IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 June 2023, accepted 23 August 2023, date of publication 8 September 2023, date of current version 25 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313409

RESEARCH ARTICLE

Efficient Malware Analysis Using Subspace-Based Methods on Representative Image Patterns

BENCHADI. DJAFER YAHIA M[1], BOJAN BATALO[2], AND KAZUHIRO FUKUI[3], (Member, IEEE)

[1]Degree Programs in Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8577, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Koto-ku, Tokyo 135-0064, Japan
[3]Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8577, Japan

Corresponding author: Benchadi. Djafer Yahia M (djafer@cvlab.cs.tsukuba.ac.jp)

This work was supported in part by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) Scholarship.

ABSTRACT In this paper, we propose a new framework for classifying and visualizing malware files using subspace-based methods. The rise of advanced malware poses a significant threat to internet security, increasing the pressure on traditional cybersecurity measures which may no longer be adequate. As signature-based detection is limited to known threats, sophisticated methods are needed to detect and classify emerging malware that can bypass traditional antivirus software. Using representative image patterns to analyze malware features can provide a more detailed and precise approach by revealing detailed patterns that may be missed otherwise. In our framework, we rely on subspace representation of malware image patterns; a set of malware files belonging to the same class is compactly represented by a low-dimensional subspace in high dimensional vector space. Then, we use Subspace method (SM) and its kernel extension Kernel Subspace method (KSM) to classify a malware file by measuring the angle between the corresponding input vector and each class subspace. Further, we propose a visualization framework based on subspace representation and occlusion sensitivity analysis which enables detection of critical malware features. These visualizations can be used in conjunction with the proposed classification method to aid in interpretation of results and can lead to better understanding of malicious threats. We evaluate our methods on Malimg and Dumpware datasets and demonstrate the advantage of our methods over previous single-image verification methods that are vulnerable to varying conditions. With 98.07% and 97.21% accuracy, our algorithm outperforms other state-of-the-art techniques.

INDEX TERMS Malware, malware image, subspace method, kernel subspace method, occlusion sensitivity analysis.

I. INTRODUCTION

The realm of cyber security is plagued with numerous challenges, and among them the presence of a myriad types of malicious software, often referred to only as *malware*. These malicious programs can cause serious harm to computer systems and compromise sensitive information, making it imperative for individuals and organizations to remain vigilant against such security threats. Malware is frequently used by cybercriminals to launch suspicious activities and cyberattacks. Malware refers to any computer program designed to cause harm to a system or network, including unauthorized access, theft of sensitive information, disruption of normal operations, and more. This form of software operates covertly and aims to exploit vulnerabilities in the target system, allowing the attacker to gain unauthorized access or control over the affected device. Common examples of malware include viruses, worms, Trojans, and ransomware. The proliferation of malware has become a major concern for individuals and organizations alike, as the impact of a successful attack can be substantial, leading to loss of sensitive information, financial damage, and damage to the reputation of the affected entity.

To mitigate the threat posed by malware, it is important to employ robust cybersecurity measures. Such measures should include the regular updating of software, the utilization of

The associate editor coordinating the review of this manuscript and approving it for publication was Gangyi Jiang.

antivirus programs, and the education of users. The timely identification of malware once it has infiltrated the system is of utmost importance, and can only be achieved through the implementation of effective security measures and practices.

The widespread use of wireless over-the-air (OTA) delivery mechanisms in software (SW) updates highlights the need for advanced malware analysis techniques that can accurately detect and respond, in real time, to new and emerging threats [35]. This is particularly important as OTA technology enables the seamless and efficient distribution of SW updates, but also presents a path for malware to enter a system. Attackers can leverage OTA technology to launch or replace safe files with malicious ones. Consequently, the development of robust malware analysis methodologies that can effectively differentiate between benign and malicious files is essential for ensuring the security of modern computing systems.

In this paper, we aim to address the tasks of malware classification and malware analysis, two similar but distinct problems. Especially, we emphasize the importance of accurately differentiating between benign and malicious files in the context of wireless OTA delivery mechanisms.

Malware classification is one of the most significant problems in the area of cybersecurity [28]. Malware authors introduce evasion strategies like obfuscation, encryption, packing, and so on. Traditionally, two types of methods are used to identify the malicious executable in personal computers systems: static and dynamic. A malware researcher employs specific tools like IDA-Pro [65] to analyze an executable file following its disassembly in the static method. If it is determined that the executable file is malicious, its signature is created and stored in the database for future reference. This procedure, also known as signature-based detection, is labor-intensive and time-consuming, making it potentially unfit for real-time detection [37] and [38]. Furthermore, a distinct signature is required for the detection of each malicious file, which means that these systems may be vulnerable to new types of malware. Obfuscation techniques that transform the malware binary into a self-compressed or uniquely structured binary can further hinder the effectiveness of static feature analysis.

Various traditional machine learning approaches such as support vector machine [1], k-nearest neighbors [2], random forests [3], naive bayes [4] and decision tree [5] have been used to detect and classify known malware. These approaches, while they have proven successful, are hard to interpret due to black-box nature of some machine learning algorithms and feature selection. Recently, rather than focusing on non-visible features for malware classification, Nataraj et al. [6] proposed malware Vision-based Analysis Technique, a new approach based on image processing, also known as malware visualization. They used a malware image dataset consisting of 9,342 malware samples belonging to 25 different classes. This work transformed the structure of packed binary samples into two-dimensional grayscale images. The resulting images revealed specific texture

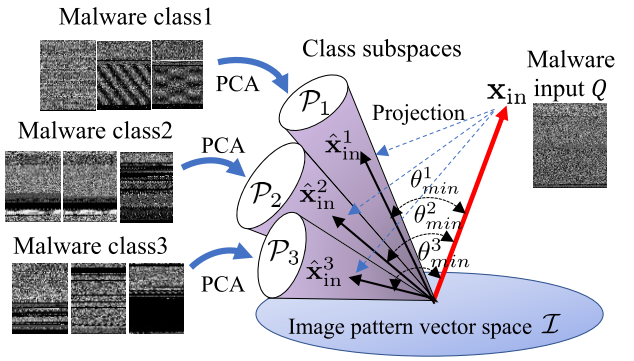


FIGURE 1. Conceptual diagram of subspace method for classifying malware image patterns into three classes.

patterns that were highly informative about the underlying malware class. However, the primary difficulty in these supervised machine learning algorithms is modeling the image set in such a way that we can effectively exploit the semantic knowledge that distinguishes malware files from benign ones.

In this paper, we introduce a novel method for malware classification and analysis that relies on subspace representation of pattern sets, which has proven successful in numerous pattern recognition problems [8], [9], [10]. A convenient way to deal with image-sets is to compactly represent them by a low-dimensional subspace of a high-dimensional feature vector space. The subspace representation is well known to be valid for representing a set of digital images, where in our case, each malware class family can be considered as a set of malware-visualized images and represented as *Malware Subspaces* through Principal Component Analysis (PCA) without data centering. This approach reveals interesting statistical information about each data distribution through their corresponding eigenvectors.

Subspace Method (SM) [8], [9], [10] is known as the typical classification method based on subspace representation. Figure 1 shows a conceptual diagram of the subspace method for classifying malware image patterns into three classes. In this method, an input malware image Q is converted to a vector $\mathbf{x}_{in}$ and then it is classified by using the similarity between $\mathbf{x}_{in}$ and the class subspace $\mathcal{P}_c$ for the c-th malware classe. Here, the similarity is defined based on the minimum angle θ^c_{min} between $\mathbf{x}_{in}$ and $\mathcal{P}_c$ or the length of the orthogonal projection of $\mathbf{x}_{in}$ onto $\mathcal{P}_c$, $\hat{\mathbf{x}}^c_{in}$. Finally, the input vector $\mathbf{x}_{in}$ is classified into the class with the smallest angle or the maximum projection length. In this paper, we will use the angle-based similarity.

However, in many image classification tasks, the data distribution cannot be exactly represented by a linear subspace, as it often contains a nonlinear structure. To address this limitation, we utilize a nonlinear extension of SM using the kernel trick, kernel subspace method (KSM) [62], [63], [64], to achieve high classification performance.

Due to the high similarity between malicious and benign files, it is often difficult for security practitioners to identify

vulnerable sections of code that might cause security concerns. To help address this issue, alongside KSM classification, we introduce a new framework, called Occlusion Sensitivity Analysis based on Kernel Difference Subspace (OSA_KDS), for enhancing and visualizing the importance of each element of a given malware image vector, aiding in discriminating the malware file from benign (safe) files.

The basic idea of OSA_KDS is to apply the Occlusion Sensitivity Analysis (OSA) to a discriminative feature vector. OSA_KDS measures the importance of a specific element in a discriminative feature vector between malware and benign classes. The importance of each element is estimated based on the change in the length of the discriminative feature vector when the element is occluded by a small window mask. The occluded element that produces a larger change degree is regarded as a more critical element in the malware analysis. The discriminative feature vector that we consider here is extracted by projecting a malware pattern vector onto a Kernel Difference subspace (KDS) [31], [34], which represents the difference between malware and benign class subspaces. By sliding the mask over the whole feature vector, OSA_KDS obtains all the elements' importance to generate a saliency map that visualizes them. This saliency map can provide a comprehensive and insightful visualization of discriminative features of a given malware file to security practitioners.

