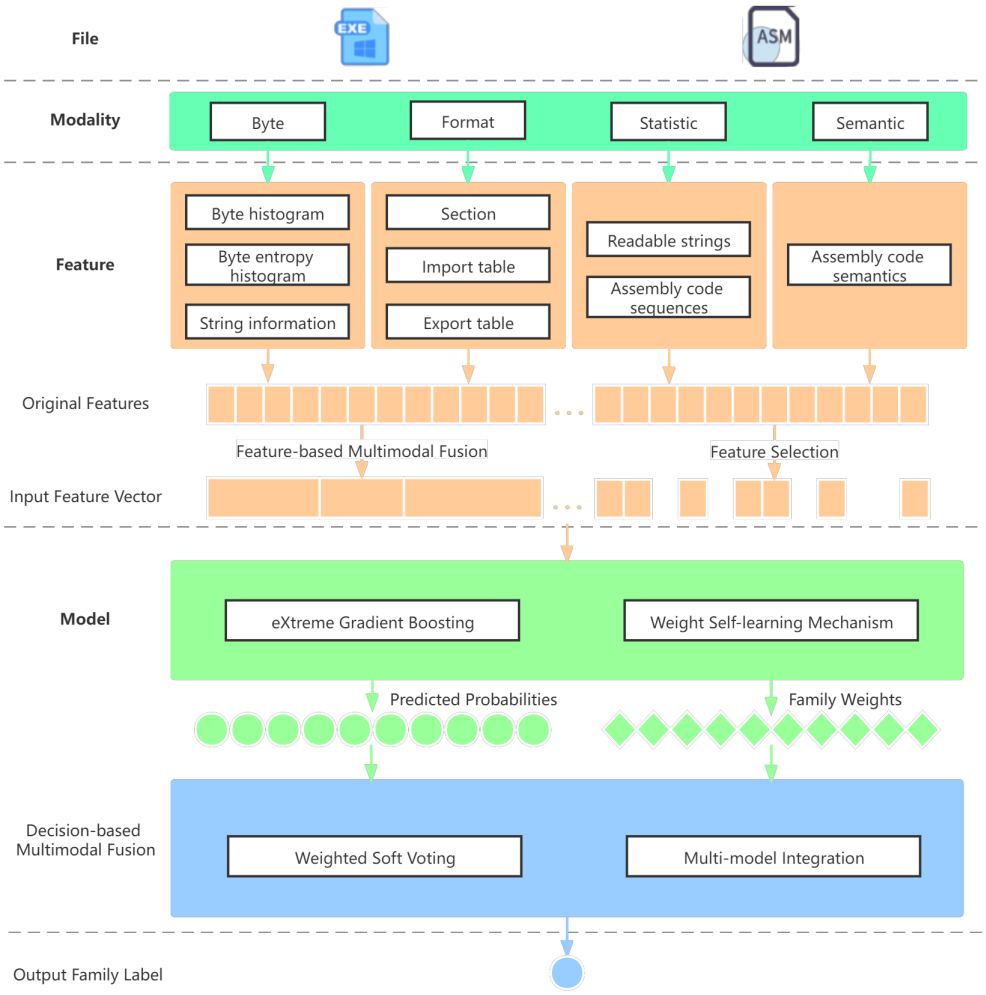

Fig. 1. The framework for imbalanced malware family classification using multimodal fusion and weight self-learning.

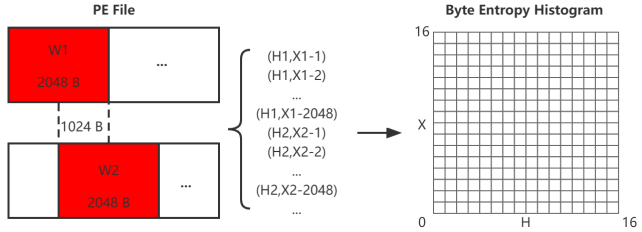

Fig. 2. Generation of byte entropy histogram.

to 0x7f, corresponding to 96 cells) of printable characters contained. And on the other hand, some substrings with special meanings may be derived based on existing printable strings. For example, a string containing "C:\" may indicate a file path, a string containing "http://" or "https://" usually indicates a URL, a string containing "HKEY_" may indicate a registry entry, and a string containing "MZ" usually means a code segment like PE file executable. These special string statistics can help us further explore the hidden attribute information in the file. By collecting the above string statistics and making them up together, we get the final string feature vector.