One may think that we could explicitly visualize the projection vectors. However, it is impossible to do so, as we use the kernel mapping. We can also realize the visualization of discriminative elements using the preimage technique [68]. However, this method is so complicated due to the iterative calculation that the visualization result tends to be unstable.

The main contributions of our work are:

- We propose a novel subspace representation strategy for the purpose of detecting and classifying malware families, leveraging representative digital images of each class family. The framework is simple, computationally efficient and highly interpretable, without sacrificing good performance.
- We propose a subspace-based framework for visualizing features significant for malware detection, which can serve as a starting point for reverse-engineering and studying critical parts of software that differentiate between benign and malicious behaviours.
- We evaluate our malware classification framework on two publicly available datasets and compare it to various state-of-the-art approaches, demonstrating its effectiveness.

The paper is structured as follows: Section II discusses the background and related work on various malware analysis and detection techniques; Section III introduces the detailed of the data pre-processing and Section IV presents the proposed methods for malware classification and visualization. Section V, describes the details of the datasets employed and an empirical evaluation of the experimental results obtained. Additionally, quantitative and qualitative experiments on malware classification and malware visualization are described. Finally, Section VI concludes paper and offers possible future directions.

A. MATHEMATICAL NOTATION

In this section, we summarize the notations used in this paper. We use lowercase symbols for scalars (e.g., a), bold lowercase for vectors (e.g. $\mathbf{x}$), uppercase for sets (e.g., X), bold uppercase for matrices (e.g., $\mathbf{A}$) and calligraphic for subspaces (e.g., $\mathcal{P}$). We also use the superscript $\cdot^{(s)}$ to refer to safe "*Benign*"(e.g., $\mathcal{P}^{(s)}$ refers to a subspace of benign image-set), $\cdot^{(m)}$ to refer to malicious "*Malware*" (e.g., $\mathcal{P}^{(m)}$ refers to a subspace of malware image-set), and $\cdot^{(in)}$ to refer to unknown "*Input*" image.

II. BACKGROUND

In this section we provide background on traditional malware detection systems, starting with classic static and dynamic analysis methods in subsection II-A, followed by more advanced methods relying on feature extraction in subsection II-B, and provide context for our contributions.

A. STATIC AND DYNAMIC ANALYSIS

There are two primary approaches used in malware detection and analysis: static and dynamic. While dynamic analysis requires running a program or a file and observing its behavior in real time, static analysis examines a program or a file without actually running it. Both approaches have advantages and disadvantages, and the choice of one often depends on the context and desired result. However, in order to achieve the most robust and efficient results, a comprehensive malware detection system typically employs a combination of both static and dynamic analysis methods.

Static analysis eliminates the need for a runtime inquiry based on the execution of suspicious files, as it identifies the malware sample's structure without actually running the code. Additionally, this type of analysis makes use of a number of features, including strings, opcodes, API calls, byte sequences, and control flow diagrams, all of which are disclosed from the portable executables' raw bytes (PEs) [11]. In advanced static analysis, the program commands are thoroughly examined using a disassembler, which is used to generate assembly code from machine code [12], [13]. During the analysis, the assembly instructions are thoroughly examined to identify the characteristics of malware.

In a notable example of static analysis, Tian et al. [15] used printable strings and a frequency length function to classify the malware. Their findings indicate that the frequency function can be used to identify the malware class family and can be combined with other features to classify the malicious code.

Static approaches provide a fast recognition system while utilizing significantly less computational resources, as there is no requirement for execution. However, they are often disrupted by obfuscation techniques such as instruction

reordering, register renaming, and garbage insertion that transform the malware binary into a self-compressed or uniquely structured binary can hinder the effectiveness of static feature analysis.

Dynamic analysis offers a complementary approach to static analysis in malware detection. Static detection techniques can only deal with malware that has been compressed by utilizing packers with known signatures. Therefore, dynamic analysis uses behavioral analysis to identify distinguishing patterns in suspicious files when they are processed in sandboxes and virtual computers to protect the machines from malware [39], [40] and [41]. It looks at parameters, function calls, information flows, file-registry changes, and network activities. More precisely, the main elements that the dynamic analysis focuses on can be stated as function call analysis [16], dangerous activity detection, and alteration of Windows registry entries.

Basic dynamic analysis looks into how malware behaves using monitoring tools such as Process Monitor, API Monitor, Process Explorer, Regshot, ApateDNS, Wireshark, and Sandboxes [13]. Debugging tools such as OllyDbg and WinDbg, on the other hand, are utilized in advanced dynamic analysis, and debuggers enable malware analysts to view the contents of variables, parameters, and memory areas by individually executing each command [17].

The fact that dynamic malware analysis typically necessitates significantly more resources, such as memory and CPU usage, due to the large amount of computational overhead makes it inefficient when dealing with a large dataset. Additionally, dynamic analysis may ignore certain forms of malicious code if the execution environment does not accurately replicate the actual operating conditions in which the malware operates. To uncover discriminatory features, decompression and unpacking of some portable executable (PE) files is necessary.

B. FEATURE EXTRACTION TECHNIQUES

1) N-GRAMS

N-grams are one of the most common types of features used for the purpose of identifying and classifying malware. N-grams enable the extraction of sequential patterns and features from the underlying textual content of malware samples, making it easier to identify common characteristics and behaviors. By leveraging n-grams as a means of representing and analyzing malware, security researchers are better equipped to detect and classify malicious software, and ultimately mitigate the associated risks and impacts.

Zolotukhin and Hamalainen [18] used the n-gram features to extract the opcode sequence as an input feature of malware detection, implying that using such features can lead to high efficiency in analysis as it does not require the malware to be actually executed. Additionally, Santos et al. [19] found that the statistical characteristics of both malware and benign software's opcode sequences differ, indicating that after disassembly, malware can be identified by the frequency of various opcode sequences.

Fuyong et al. [20] presented a novel approach for detecting and classifying malware using n-grams attribute similarity. Their proposed method involved selecting n-grams with the highest information gain as features and calculating the information gain of each bytes n-gram in the training samples. The averages of each attribute of the feature vectors from the malware and benign samples were then calculated separately. Lastly, a similarity metric based on Jaccard, cosine, or Tanimoto distances was then used to compare the average vectors of the two categories with the feature vector of the unknown sample, and the sample was classified as malware or benign based on the resulting similarity score.

Despite the effectiveness of n-gram approaches for detecting malware, there are several issues that warrant consideration. Firstly, Raff et al. [21] have suggested that byte n-grams tend to rely heavily on executable string content, particularly PE header items. Consequently, feature selection methods tend to prioritize frequently occurring n-grams, such as low-entropy features like strings and padding, which may limit the ability of the approach to capture a wide range of diverse and significant malware features. In addition, the exponential increase in the number of possible n-grams with larger n values makes it computationally expensive to enumerate every n-gram, leading to the curse of dimensionality. Consequently, feature reduction and selection techniques should be employed to overcome this issue.

2) GRAPH-BASED

Graph-based methods can be used for feature extraction in malware classification by representing the program behavior in a graph format. The dynamic execution of the program is analyzed to identify system calls, API invocations, and other interactions with the operating system, which are then used to build the graph. The graph's nodes symbolize events or actions, while edges represent the dependencies or ordering between them.

Park et al. [14], proposed a novel method for classifying malware based on the maximal common subgraph. A software sandbox model helped to capture malware system calls. To enable malware classification, behavior graphs are then generated from the captured system calls. The proposed method successfully classified new malware with low false-positive rates.

Kong et al. [22], used structural data to build a model in order to classify the malware. They make use of the function call graph to obtain the structural information for each malware sample. They employed the discriminate distance metric learning strategy, which clusters malware samples belonging to the same class family, in addition to the assemble of a classifier strategy, which divides malware into its various families.

Graph-based methods have proven useful in identifying both known and unknown malware. Nevertheless, the computation of pairwise graph similarity in large malware

datasets takes a considerable amount of time, which scales quadratically with the size of the dataset. Additionally, certain methods used to calculate pairwise graph similarity, such as graph matching or isomorphism, can be computationally impractical.

3) VISION-BASED

In vision-based feature extraction, malware binaries are represented as an image. A 2D array is created after the malware binary is transformed into an 8-bit vector [6]. According to previous studies, it is found that malware types, which belong to the same class family, have similar images [6], [12].

Malware classification based on image features has been performed by malware researchers using both machine learning and deep learning classifiers. Notably, texture features such as GIST, Local Binary Patterns (LBP), and Wavelet-based features have been successfully extracted from malware images [23], [24], [25]. Experimental analyses incorporated various machine learning classifiers, including Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF) [26], [27], K-Nearest Neighbor (KNN) [6], and Support Vector Machine (SVM) [6].