As for format features, we disassemble the PE file to get the assembly file, and by analyzing its contents, we find that it contains a lot of information about the structural properties and execution logic of the PE file, from which we extract the executable format features such as section and import/export table.

4) Section: In an assembly file, in addition to the assembly code that implements various logical functions, another part is the annotation that explains file properties and code functions, from which we can obtain information corresponding to certain fields in the PE file header. First, we focus on the statistics of the section in the file, including the number of sections, the number of segments, the number of sections with different exception types (section name is empty/size is 0), the number of readable/executable/writable sections, and the presence of debugging/relocation/resource/TLS sections, etc. Secondly, we focus on the basic attributes of a section, including section name, section size, and a list of strings indicating the characteristics of the section, where we refine the information on the section size, extracting the size of the section before alignment and the actual size of the space occupied by the section on disk, respectively. Furthermore, we separately extract the name of the file entry section (i.e., the first executable section) and the list of strings that represent the features of that section. In addition to the simple statistical information, we pair the values of other features with section names and perform hashing, the results of which are joined together to obtain the final section feature vector.

5) Import Table: We parse for the import address table in the annotation of the assembly file and extract the dynamic link libraries recorded and the functions imported from each one. And then hash the dynamic link libraries and the pairs made up by each link library and its inclusive functions, calculate the total number of imported functions, and finally join them together to obtain the import table feature vector.

6) Export Table: Similar to the import table, we parse the export address table in the annotation of the assembly file to extract the export functions recorded and then perform hashing to obtain the export table feature vector.

As for statistical features, we extract the readable strings from PE files and assembly code sequences from assembly files for Term Frequency-Inverse Document Frequency (TF-IDF) [29] processing, which is applied on the principle that the more a string or a code sequence appears in a sample while the less it appears in all samples, the more representative it is as for the sample.

The TF-IDF value of each element (a string or item of sequence) is equal to multiplying the values of TF and IDF. The TF and IDF are calculated as Eq. 2 and Eq. 3 respectively.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2}$$

where $TF_{i,j}$ represents the frequency of element i in sample j; $n_{i,j}$ is the times that element i emerges in sample j, and $\sum_k n_{k,j}$ means the total amount of elements in sample j.

$$IDF_i = \log \frac{|D|}{1 + |\sum i \in j|} \tag{3}$$

where $|D|$ is the total amount of samples, and $|\sum i \in j|$ means the number of samples containing element i. In order to prevent the denominator from being 0, we add 1.

7) Readable Strings: Here we define readable strings as the strings with similar patterns to English words, and on this basis, only strings between 4 and 20 in length and containing vowel letters are retained. Based on this rule, we extract a large number of readable strings from the PE file samples to build a vocabulary, and only the first 1000 words in descending order of word frequency are considered in the vocabulary. And then TF-IDF is applied to the set of samples to obtain a readable string feature matrix, where the readable string feature vector of each PE file sample corresponds to each row vector of it.

8) Assembly Code Sequences: Here we extract the opcode, the first operand, and the subsequent annotation by the line from the code segment in the assembly file, and then join them together sequentially, which serves as an element to be added to the assembly code sequence. As for the first operand, if the opcode is "call", it is usually the called function, from which we eliminate the function name prefixed with "sub", "dword" and "unknown" and then keep it; otherwise, the operand of register type is selected. Based on this rule, we generate an assembly code sequence from the assembly file samples and then use (1,3) n-gram to get a new assembly code sequence with increasing dimensionality. The procedure for generating assembly code sequences is shown in Fig. 3. After that, a vocabulary can be built from the assembly code sequence while considering only the first 1000 words in descending order of word frequency, and then TF-IDF is applied to the set of samples to obtain an assembly code sequence feature matrix, where the assembly code sequence feature vector of each assembly file sample corresponds to each row vector of it.

As for the semantic feature, similar to the assembly code sequences in the statistical feature, we extract the "condensed" assembly code semantics involving opcode, first operand, and subsequent annotation by the line from the code segment in the assembly file.

9) Assembly Code Semantics: Differ from the assembly code sequences, we take a function as the basic unit and abstract the semantic information of each function into a sentence, generate a file containing the assembly code semantics for each assembly file, and collect these files to build a corpus. The procedure of generating assembly code semantics is shown in Fig. 3. Then we use Word2Vec [30] to learn assembly code semantics from the corpus by unsupervised means, generate the corresponding word vectors, and train it to obtain an assembly code semantic model that stores the relevance vector corresponding to each assembly code sequence word in the corpus. Then, for all the assembly code sequence words extracted from each assembly file sample, we obtained their relevant vectors from the assembly code semantic model, summed and averaged them to get a new vector, defined as the relevant vectors of that sample in the model, i.e., the corresponding assembly code semantic feature vectors.