Dai et al. [50] proposed a method that based on memory dump data, involving the conversion of malware memory data dump binary files into grayscale images. They utilized HOG features to train a multilayer perceptron (MLP) classifier. Bozkin et al. [52] employed a combined representation with GIST and HOG features to create signatures from malware RGB images. These signatures are then used to train various machine learning classifiers such as Random Forest and linear SVM.

Such experiments across the literature have established a varying degree of success for pairing classification algorithms and different extracted image features.

Furthermore, the field of malware classification has seen the emergence of various deep learning architectures and techniques, each with its own strengths and weaknesses.

Gibert et al. [43] proposed a notable deep learning-based approach for detecting and classifying malware. Their methodology involved utilizing grayscale images of malware's binary content as inputs for a convolutional neural network (CNN). The process also relies on transforming the malware binary into an 8-bit vector, which is then used to create a 2D array representing the malware sample as a grayscale image.

In their proposed architecture, the grayscale image was fed into a CNN for further processing. The architecture consisted of three convolutional blocks, a fully connected layer, and an output layer. Each convolutional block incorporated essential operations such as convolution, ReLU activation, max-pooling, and normalization. The fully connected layers utilized the learned features to identify specific target outputs, while the convolutional layers acted as detection filters for particular data features or patterns. They tested their method on the Microsoft Malware Classification Challenge dataset [56], against hand-crafted feature extractors, and the results show that a deep learning architecture performs better at classifying malware that is represented as grayscale images. Rezende [51], on the other hand, utilized a pretrained VGG16 neural network for feature extraction; a Support vector machine (SVM) classified the input malware using the extracted features.

Although deep learning-based methods to malware detection have achieved good results, there is still a need for further research for developing robust and efficient models, especially in cases where large amounts of training data are not readily available.

One of the other open challenges in malware detection and classification lies in its multimodal nature, where multiple types of features need to be considered. Solely relying on assembly language instructions and raw byte sequences or their compressed representations as inputs can lead to a loss of valuable information crucial for characterizing malware. To address this challenge, we employed subspace model representation to identify and classify images efficiently.

III. DATA PRE-PROCESSING

In this section, the Portable Executable (PE) Malware files will be briefly discussed, followed by an explanation of the pre-processing algorithm used to convert PE files into their grayscale image.

A. PE MALWARE

Files such as executables, object code, DLLs, FON Font files, and others are saved in PE format, which is utilized in both 32-bit and 64-bit versions of the Windows operating system. Portable Executables in the 32-bit format are referred to as PE32, while PE32+ refers to Portable Executables in the 64-bit format. Figure 2 depicts a PE file's fundamental structure. A brief MS-DOS executable serves as the base for every PE file. The number of sections, the size of the "PE Optional Header," the file's characteristics, and other details about the executable are all contained in the PE Header. After the DOS header, which has a main file header and an optional header that tells how the PE file is stored, there is a PE header. The optional header entries can then be divided into two parts. The general information in the first section can be used by an operating system to load and run an executable. In addition, the data directory is the second section in which each entry specifies the section's address and size. The section table follows immediately after the Optional header. The section table provides information about all section names, locations, lengths and characteristics. Finally, the data directory is followed by section table which is a collection of section headers. Each section's messages are summarized in the fields of section headers.

B. BINARY TO IMAGE

The widespread use of Nataraj et al.'s [6] invention of binary-to-image conversion, has provided numerous new

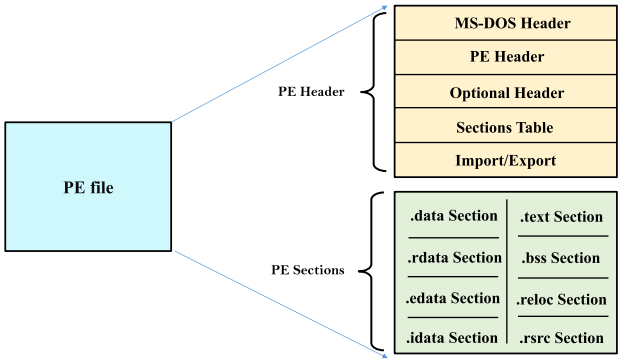


FIGURE 2. PE file structure.

anti-malware research ideas. This type of binary file visualization has bridged the gap between computer vision and executable byte-level sequences. Further, it allows practitioners and researchers to better comprehend malware structure because the patterns contained within such images are easily discernible.

There are typically a number of methods for transforming binary code into images [45]. It is also possible to identify malware families by using the most significant patterns of features in the malware images. The process of converting malware binary files into grayscale images is depicted in Figure 3. An algorithm that converts a binary PE file into a sequence of 8-bit vectors or hexadecimal values can be used to translate malware binaries into images. The malware binary file is first read in a vector of unsigned 8-bit integers and can be represented in the range [0,255] (0:black, 255:white). Then we convert the binary value of each component into its equivalent decimal value. Finally, the resulting decimal vector is reshaped to a 2D matrix and then interpreted as a grayscale image. The size of the malware binary file largely determines the 2D matrix's width and height.

IV. PROPOSED METHOD

In this section, we explain the algorithms of Subspace Method and Kernel Subspace Method, where we describe how malware classification is accomplished using subspace representation and its kernel extension. We then propose malware visualization framework based on occlusion sensitivity analysis.

A. MALWARE SUBSPACE REPRESENTATION

The subspace-based methods have shown great performance in various applications [29], [30], [31], [32], [33], [34], as they can effectively capture the data variation and stably output the similarity between a vector and a subspace or two subspaces. It involves creating a subspace representation per image set, calculating similarities between an input vector/subspace and class subspaces, and finally performing the classification. In related work, subspace representation has been widely used to preserve significant features of high-dimensional data, resulting in improved classification accuracy and efficiency. Given the increasing complexity and sophistication of malware attacks, it is becoming more challenging to detect and prevent these attacks using traditional classification methods. Therefore, we propose to leverage subspace representation to address these challenges in malware classification.

A set of n malware images is arranged into a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, where each image $\mathbf{x}_i$ is a d-dimensional vector. The orthogonal normal basis $\mathbf{B}$ of a d_s-dimensional subspace $\mathcal{P}$ is obtained by applying the principal component analysis (PCA) to a set of image patterns of the class. In practice, we can obtain $\mathbf{B}$ by computing the SVD of $\mathbf{X}$ as follows:

$$\mathbf{X} = \mathbf{U}\Lambda\mathbf{V}^\top, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{d\times n}$ contains eigenvectors as columns and $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing eigenvalues. $\mathbf{B}$ is represented as $[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{d_s}]$ using the d_s eigenvectors of $\mathbf{U}$, corresponding to the d_s highest eigenvalues.

This procedure yields a low-dimensional subspace for the given image set, which can preserve certain information such as the distribution of pixel values and structural features relevant to malware analysis.

B. SUBSPACE METHOD FOR CLASSIFICATION

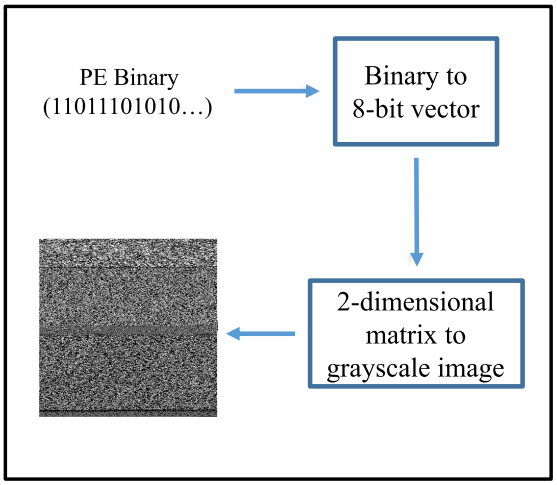
Subspace Method (SM) is a classification method based on computing similarities between an input vector and class subspaces. The similarity of an input vector $\mathbf{x}_{in} \in \mathbb{R}^d$ to the c-th class subspace $\mathcal{P}_c$ is defined based on the minimum angle θ^c_{min} between $\mathbf{x}_{in}$ and $\mathcal{P}_c$ as shown in Fig 1, by the following equation:

$$sim(\mathbf{x}_{in}, \mathcal{P}_c) = \cos^2\theta^c_{min} = \sum_{i=1}^{d_s} \frac{(\mathbf{x}_{in}\cdot\mathbf{u}^c_i)^2}{||\mathbf{x}_{in}||^2}, \tag{2}$$

where d_s is the dimension of $\mathcal{P}_c$, $\mathbf{u}^c_i$ is the i-th d-dimensional orthogonal basis vector of $\mathcal{P}_c$, and $(\cdot)$ is an inner product.

For our malware classification task containing C malware classes where each class has $\{\mathbf{x}^c_i\}^n_{i=1}$, SM works as follows: (I) Classes subspaces $\{\mathcal{P}_c\}^C_{c=1}$ are created using Eq. (1), (II) an input vector $\mathbf{x}_{in}$ is created from a single input malware image. Then, $\mathbf{x}_{in}$ is compared to class subspaces $\{\mathcal{P}_c\}^C_{c=1}$ using Eq. (7) and is classified into the class with the highest similarity.