10) Feature-Based Multimodal Fusion: After the extraction of single features is completed, we consider the multimodal fusion of some of the features for creating new features with more classification discriminative power. This part uses a feature-based multimodal fusion approach, i.e., concatenating and merging between specific features to learn and exploit the correlations and interactions between the lower-level features of each modality [23]. First, we fuse byte features such as byte histogram, byte entropy histogram, and string information to obtain a completely new byte class feature. On this basis, we set up two new ways of feature fusion based on the preliminary prediction results of all the features mentioned above: on the one hand, byte class feature, section feature, and statistical features such as readable strings and assembly code sequences are fused, where only the first 300 words in descending order of word frequency are considered for generating the vocabulary of readable string features; on the other hand, considering the importance of assembly code semantic feature, we fuse it with byte class feature, section feature, and assembly code sequence feature. Finally, we replace the byte features with the new byte class feature and add two newly constructed fusion features to obtain a more complete malware family classification feature set, including the new byte class feature, format features (such as section, import table, and export table), statistical features (such as readable strings and assembly code sequences), assembly code semantic feature and two multidimensional fusion features.

11) Feature Selection: Further, we consider performing feature selection for the features generated after TF-IDF processing (i.e., statistical features and their constituent fused features) [31], and here we use the tree-based evaluator

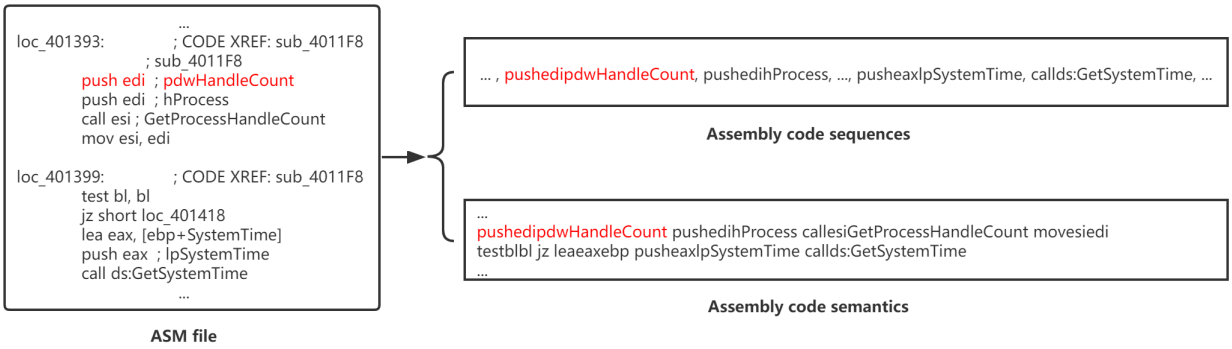

Fig. 3. Generation of assembly code sequences and semantics.

Extremely Randomized Trees Classifier (Extra Trees Classifier) [32] to calculate the importance of features and eliminate irrelevant or redundant features based on their importance. Extra Trees Classifier is a type of ensemble learning technique that aggregates the results of multiple de-correlated decision trees collected in a "forest" to output its classification result. To perform feature selection using the evaluator, during the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria (usually Gini Index) used in the decision of feature of the split is computed, which is called the Gini Importance of the feature. Then each feature is ordered in descending order according to the Gini Importance of each feature and the top k features are preferred.

It is noted that, with due consideration to the fact that the sample size of each malware family in the real environment presents an extreme imbalance, we first need to calculate the category weights of each family before selecting features, and then performed feature selection based on the weights to ensure the accuracy of the classification task.

B. Model Construction

In the model construction part, we use eXtreme Gradient Boosting (XGBoost) [33] as the base classification model, and then further apply optimized strategies such as weight self-learning and decision-based multimodal fusion to finally build a feasible model for malware family classification.

1) Weight Self-Learning Mechanism: Considering that malware is still evolving in the real environment and the generation and proportion of malware families are always in a dynamic varying pattern [34], if we can sense this change in the network environment in real-time and self-learn the distribution weights of malware families, it will help to mitigate the performance loss of classification model caused by the concept drift to and make it more robust at the same time.