While SM is effective at classifying many types of malware, it may not perform as well when attempting to distinguish between classes that exhibit significant inter class variation. Due to the strong non-linearity and significant variations in appearance within malware images as depicted in Figure 3, SM may not be fully effective on its own. To address this limitation, we introduce the Kernel Subspace Method (KSM) [62], [63], [64] as an extension for SM, which is well-suited for classifying the complex and diverse features of malware images.



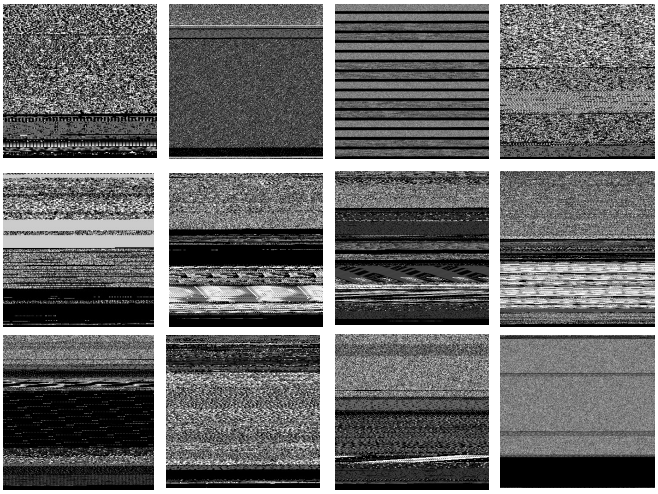


FIGURE 3. Pre-processing of malware files.

We apply KSM to the malware classification task, as it can handle this non-linearity resulting from the large variation of image appearance, leading to higher accuracy in classifying the malware images. Moreover, KSM is computationally efficient and requires only a few parameter adjustments, making it a practical and effective solution for malware classification.

C. KERNEL SUBSPACE METHOD FOR CLASSIFICATION

Figure 4 illustrates the flow of the KSM algorithm for malware classification with $C(=3)$ classes.

Given n images $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ in a d-dimensional image pattern vector space $\mathcal{I}$, they will be mapped onto an f-dimensional feature space $\mathcal{F}$ using a nonlinear mapping function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^f$.

The inner product $(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$ between their respective function values need to be calculated so that PCA can be applied to the mapped images. However, this calculation can be challenging, as the dimension of the feature space $\mathcal{F}$ can be very high or even infinite. The kernel trick provides a solution by defining the kernel mapping ϕ through a kernel function $k(\mathbf{x}, \mathbf{y})$ which satisfies Mercer's conditions, allowing for the calculation of inner products $(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$ of the mapped images from the inner products of the original input patterns $(\mathbf{x} \cdot \mathbf{y})$. A common choice is to use the Gaussian kernel function [36], defined as:

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{\sigma^2}\right). \tag{3}$$

The PCA of the mapped images is referred to as the kernel PCA [36], results in a nonlinear subspace in the input space $\mathcal{I}$.

Consider that the d_s-dimensional nonlinear subspace $\mathcal{V}_c$ of the c-th class is generated from n training images $\{\mathbf{x}_j^c\}_{j=1}^n$. The i-th orthonormal basis vector $\mathbf{e}_i^c$ of $\mathcal{V}_c$ can be represented by the linear combination of $\{\phi(\mathbf{x}_j^c)\}_{j=1}^n$ as folows:

$$\mathbf{e}_i^c = \sum_{j=1}^{n} a_{ij}^c \phi(\mathbf{x}_j^c), \tag{4}$$

where the coefficient a_{ij}^c is the j-th component of the eigenvector $\mathbf{a}_i$ that corresponds to the i-th largest eigenvalue λ_i of the $n \times n$ matrix $\mathbf{K}$, which is defined by the following equation:

$$\mathbf{K}_{ij} = (\phi(\mathbf{x}_i^c) \cdot \phi(\mathbf{x}_j^c)). \tag{5}$$

$\mathbf{a}_i^c$ is normalized to satisfy $\lambda_i(\mathbf{a}_i^c \cdot \mathbf{a}_i^c) = 1$. The projection of the mapped $\phi(\mathbf{x}_{in})$ onto the i-th orthonormal basis vector $\mathbf{e}_i^c$ of the nonlinear subspace $\mathcal{V}_c$ can be computed using the following equation:

$$(\phi(\mathbf{x}_{in}) \cdot \mathbf{e}_i^c) = \sum_{j=1}^{n} a_{ij}^c k(\mathbf{x}_{in}, \mathbf{x}_j). \tag{6}$$

Based on the above, the similarity between the kernel mapping $\phi(\mathbf{x}_{in})$ of an input vector $\mathbf{x}_{in}$ and the c-th nonlinear class subspace $\mathcal{V}_c$ is defined by the following equation:

$$sim(\mathbf{x}_{in}, \mathcal{V}_c) = \sum_{i=1}^{d_s} \frac{(\phi(\mathbf{x}_{in}) \cdot \mathbf{e}_i^c)^2}{||\phi(\mathbf{x}_{in})||^2}, \tag{7}$$

where d_s is the dimension of the class subspace $\mathcal{V}_c$ and $||\phi(\mathbf{x}_{in})||^2 = 1$ when using the Gaussian kernel function (Eq.3). Then, $\mathbf{x}_{in}$ is classified into the class with the highest similarity.

D. MALWARE VISUALIZATION BASED ON OCCLUSION SENSITIVITY ANALYSIS (OSA)

In this section, we explain our new proposed framework, Occlusion Sensitivity Analysis based on Kernel Difference Subspace (OSA_KDS), for visualizing the discriminative

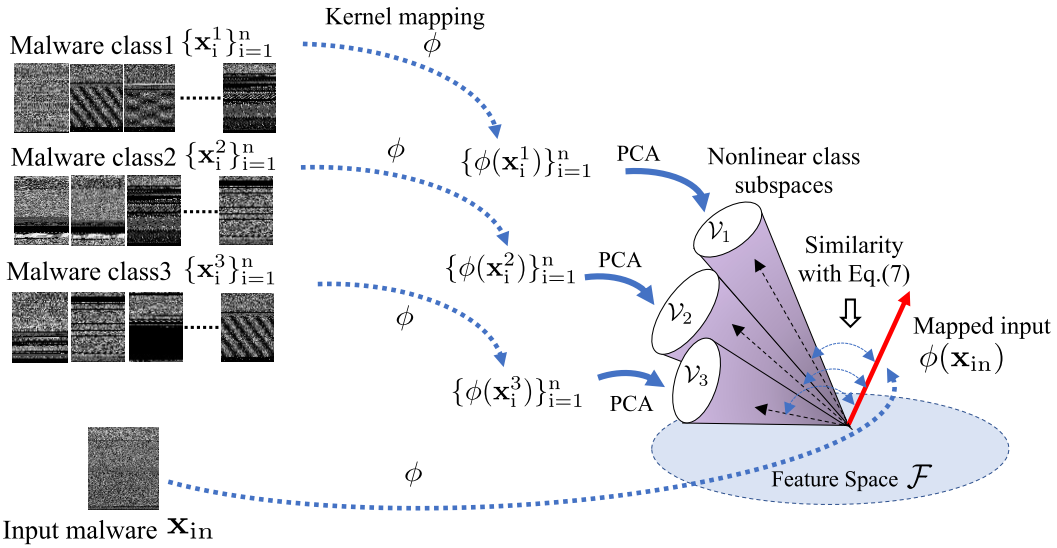


FIGURE 4. The process flow of kernel subspace method (KSM) in malware classification.

elements of a given malware image pattern. In the following, we assume that malware and benign files are represented in a vector form with d dimensions.

1) BASIC IDEA OF OSA_KDS

Figure 5 shows the overview of OSA_KDS for malware visualization. OSA_KDS applies the Occlusion Sensitivity Analysis (OSA) to a discriminative feature vector extracted from a given malware vector. OSA estimates the importance of a specific element of a given feature vector depending on the change degree in the class score when the element is occluded by a small mask. Here, the occluded element with a larger change in class score is regarded as a more critical one. Motivated by this simple idea, the essence of OSA_KDS is to measure the change degree in the length of a discriminative feature vector when a specific element is occluded by a small window mask. Similarly, the occluded element with a larger length change is regarded as a more critical one. A discriminative feature vector is extracted by projecting a malware image vector onto a kernel difference subspace (KDS) [31], [34] that represents the difference between malware and benign class subspaces.

2) KERNEL DIFFERENCE SUBSPACE BETWEEN MALWARE AND BENIGN CLASS SUBSPACES

We elaborate on the fundamental component of OSA_KDS, kernel difference subspace (KDS) [31], [34]. KDS represents the difference components between two nonlinear subspaces, and is essentially a nonlinear and multi-dimensional generalization of a difference vector between two vectors.

To generate KDS between d_s-dimensional malware and benign (safe) class subspaces for our problem, we constructed our private data set of safe grayscale images generated from PE benign files collected from different operating system safe programs. We utilized Nataraj [6] approach to generate the grayscale images from the safe files.