First, we extract features from the malware samples participating in the training based on the malware family classification feature set obtained from the feature engineering part, feed these features individually into the base classification model for training, and then, in turn, use the trained model to predict these samples to obtain the originally predicted probability corresponding to each feature. Second, these malware samples are grouped by their families, and for each group, the values of log-loss are calculated based on the family label and the probabilities predicted by each feature, and the negative logarithmic value of each log-loss is used as the weight of the current feature on the current family. In this way, by collecting the latest generated malware samples with their families in the network environment, and then continuously updating them to the training set of the classification model to participate in training, we are able to keep the latest weights of malware families self-learned. Algorithm 1 summarizes the main procedure of weight self-learning.

Algorithm 1 Weight Self-Learning

Input : A malware family classification feature set F, a matrix of training samples D with family label $y \in N_m$, m is the number of families

Output: A matrix of weight $W_{m\times s}$, s is the size of F

Extract features from training samples first.

for $F_i \in F$ **do**
  train F_i by XGBoost to get Model M_i;
  use M_i to predict D to obtain the probability matrix P;
  for $j \leftarrow 0$ *to* m **do**
    get the sub matrix D_j with family label j;
    calculate $logloss_{ji} = -\frac{1}{n}\sum_{k=1}^{n}\log(P_{kj})$, n is the value of the first dimension of D_j;
    if $logloss_{ji} \geq 1$ **then**
      $W_{ji} = 0$
    else
      $W_{ji} = -\log(logloss_{ji})$

2) Decision-Based Multimodal Fusion: Finally, we use a decision-based multimodal fusion approach, which allows us to use different models for each modality since different models can better model each individual modality, thus making the prediction more flexible [23], [35]. Decision-based multimodal fusion consists of two steps: weighted soft voting and multi-model integration.

After obtaining the above weights, we first multiply the predicted probabilities of each model by the corresponding weights and then add them to obtain the new predicted probabilities. For each sample, we identify the class with the largest probability after summation as the sample family (i.e., the voting result), and then select the result with the predicted result consistent with the voting result and the largest predicted probability from all models participating in the voting as the

TABLE I
FAMILY DISTRIBUTION IN THE TRAINING SET OF CCF BDCI-21

Family label	Amount of samples	Population
0	214	0.073
1	373	0.128
2	10	0.003
3	130	0.045
4	160	0.055
5	90	0.031
6	388	0.133
7	675	0.231
8	297	0.102
9	582	0.199

final prediction result for that sample, thus completing the weighted soft voting process.

For each feature in the malware family classification feature set, after calculating its original predicted probability and weight on each family, we first selected the following four sets of features for the weighted soft voting process to obtain the predicted probabilities of the corresponding models.

- Byte class feature, format features such as section, import table and export table.
- Section, export table, readable strings, assembly code semantics.
- Section, export table, readable strings, the multidimensional fusion features with assembly code semantics.
- Section, export table, all two multidimensional fusion features.

Then, these four predicted probabilities are summed and averaged to integrate the decision result of the multi-model, so as to obtain the final predicted result in probability.

IV. EXPERIMENTS

A. Experimental Setup

1) Malware Family Datasets: The CCF BDCI-21 [36] dataset contains 5841 malware from 10 malware families with a size of 11.5 GB, of which the number of samples in the training set and test set are 2919 and 2922 respectively. Each malware sample corresponds to two types of files in the dataset: a PE file with header information removed and an ASM file corresponding to the PE source file generated using the disassembly tool IDA Pro. The family distribution of the training set is shown in Table I.

The Microsoft BIG-15 [37] dataset contains 10868 malware samples from 9 malware families with a size of 184 GB, of which the number of samples in the training set and test set are 5432 and 5436 respectively. Each malware sample corresponds to two types of files in the dataset: a BYTE file representing the binary content of the PE file in hexadecimal form (without PE header) and an ASM file generated using IDA Pro. In order to better apply the proposed approach in this paper to this dataset, we performed pre-processing on the files in the dataset, including converting the.bytes files to corresponding PE files, processing the.asm files to source files containing only assembly code and its annotation, and re-labeling the family tags starting from 0. The family distribution of the training set is shown in Table II.

TABLE II
FAMILY DISTRIBUTION IN THE TRAINING SET OF MICROSOFT BIG-15

Family label	Amount of samples	Population
0	770	0.142
1	1239	0.228
2	1471	0.271
3	237	0.044
4	21	0.004
5	375	0.069
6	199	0.037
7	614	0.113
8	506	0.093

2) Evaluation Metrics For Multi-Classification Issues: Accuracy acc is the proportion of the number of correctly classified samples to the total number of samples, which is given as Eq. 4.

$$acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

where TP, FP, TN, and FN are the number of samples that are true positive, false positive, true negative, and false negative respectively.

Precision P is the proportion of samples predicted to be positive that is actually positive, which is given as Eq. 5. Recall R is the proportion of samples actually positive that are predicted to be positive, which is given as Eq. 6. Macro-Precision $macro\text{-}P$ and Macro-Recall $macro\text{-}R$ are averaged after calculating the corresponding metrics for each class, which is given as Eq. 7 and Eq. 8 respectively. Macro-F1 score $macro\text{-}F1$ is based on the harmonic mean of macro-accuracy and macro-recall, which is given as Eq. 9.

$$P = \frac{TP}{TP + FP} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

$$macro\text{-}P = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{7}$$

$$macro\text{-}R = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{8}$$

$$macro\text{-}F1 = \frac{2 \times macro\text{-}P \times macro\text{-}R}{macro\text{-}P + macro\text{-}R} \tag{9}$$

where n is the number of classes, P_i and R_i are the calculated precision and recall of class i respectively.

Multi-class log-loss $logloss$ is the negative log-likelihood of the predicted sample given its true class, which is given as Eq. 10.

$$logloss = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij}) \tag{10}$$

where n is the number of samples, m is the number of classes. For y_{ij}, when the sample i belongs to the class j, it values 1, otherwise 0. p_{ij} is the probability that sample i belongs to the class j.

B. Performance of Classification

In this part, we conduct multiple sets of comparison experiments on two datasets, CCF BDCI-21 and Microsoft BIG-15,

TABLE III
CLASSIFICATION PERFORMANCE OF SINGLE FEATURES

Dataset	Modality	Feature	Accuracy	Macro-F1 score	Multi-class log-loss
CCF BDCI-21	Byte	Byte histogram	0.975	0.939	0.083
		Byte entropy histogram	0.979	0.966	0.075
		String information	0.978	0.945	0.073
	Format	Section	0.992	0.977	0.034
		Import table	0.927	0.865	0.251
		Export table	0.350	0.233	1.669
	Statistics	Readable strings	0.990	0.981	0.035
		Assembly code sequences	**0.993**	**0.989**	**0.029**
	Semantic	Assembly code semantics	0.988	0.976	0.044
Microsoft BIG-15	Byte	Byte histogram	0.979	0.958	0.073
		Byte entropy histogram	0.974	0.932	0.089
		String information	0.978	0.948	0.081
	Format	Section	0.967	0.945	0.106
		Import table	0.960	0.856	0.137
		Export table	0.361	0.155	1.722
	Statistics	Readable strings	0.979	0.927	0.063
		Assembly code sequences	**0.985**	**0.978**	**0.053**
	Semantic	Assembly code semantics	0.978	0.926	0.079

TABLE IV
CLASSIFICATION PERFORMANCE OF FUSED FEATURES

Dataset	Feature-based multimodal fusion	Accuracy	Macro-F1 score	Multi-class log-loss
CCF BDCI-21	Byte class	0.987	0.973	0.053
	Byte + Section + Statistics	0.995	**0.992**	0.020
	Byte + Section + Assembly code sequences & semantics	**0.995**	0.981	**0.019**
Microsoft BIG-15	Byte class	0.985	0.947	0.054
	Byte + Section + Statistics	**0.990**	0.976	**0.032**
	Byte + Section + Assembly code sequences & semantics	0.989	**0.981**	0.039

to evaluate the classification performance of the approach described in this paper, and validate its effectiveness in malware family classification tasks from the perspectives of both feature and model [38].

1) Effectiveness of Features: From Section III-A, we can learn that the approach proposed in this paper targets PE malware and its disassembly file, focusing on multiple modalities such as the byte, format, statistic, and semantic, based on which multi-dimensional feature-oriented feature engineering (including feature extraction, feature fusion, and feature selection, etc.) is performed based on static analysis. In order to verify the effectiveness of the extracted or constructed features, two sets of comparison experiments are set up. The first set of experiments was conducted on two malware family datasets using single features of each modality as input features and XGBoost as the classification model for the task, and the performance is shown in Table III.

As shown in Table III, the assembly code sequence feature outperforms other single features on both datasets and achieves the best value in all evaluation metrics, indicating that the file execution logic contained in the assembly code can distinguish different malware families to a certain extent. At the same time, the export table, which is missing in most of the samples, is usually difficult to achieve the desired classification performance when used as a separate feature input. In addition, for the CCF BDCI-21 dataset, features reflecting PE file format and text content, such as section and readable strings, also perform great classification performance; for the Microsoft BIG-15 dataset, in addition to readable strings, byte histogram can also achieve nice classification results. The strong classification ability exhibited by these features provides a reliable basis for feature-based multimodal fusion.