Let the gallery set of n images from *Safe* and *Malware* classes, denoted as $X^{(s)} = [\mathbf{x}_1^{(s)}, \ldots, \mathbf{x}_n^{(s)}]$ and $X^{(m)} = [\mathbf{x}_1^{(m)}, \ldots, \mathbf{x}_n^{(m)}]$, respectively. The orthonormal basis vectors of the d_s-dimensional nonlinear class subspaces for safe and malware, $\mathcal{P}^{(s)}$ and $\mathcal{P}^{(m)}$, are obtained by applying KPCA to the training patterns as described in Sec. IV-C.

Let $\mathbf{E}$ be the matrix that contains all the orthonormal basis vectors of the two nonlinear subspaces as columns: $\mathbf{E} = [\mathbf{e}_1^{(s)}, \ldots \mathbf{e}_{d_s}^{(s)}, \mathbf{e}_1^{(m)}, \ldots \mathbf{e}_{d_s}^{(m)}]$. Then, we calculate the matrix $\mathbf{D}$, defined as $\mathbf{E}^\top \mathbf{E}$. We obtain each element of the matrix $\mathbf{D}$ by calculating the inner product between the i-th orthonormal basis vector $\mathbf{e}_i^{(s)}$ of the Safe class subspace and the j-th orthonormal basis vector $\mathbf{e}_j^{(m)}$ of the Malware class subspace as follows:

$$\mathbf{D}_{ij} = (\mathbf{e}_i^{(s)} \cdot \mathbf{e}_j^{(m)}), \tag{8}$$

$$= \left(\sum_{l=1}^{n} a_{il}^{s} \phi(\mathbf{x}_l^{(s)}) \cdot \sum_{l'=1}^{n} a_{jl'}^{m} \phi(\mathbf{x}_{l'}^{(m)}) \right), \tag{9}$$

$$= \sum_{l=1}^{n} \sum_{l'=1}^{n} a_{il}^{s} a_{jl'}^{m} (\phi(\mathbf{x}_l^{(s)}) \cdot \phi(\mathbf{x}_{l'}^{(m)})), \tag{10}$$

$$= \sum_{l=1}^{n} \sum_{l'=1}^{n} a_{il}^{s} a_{jl'}^{m} k(\mathbf{x}_l^{(s)}, \mathbf{x}_{l'}^{(m)}). \tag{11}$$

Finally, KDS is subspace spanned by the eigenvectors corresponding to the d_s smallest eigenvalue of the matrix $\mathbf{D}$ [31], [34]. The i-th eigenvector $\mathbf{d}_i$ of $\mathbf{D}$ is represented as $\sum_{j=1}^{d_s \times 2} b_{ij} \mathbf{e}_j$, where $\mathbf{e}_j$ is the j-th column of the matrix $\mathbf{E}$, and the coefficient b_{ij} is the j-th component of the eigenvector $\mathbf{b}_i$ that corresponds to the i-th smallest eigenvalue β_i of the matrix $\mathbf{D}$. $\mathbf{b}_i$ is normalized such that $\beta_i(\mathbf{b}_i, \mathbf{b}_i) = 1$.

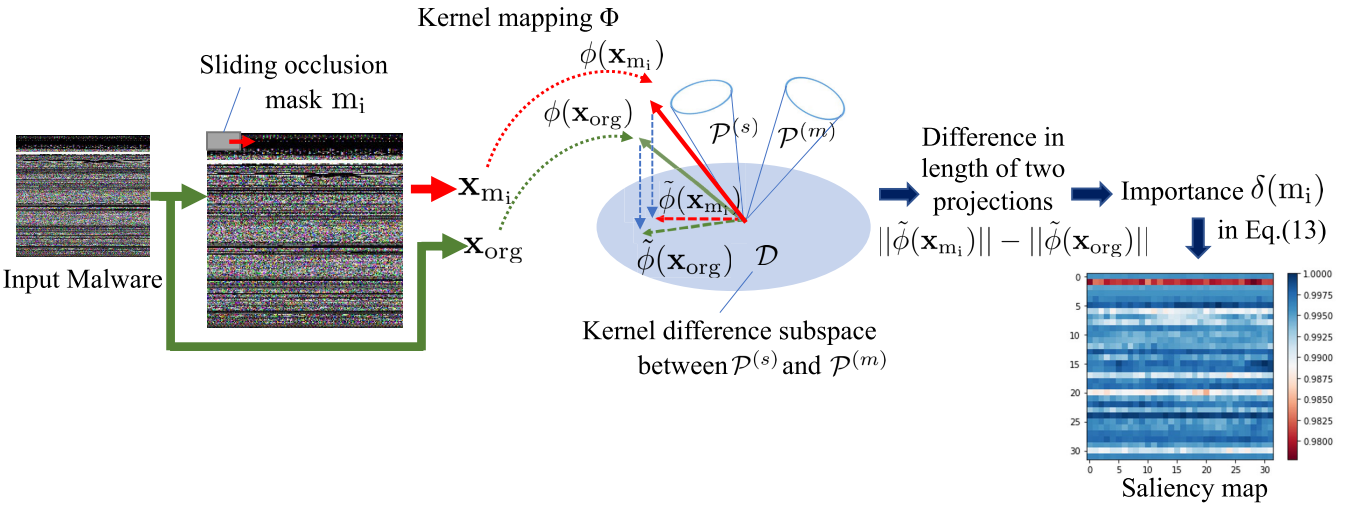


FIGURE 5. Overview of malware visualization framework based on OSA_KDS.

3) PROJECTION OF A MALWARE IMAGE VECTOR ONTO KDS

We project a kernel mapped malware vector $\phi(\mathbf{x}^{(m)})$ onto the d_s-dimensional KDS to extract the discriminative feature vector $\widetilde{\phi}(\mathbf{x}^{(m)})$ as follows:

$$\widetilde{\phi}(\mathbf{x}^{(m)}) = (z_1, z_2, \ldots, z_{d_s}), \tag{12}$$

$$z_i = (\phi(\mathbf{x}^{(m)}) \cdot \mathbf{d}_i) = (\phi(\mathbf{x}^{(m)}) \cdot \sum_{j=1}^{d_s \times 2} b_{ij}\mathbf{e}_j). \tag{13}$$

Note that the last equation can be calculated by using Eq.(6).

4) PROCESS FLOW OF OSA_KDS

Figure 5 illustrates the whole process of OSA_KDS. Each step is summarized as follows:

1) We generate KDS, $\mathcal{D}$, between d_s-dimensional malware and benign class subspaces, $\mathcal{P}^{(s)}$ and $\mathcal{P}^{(m)}$.
2) We project kernel mapped feature vector $\phi(\mathbf{x}_{org})$ of the original malware image pattern onto $\mathcal{D}$ by using Eq.(12). The projection is represented by $\widetilde{\phi}(\mathbf{x}_{org})$.
3) We occlude a specific element of $\mathbf{x}_{org}$ by a mask m_i and then obtain its kernel mapped $\phi(\mathbf{x}_{m_i})$.
4) We project kernel mapped feature vector $\phi(\mathbf{x}_{m_i})$ onto $\mathcal{D}$ by using Eq.(12). The projection is represented by $\widetilde{\phi}(\mathbf{x}_{m_i})$.
5) We compare the length of the projection, $\widetilde{\phi}(\mathbf{x}_{m_i})$ with that of $\widetilde{\phi}(\mathbf{x}_{org})$ to calculate the difference $\delta(m_i)$ between them as the importance for discriminating malware and benign classes as follows:

$$\delta(\boldsymbol{m}_i) = 1.0 - max(||\widetilde{\phi}(\mathbf{x}_{org})|| - ||\widetilde{\phi}(\mathbf{x}_{m_i})||, 0), \tag{14}$$

where negative differences are discarded. The element that produces a larger $\delta(m_i)$ is regarded as more important.
6) The differences δ of all elements are obtained by sliding the occlusion mask over the whole malware pattern vector.
8) All the importances $\{\delta(m_i)\}$ are intergrated and shown as a saliency map to provide a comprehensive and insightful visualization of discriminative features of the malware file.

V. EXPERIMENTS AND DISCUSSIONS

This section presents an empirical evaluation of the results obtained by our proposed methods. First, in Subsection V-A we introduce the performance metrics used in this study and in Subsection V-B we describe the datasets used for our experiments. Then, in subsections V-C and V-D we detail our quantitative and qualitative experiments on malware classification and malware analysis.

A. PERFORMANCE METRICS

For the sake of completeness, we define the performance metrics used to evaluate our approach; concretely, we will report four metrics: Accuracy, Precision, Recall, and F1-score.

The accuracy is simply defined as the fraction of total predictions that are correct. More formally, it is defined as follows:

$$Accuracy(\%) = 100\% \times \frac{\text{Correct predictions}}{\text{Total predictions}}. \tag{15}$$

While accuracy is a standard metric used for model evaluation and is a good estimator of model quality, for security-related applications it is useful to investigate other metrics. More specifically, it is often useful to evaluate a model in terms of *true positives (TP)*, *true negatives (TN)*, *false positives (FP)* and *false negatives (FN)*. Consider a dataset with n total samples and C classes (e.g. malware families). The above-mentioned building blocks of advanced metrics are defined as:

- TP_i refers to the number of samples correctly classified as belonging to class family $i \in C$.