Referring to the idea of feature-based multimodal fusion, the second set of experiments still uses the XGBoost classification model, but carries out the classification task with different fused features as input features, and the performance is shown in Table IV.

As shown in Table IV, the experimental results perform some similarity between the two datasets. On the one hand, compared with the single byte features, the byte class feature obtained by fusion obviously get further improvement in classification performance; on the other hand, the two new features obtained by fusing single features with strong classification ability both perform excellent classification performance and are significantly better than the original single features, which also proves that focusing on the correlation between different modalities can highly enhance classification performance.

2) Effectiveness of Models: To verify the effectiveness of the adopted model constructing strategies, we also designed a set of comparison experiments. The experiments take the malware family classification feature set defined as input features, construct classification models with different model constructing strategies, and carry out classification tasks on two malware family datasets, and the performance is shown in Table V.

As shown in Table V, by observing different decision-based multimodal fusion methods and their experimental results, we can obtain the following two conclusions. On the one hand, compared to the combination of the best feature which has the strongest classification ability (assembly code sequence) and the base model (XGBoost), soft voting, especially the weighted soft voting using the weight self-learning mechanism, is significantly helpful to improve the classification performance, and all evaluation metrics achieve to perform

TABLE V
CLASSIFICATION PERFORMANCE OF MODEL CONSTRUCTED STRATEGIES

Dataset	Decision-based multimodal fusion	Accuracy	Macro-F1 score	Multi-class log-loss
CCF BDCI-21	Base model + Best feature	0.993	0.989	0.029
	Soft voting	0.995	0.992	0.024
	Weighted soft voting	0.995	0.992	0.022
	Weighted soft voting + Multi-model integration	**0.997**	**0.994**	**0.010**
Microsoft BIG-15	Base model + Best feature	0.985	0.978	0.053
	Soft voting	0.992	0.984	0.038
	Weighted soft voting	**0.992**	**0.985**	0.037
	Weighted soft voting + Multi-model integration	**0.992**	0.981	**0.029**

TABLE VI
EFFICIENCY AND RESOURCE OVERHEAD OF FEATURES

Dataset	Feature set	Train time (s)	Predict time (ms)	Saved model size (KB)
CCF BDCI-21	Byte class	**8.22**	55.1	937
	Section	0.88	47.3	739
	Import table	2.57	70.9	826
	Export table	0.65	45.4	640
	Readable strings	1.44	35.7	743
	Assembly code sequences	1.42	34.1	635
	Assembly code semantics	**4.48**	56.2	648
	Byte + Section + Statistics	2.28	31.3	582
	Byte + Section + Assembly code sequences & semantics	3.51	34.4	573
	Final feature set	**31.6 (8.22*)**	**140**	**6170**
Microsoft BIG-15	Byte class	**41.6**	69.0	1442
	Section	2.39	71.0	1647
	Import table	7.04	82.9	1768
	Export table	1.55	48.0	812
	Readable strings	5.01	72.0	1606
	Assembly code sequences	8.97	65.0	1351
	Assembly code semantics	**44.8**	81.0	1452
	Byte + Section + Statistics	15.0	58.4	1149
	Byte + Section + Assembly code sequences & semantics	28.7	60.0	1122
	Final feature set	**170 (41.6*)**	**210**	**12000**

* Training time under a parallel training mechanism.

extremely well. On the other hand, after combining the weighted soft voting and the multi-model integration strategies (i.e., the model constructing strategy proposed in this paper), the value of multi-class log loss significantly reduces while those of other metrics maintain relatively stable, indicating that the approach described in this paper performs more robustly and reliably in the malware family classification task.

C. Efficiency and Resource Overhead

For any effective malware family classification approach, in addition to focusing on its classification performance, its operational efficiency and various resource overheads during operation should be considered to assess its feasibility and potential value for industrial application [39]. In each set of experiments in Section IV-B, we also obtain various metrics reflecting efficiency and resource overhead when each feature in the malware family classification feature set is used as feature input respectively, including training and prediction time of models and space size occupied by saving the trained models. The relevant metrics are shown in Table VI.

As shown in Table VI, comparing the efficiency and resource overhead metrics of the proposed approach on the two datasets, it can be found that the training time of the model increases significantly as the size of the dataset expands, and the space occupied by saving the trained model also increases significantly, while the prediction time of the model rarely increases, which reflects the advantage that the approach can maintain stable efficiency when the size of test set expands continuously. For the malware family classification feature set, the training time of models with byte class feature and assembly code semantic feature as input is significantly longer than that of other features, which implies that if the aim is to further reduce the overall training time, a parallel training mechanism of models can be employed and suitable accelerated training techniques can be used for the aforementioned time-consuming features.

D. Comparison With Previous Methods

To demonstrate the superiority of the proposed approach in this paper, we investigate the existing malware family classification methods and select several representative ones for comparison. The dataset for comparison is Microsoft BIG-15, and the dimensions considered include classification performance, efficiency, and resource overhead, etc. The comparison is shown in Table VII.

As shown in Table VII, we selected typical malware family classification methods from each of the image-based, binary-based, and disassembly-based methods, and applied them to the same dataset as the approach proposed in this paper. Our approach, in terms of classification performance, performs significantly better than several other methods in terms of accuracy and macro-F1 scores, which proves to perform an

TABLE VII
COMPARISON AMONG OUR APPROACH AND OTHERS

Approach	Accuracy	Macro-F1 score	Predict time (ms)	Peak memory (MB)
IMCFN [14]	0.978	0.951	**2.23**	2299.00
MalConv [17]	0.970	0.933	14.0	2378.50
MCSC [20]	0.979	0.961	3478	2275.67
Ours	**0.992**	**0.981**	210	**369.99**

extremely strong ability to classify malware families. In terms of efficiency, its prediction speed is slightly inferior to that of raw file byte-based methods such as IMCFN and MalConv, but still much faster than that of methods such as MCSC that require constructing classification features. In terms of resource overhead, the peak memory consumption during its operation is lower, which enables it to be applied to various types of systems while the impact on system performance can be significantly attenuated.

V. CONCLUSION AND FUTURE DIRECTIONS

In this study, a multimodal fusion and weight self-learning based malware family classification approach is proposed, which achieves the family identification and classification of malware by effectively fusing multiple modalities such as the byte, format, statistic, and semantic of malware, while adding the weight self-learning mechanism of malware families to the classification model. It has many noteworthy advantages for both academic and industrial applications. (1) The feature-based and decision-based multimodal fusion approaches are used in the feature engineering and model construction stages, respectively, which effectively solves the overfitting problem. (2) A weight self-learning mechanism of malware families is added to the classification model, which is used to automatically learn the weights of different features for each family during the training process, and the weights are self-learning and transferable, which may alleviate the concept drift. (3) The designed model occupies less memory, requires less computational resources, performs fast in parallel training, and the prediction efficiency is less affected by the size of the sample set. From the experimental results, the proposed approach achieves excellent classification performance on highly imbalanced malware family datasets and provides a feasible idea to solve the concept drift problem to a certain extent.

In future research, we are committed to carrying out research in the following directions. (1) To perform code-level/core-level analysis for specific malware families (e.g., malicious mining programs and ransomware) or specific devices (e.g., automated vehicles) to achieve a more targeted detection system. (2) To consider using more feature types (e.g., malicious behavior features) for multimodal fusion while analyzing their impact on the classification performance of malware families. (3) To apply the concept of adversarial learning to the field of malware detection and classification to explore the possibility of further improving the robustness of malware classification models.

REFERENCES

[1] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIoT devices," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7059–7067, Oct. 2022.

[2] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in IoV-assisted smart city," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1373–1385, Jul. 2021.

[3] AV-TEST. (2022). *Malware Statistics and Trends Report by AV-Test*. [Online]. Available: https://www.av-test.org/en/statistics/malware/

[4] J. Li *et al.*, "Decentralized on-demand energy supply for blockchain in Internet of Things: A microgrids approach," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1395–1406, Dec. 2019.

[5] SonicWall. (2022). *2021 Sonicwall Cyber Threat Report*. [Online]. Available: https://www.sonicwall.com/2021-cyber-threat-report/

[6] T. Wang, Q. Yang, X. Shen, T. R. Gadekallu, W. Wang, and K. Dev, "A privacy-enhanced retrieval technology for the cloud-assisted Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4981–4989, Jul. 2022.

[7] A. Çayır, U. Ünal, and H. Dağ, "Random CapsNet forest model for imbalanced malware type classification task," *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102133.

[8] M. Howard, A. Pfeffer, M. Dalai, and M. Reposa, "Predicting signatures of future malware variants," in *Proc. 12th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2017, pp. 126–132.