- TN_i refers to the number of samples correctly classified as **not** belonging to class family $i \in C$.
- FP_i refers to the number of samples incorrectly classified belonging to class family $i \in C$.
- FN_i refers to the number of samples incorrectly classified as not belonging to class family $i \in C$.

In those terms, Precision is defined as the number of true positives over the sum of true and false positives:

$$Precision = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FP_i)}. \quad (16)$$

and Recall as the number of true positives over the sum of true positives and false negatives.

$$Recall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FN_i)}. \quad (17)$$

While Precision measures the percentage of actual malware among those identified as malware by the model, Recall measures the percentage of malware successfully identified. In ideal situation, a model would achieve both high Precision and Recall, but often these two measures are conflicted: increasing Precision decreases Recall and vice-versa. A metric unifying these two measures is the F1_score, a weighted average of Precision and Recall, defined as follows:

$$F1\_score = \frac{2 \times Recall \times Precision}{Recall + Precision}. \quad (18)$$

B. DATASETS

In this section we briefly describe public datasets used in our experiments for the tasks of malware classification and malware analysis.

1) MALIMG DATASET

Malimg [6] is a public benchmark dataset, hosted on data science platform Kaggle [66], and comprises 9,339 malware samples belonging to 25 different malware classes. These classes are: Adialer.C, Agent.FYI, Allaple.A, Allaple.L, Alueron.gen!J, Autorun.K, Benign, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator. AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, VB.AT, Wintrim.BX, and Yuner.A. Additionally, the number of samples belonging to a malware class differs across the dataset, making it a highly-imbalanced dataset. Malware samples in this dataset are publicly provided as images of various sizes, with an average size of 510 × 410 pixels. For our experiments, we uniformly resized all samples to 32 × 32 pixels images.

2) DUMPWARE DATASET

The Dumpware10 malware dataset covers 4,294 portable executables and is entirely based on memory forensics. The memory dump files were obtained through the creation of a virtual Windows 10 environment, and the malware files were stored in PNG format. The dataset is divided into 11 categories that include 10 malware classes and one benign class; it comprises 3,686 malware samples collected from diverse malware families and 608 benign samples.

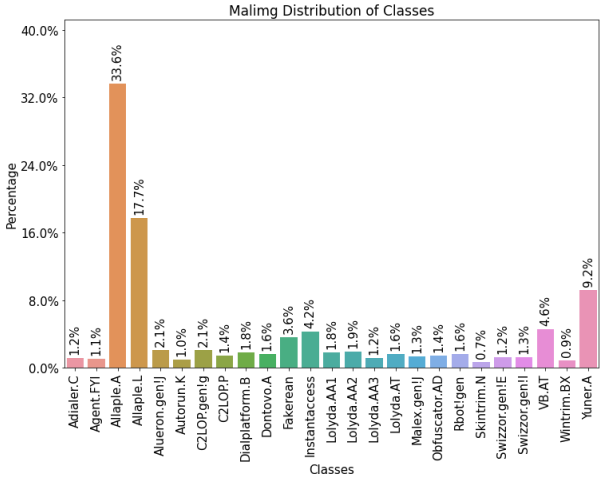


FIGURE 6. Detailed malware distribution of Dumpware dataset.

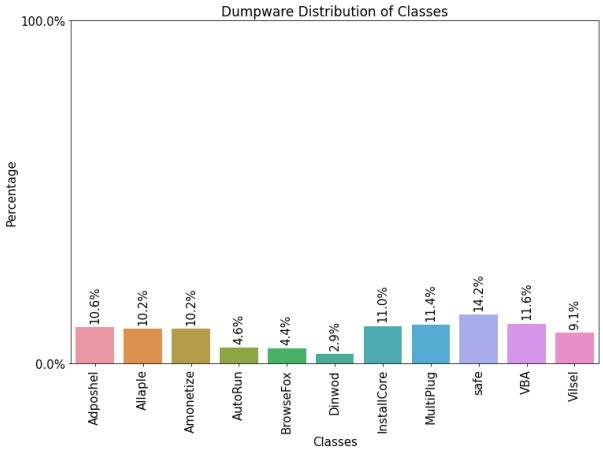


FIGURE 7. Detailed malware distribution of Dumpware dataset.

Malware binaries were converted into different greyscale images by setting the rendering parameter called "column widths" [52], resulting in images of 224 × 224 and 300 × 300 pixels, which we use. For our initial set of experiments, we resized the 224 × 224 images into 32 × 32 images, resulting in a 1024-dimensional feature vector. However, for the sake of comparison with competing methods, for the second part of the classification experiment we used 224 × 224 pixels images without resizing.

The distribution of malware samples in both Malimg and Dumpware datasets are given in Figures 6 and 7.

C. EXPERIMENT ON MALWARE CLASSIFICATION

In this experiment, our goal is to evaluate the quality of our proposed malware classification framework based on

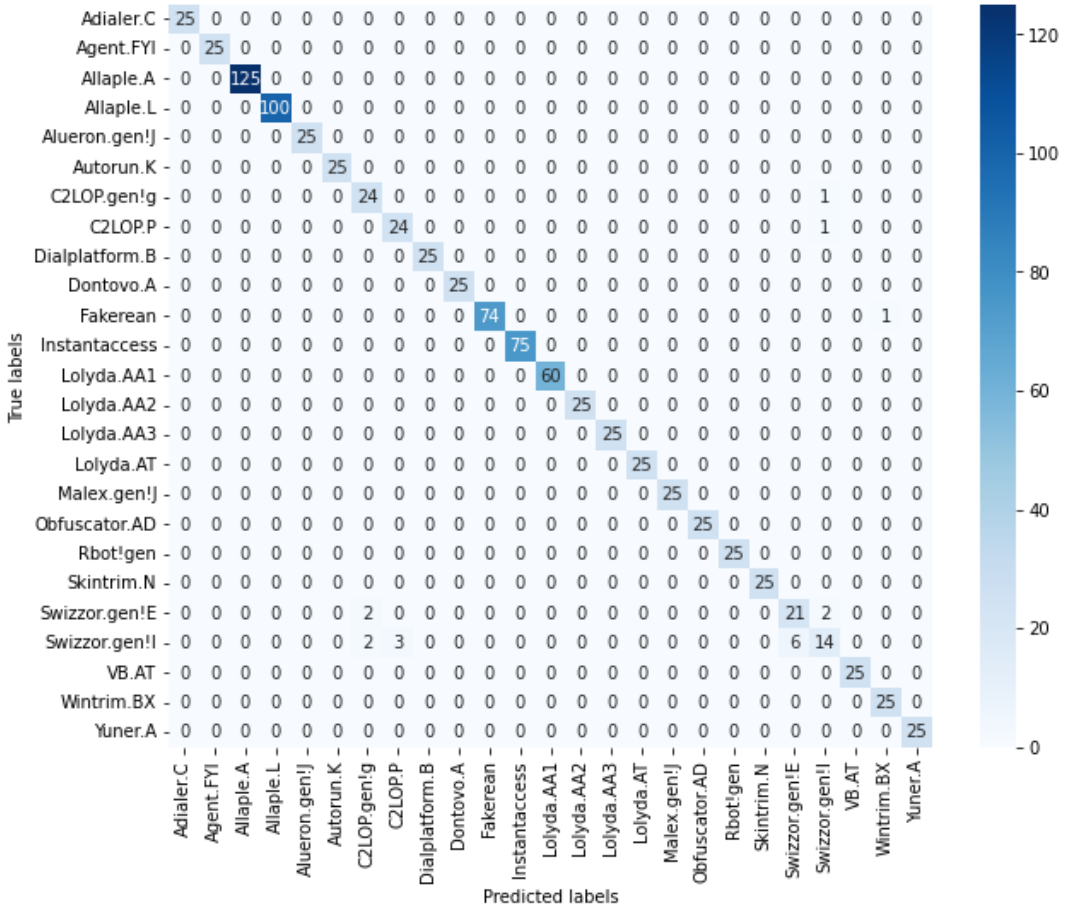


FIGURE 8. KSM confusion matrix on Malimg dataset for malware classification.

kernel subspace method (KSM). We utilize both Malimg and Dumpware datasets to this end; in case of Malimg dataset, the task is malware class family classification, and for Dumpware dataset, the task consists of classifying between malware families and benign files. We compare the proposed KSM framework to established state-of-the-art methods in the task of malware classification for given datasets and discuss results, pros and cons of our proposed approach.

The experimental setup followed the author protocol for each benchmark dataset. For the Malimg dataset, we divided it into a training and testing set in a 90:10 ratio. The training set consisted of 8,405 samples, while the testing set comprised 934 samples, collectively representing 25 different malware families. For the Dumpware dataset, we utilized 4,294 samples, including 10 malware families and a single class of benign executables.

1) RESULTS AND DISCUSSION

The results of the experiment, presented in Tables 1 and 2, demonstrate that the proposed KSM showed a higher accuracy compared to existing systems discussed in the literature, with 98.07% and 97.09% on Malimg and Dumpware malware datasets, respectively. The proposed model's validity, low complexity, and interpretability make it a viable alternative to deep learning-based methods, and provides more opportunity to tune hyperparameters.

In KSM, since the kernel trick makes use of a Gaussian exponential function, the value of σ is an additional hyperparameter to take into account. For optimal performance, we found that setting σ to 0.1 consistently achieved the best results on both datasets. In our experiments, the dimensions of the reference subspaces were selected empirically. Before generating the subspaces, we can suggest if the data is proper or not for a subspace representation, by observing the distribution of the eigenvalues computed when performing PCA. We observed that retaining a small number of eigenvectors, typically between 8-30 in the case of Malimg dataset, and between 15-45 with Dumpware dataset, resulted in the highest classification performance in terms of accuracy, precision, recall, and F1-score metrics. Moreover, these eigenvectors possess the highest discriminative power, while increasing the the subspace dimension may introduce more noise than meaningful information, leading to reduced classifier performance.

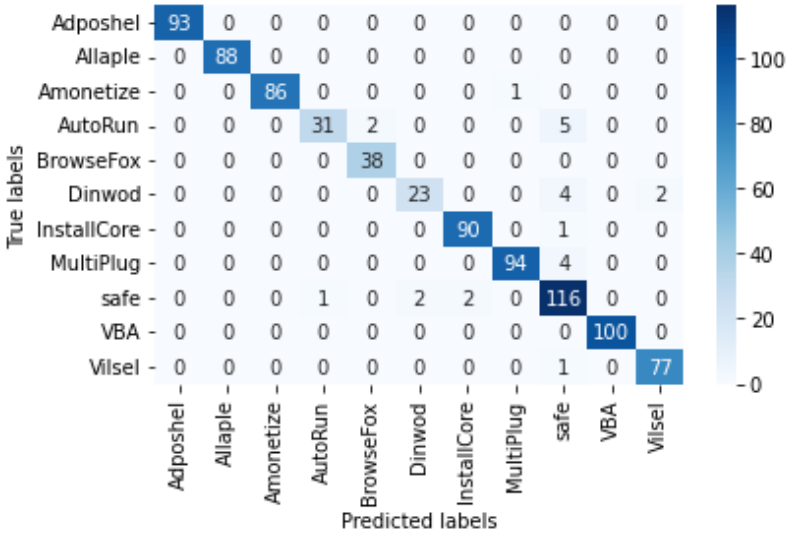


FIGURE 9. KSM confusion matrix on Dumpware dataset for malware classification.

TABLE 1. Malware classification performance comparison with state-of-the-art approaches on the Malimg dataset.

Method	Acc (%)	Precision	Recall	F1
Agarap et al.,2017 [46]	84.92	85.0	85.0	85.0
Luo et al.,2017 [48]	93.72	/	/	/
Cui et al.,2018 [47]	94.05	94.6	94.5	94.5
Vinyakumar et al.,2019 [26]	96.30	96.3	96.2	96.2
Singh et al., 2019 [59]	96.08	/	/	/
Roseline et al., 2020 [60]	97.49	97.0	97.0	97.0
Awan et al., 2021 [61]	97.62	/	/	/
Aslan et al., 2021 [49]	97.78	/	/	/
KSM (ours)	**98.07**	**97.0**	**97.2**	**97.0**

TABLE 2. Malware classification performance comparison with state-of-the-art approaches on the Dumpware dataset.

Method	Acc (%)	Precision	Recall	F1
Logistic reg.	89.40	89.8	89.4	87.1
Naïve bayes	80.40	85.4	80.4	82.2
K-near neighbor	94.80	94.7	94.8	94.5
Decision tree	87.10	86.8	87.1	86.9
Random forest	91.50	91.4	91.5	90.4
DNN	84.70	72.0	85.0	78.0
GRU	94.90	95.0	95.0	95.0
RNN	85.90	91.0	86.0	87.0
LSTM	94.50	94.0	95.0	94.0
KSM(ours)	**97.09**	**97.0**	**95.4**	**96.0**

Figures 8 and 9 depict confusion matrices, where the actual category of each malware class family is represented by the ordinate of the confusion matrix, and the prediction category of the malware class family is represented by the horizontal coordinate; each value on the diagonal from [0,0] to [25,25] (for the Malimg dataset) and [0,0] to [11,11] (for the Dumpware dataset) in the confusion matrix represents the correct predicted labels of each class. The biggest errors for Malimg are in distinguishing between C2LOP.gen!g and C2LOP.P on one side, and Swizzor.gen!E and Swizzor.gen!I. In the case of Dumpware, KSM is very good at predicting malware class, marking only several malware files as safe.

Finally, a comparative analysis was conducted to evaluate the proposed KSM method against state-of-the-art approaches. Tables 1, 2 and 3 display the respective accuracy values of the proposed method and other state-of-the-art studies on the Malimg and Dumpware benchmark datasets. Notably, the proposed method demonstrated superior performance compared to existing algorithms.

In comparison to the existing methods, Aslan et al. 2021 [49] proposed a hybrid approach that combined AlexNet and Resnet-152 for malware detection, achieving an accuracy of 97.62%. While their method showed superior performance compared to its constituent models, the approach may struggle to effectively detect complex malware variants that employ packing and obfuscation techniques. Additionally, their approach investigated the use of additional hidden layers in deep learning to further enhance performance, which increases model complexity. Similarly, in [61], an image-based malware classification system incorporating a spatial attention mechanism was proposed and evaluated using the Malimg dataset, achieving an accuracy of 97.78%. However, deep learning models often require a substantial amount of data, making them less effective against malware disguised with obfuscation techniques. In contrast, our KSM framework outperformed both methods, achieving an accuracy of 98.07%, precision of 97.0, recall of 97.2, and

TABLE 3. Malware classification performance comparison with state-of-the-art approaches on the Dumpware dataset with bigger feature vector size.

Method	Image size	Acc (%)	Precision	Recall	F1
Nataraj et al., 2011 [6]	N/A	91.40	91.5	91.4	91.5
Dai et al., 2018 [50]	224 × 224 pixels	94.50	94.6	94.5	94.5
Rezende et al., 2018 [51]	224 × 224 pixels	96.90	**97.0**	**96.9**	**96.9**
Bozkir et al., 2021 [52]	224 × 224 pixels	96.30	96.4	96.4	96.4
KSM(ours)	224 × 224 pixels	**97.21**	96.7	95.8	96.2

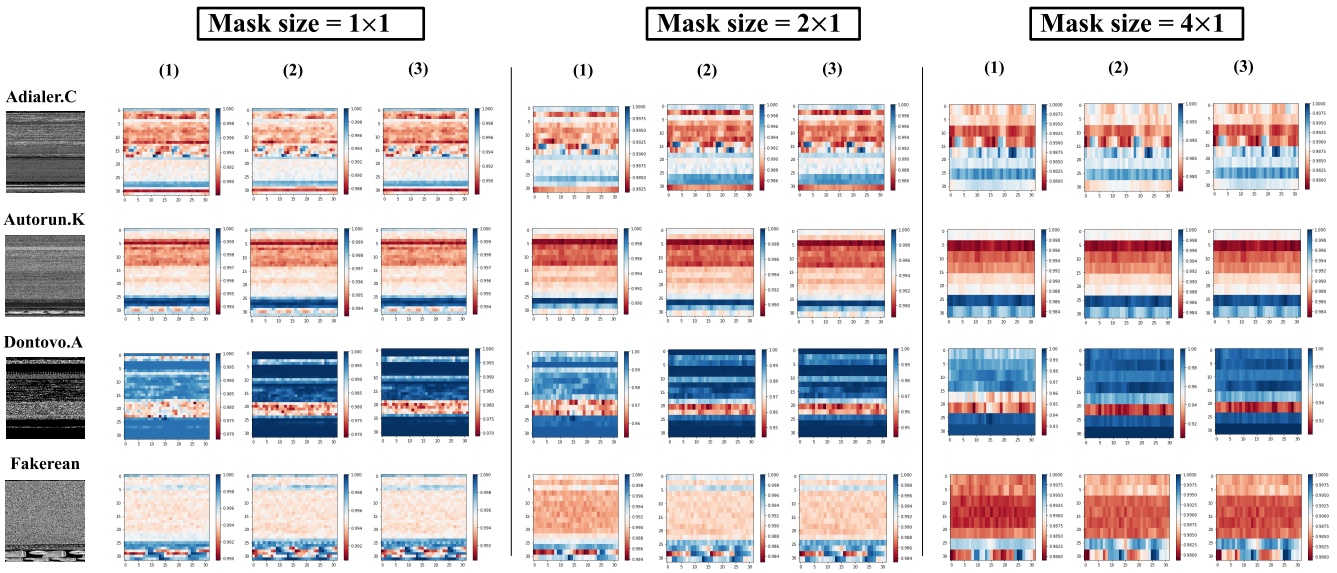


FIGURE 10. Visualization results of OSA_KDS on Malimg dataset with various mask sizes.