[9] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji, and S. Mumtaz, "Emerging technologies for 5G-IoV networks: Applications, trends and opportunities," *IEEE Netw.*, vol. 34, no. 5, pp. 283–289, Sep. 2020.

[10] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[11] F. Mandiant. (2022). *M-Trends 2021: Insights Into Today's Top Cyber Trends and Attacks*. [Online]. Available: https://www.mandiant.com/resources/m-trends-2021

[12] S. Li, Y. Li, W. Han, X. Du, M. Guizani, and Z. Tian, "Malicious mining code detection based on ensemble learning in cloud computing environment," *Simul. Model. Pract. Theory*, vol. 113, Dec. 2021, Art. no. 102391.

[13] Y. Ma, S. Liu, J. Jiang, G. Chen, and K. Li, "A comprehensive study on learning-based PE malware family classification methods," in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2021, pp. 1314–1325.

[14] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, Apr. 2020, Art. no. 107138.

[15] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018, *arXiv:1804.04637*.

[16] R. Harang and E. M. Rudd, "SOREL-20M: A large scale benchmark dataset for malicious PE detection," 2020, *arXiv:2012.07634*.

[17] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole EXE," in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[18] H. Yang, S. Li, X. Wu, H. Lu, and W. Han, "A novel solutions for malicious code detection and family clustering based on machine learning," *IEEE Access*, vol. 7, pp. 148853–148860, 2019.

[19] M. Fan, S. Li, W. Han, X. Wu, Z. Gu, and Z. Tian, "A novel malware detection framework based on weighted heterograph," in *Proc. Int. Conf. Cyberspace Innov. Adv. Technol.*, Dec. 2020, pp. 39–43.

[20] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Comput. Secur.*, vol. 77, pp. 871–885, Aug. 2018.

[21] J. Chen, "A malware classification method based on basic block and CNN," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2020, pp. 275–283.

[22] Y. Ding *et al.*, "Malware classification on imbalanced data through self-attention," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Dec. 2020, pp. 154–161.

[23] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019.

[24] D. Gibert, C. Mateu, and J. Planes, "Orthrus: A bimodal learning architecture for malware classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[25] E. Snow, M. Alam, A. Glandon, and K. Iftekharuddin, "End-to-end multimodel deep learning for malware classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–7.

[26] S. Li, L. Jiang, Q. Zhang, Z. Wang, Z. Tian, and M. Guizani, "A malicious mining code detection method based on multi-features fusion," *IEEE Trans. Netw. Sci. Eng.*, early access, Mar. 8, 2022, doi: 10.1109/TNSE.2022.3155187.

[27] Y. Zhang, Q. Huang, X. Ma, Z. Yang, and J. Jiang, "Using multi-features and ensemble learning method for imbalanced malware classification," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 965–973.
[28] Y. Chai, L. Du, J. Qiu, L. Yin, and Z. Tian, "Dynamic prototype network based on sample adaptation for few-shot malware detection," *IEEE Trans. Knowl. Data Eng.*, early access, Jan. 13, 2022, doi: 10.1109/TKDE.2022.3142820.
[29] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.
[30] X. Rong, "word2vec parameter learning explained," 2014, *arXiv:1411.2738*.
[31] S. Li, Q. Zhang, X. Wu, W. Han, and Z. Tian, "Attribution classification method of APT malware in IoT using machine learning techniques," *Secur. Commun. Netw.*, vol. 2021, pp. 1–12, Sep. 2021.
[32] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Mar. 2006.
[33] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[34] J. Song, Z. Han, W. Wang, J. Chen, and Y. Liu, "A new secure arrangement for privacy-preserving data collection," *Comput. Standards Interface*, vol. 80, Mar. 2022, Art. no. 103582.
[35] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, and Z. Tian, "A novel web attack detection system for Internet of Things via ensemble classification," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5810–5818, Aug. 2021.
[36] CCF-BDCI. (2021). *Malware Family Classification Based on Artificial Intelligence*. [Online]. Available: https://www.datafountain.cn/competitions/507/datasets
[37] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," 2018, *arXiv:1802.10135*.
[38] B. Ji, Y. Li, D. Cao, C. Li, S. Mumtaz, and D. Wang, "Secrecy performance analysis of UAV assisted relay transmission for cognitive network with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7404–7415, Jul. 2020.
[39] D. Jiang *et al.*, "QoE-aware efficient content distribution scheme for satellite-terrestrial networks," *IEEE Trans. Mobile Comput.*, early access, Apr. 22, 2021, doi: 10.1109/TMC.2021.3074917.