F1 score of 97.0. By leveraging the power of nonlinear subspace, the KSM framework excels in capturing intricate structural patterns and features, making it more suitable for detecting a wide range of malware types. Its ability to achieve better performance with a less complex model highlights its efficiency and effectiveness for malware classification.

Results on the Dumpware dataset strengthen our claim to the effectiveness of the proposed method. The simple approach of Nataraj et al. [6] relying only on GIST descriptors and a naive classifier yields an accuracy of 91.40%, and is readily beaten by the competitors. Dai et al. [50] also use simple features, but the stronger classification capabilities offered by the multilayer perceptron increase accuracy by more than 3% to 94.50%, showing that there is space for improvement by using a more sophisticated classifier.

Bozkin et al. [52] achieve an accuracy of 96.30%, making it the best method utilizing classic computer vision features. On the other hand, Rezende [51] rely on neural networks for feature extraction, and obtain the best performance of related methods at 96.60%. However, our proposed approach demonstrates superior performance when compared to these reference studies, outperforming even neural network features. KSM method achieves 97.21% accuracy on the same experimental settings.

D. EXPERIMENT ON MALWARE VISUALIZATION

In this section, we present the visualization results by the OSA_KDS and discuss their effectiveness in revealing and understanding discriminative features of malware files. The dimensions for both malware and benign nonlinear class subspaces were consistently set to six. Figures 10 and 11 show the experimental results, illustrating the visualization outcomes for various malware families from both Malimg and Dumpware datasets, obtained using different mask sizes. We utilized several mask sizes: 1 × 1, 2 × 1, and 4 × 1 pixels, to analyze the influence of occlusion levels on the visualization results. We can see that the OSA_KDS visualization results effectively capture the distinguishing features among different malware families. This can be also confirmed through the observation of correspondence relations in the attention maps of different malware samples belonging to the same class family.

The analysis conducted on the Malimg dataset using OSA visualization revealed insightful findings. For instance, the Adialer.C, Autorun.K and Fakarean malware families

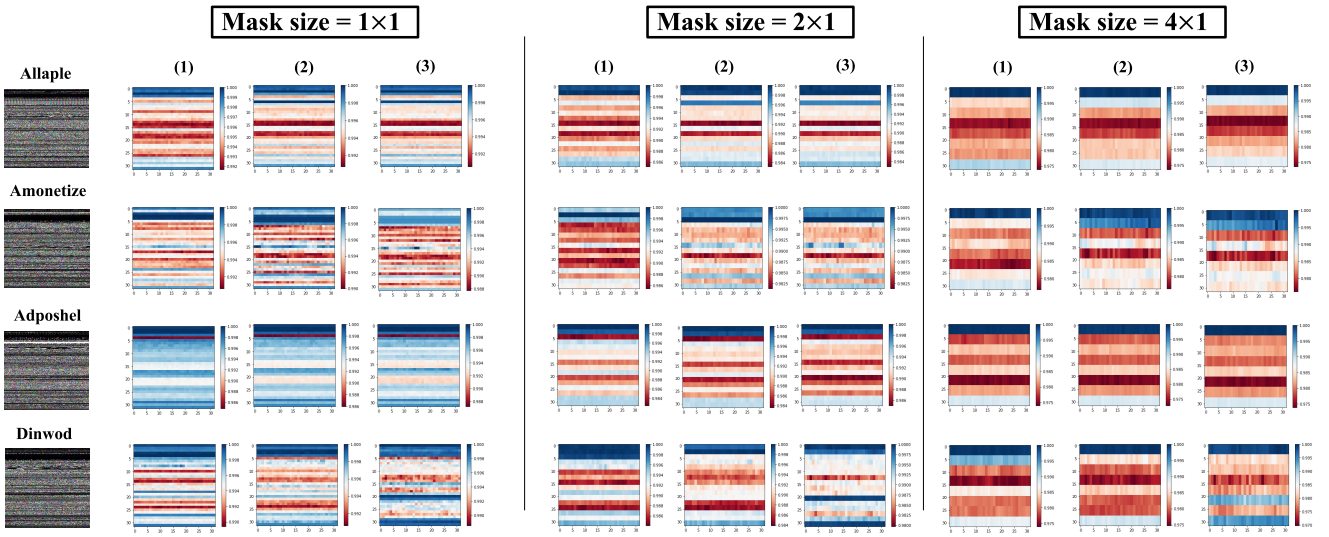


FIGURE 11. Visualization results of OSA_KDS on Dumpware dataset with various mask sizes.

exhibited relatively even distributions of pixels in their attention maps, indicating their tendency to utilize complex application programming interfaces (APIs). On the other hand, Donotovo.A class family displayed more intense pixel distributions and significant disparities in specific regions, implying a greater diversity and potentially more sophisticated malicious behavior. Notably, the visualization method successfully identified distinct important features even within similar malware families. This granularity of feature detection contributes to a more comprehensive understanding of the variations and nuances in the behavior of different malware instances. These visualization results offer the potential to assist experts trace back to the original code by reversing the pre-processing techniques on the discriminant pixels. This enables the identification of problematic PE file sections within the bytecode, enhancing the interpretability of visualized features and aiding in understanding the underlying malicious behavior.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel method for malware detection that combines the power of the kernel subspace method (KSM) with representative pattern images, leading to an efficient and effective solution for malware analysis.

Furthermore, to enhance the interpretability of the detection process, we have employed a visualization approach based on the Occlusion Sensitivity Analysis (OSA) technique in combination with a discriminant kernel difference subspace (KDS). The KDS represents a specialized subspace capturing the discriminant nonlinear components present in the training subclass safe and malware classes. By leveraging the length of the projected input vector onto the KDS, we were able to identify additional discriminant features specific to each malware class family.

The competitive performance of our method, demonstrated through experiments and visualizations, establishes its potential for practical application in real-world malware detection and classification scenarios. The reported performance analysis of our proposed approach KSM, revealed outstanding results in both Malimg and Dumpware datasets. The proposed method achieved an accuracy of 98.07% and 97.21%, respectively, surpassing the performance of existing models cited in the literature.

In future work, we aim to enhance the performance of malware analysis by integrating both bytes and ASM data. Additionally, we also plan to design a hybrid system that combines these data types in a complementary manner, aiming to contribute to the literature on advanced malware analysis techniques. To ensure the generalizability of our proposed model, beyond Malimg and Dumpware datasets used in our experimental evaluation, we would like to expand the data collection to include more types of malware and benign files to conduct further testing. Moreover, we will explore the possibility of the integration of the OSA_KDS visualization approach into automated malware analysis frameworks.

REFERENCES

[1] S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Mach. Learn.*, vol. 46, pp. 351–360, Jan. 2002.

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[3] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, Dec. 2007.

[4] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 103–130, 1997.

[5] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[6] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Visualizat. Cyber Secur.*, Jul. 2011, pp. 1–7.

[7] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.
[8] S. Watanabe and N. Pakvasa, "Subspace method of pattern recognition," in *Proc. 1st IJCPR*, 1973, pp. 25–32.
[9] T. Iijima, H. Genchi, and K.-I. Mori, "A theory of character recognition by pattern matching method," in *Learning Systems and Intelligent Robots*. Boston, MA, USA: Springer, 1974, pp. 437–450.
[10] E. Oja, *Subspace Methods of Pattern Recognition*, vol. 6. Hoboken, NJ, USA: Wiley, 1983.
[11] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, Mar. 2020, Art. no. 102526.
[12] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," *IEEE Access*, vol. 8, pp. 206303–206324, 2020.
[13] Ö. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct./Nov. 2017, pp. 1277–1284.
[14] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel, "Fast malware classification by automated behavioral graph matching," in *Proc. 6th Annu. Workshop Cyber Secur. Inf. Intell. Res.*, Apr. 2010, pp. 1–4.
[15] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," in *Proc. 4th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2009, pp. 23–30.
[16] Y. Cheng, W. Fan, W. Huang, and J. An, "A shellcode detection method based on full native API sequence and support vector machine," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 242, no. 1, 2017, Art. no. 012124.
[17] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious software*. San Francisco, CA, USA: No Starch Press, 2012.
[18] M. Zolotukhin and T. Hamalainen, "Detection of zero-day malware based on the analysis of opcode sequences," in *Proc. IEEE 11th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2014, pp. 386–391.
[19] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013.
[20] Z. Fuyong and Z. Tiezhu, "Malware detection and classification based on N-grams attribute similarity," in *Proc. 7 IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, vol. 1, Jul. 2017.
[21] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole EXE," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–13.
[22] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1357–1365.
[23] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for malware classification," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON) Ohio Innov. Summit (OIS)*, Jul. 2016, pp. 338–342.
[24] Y.-S. Liu, Y.-K. Lai, Z.-H. Wang, and H.-B. Yan, "A new learning approach to malware classification using discriminative feature extraction," *IEEE Access*, vol. 7, pp. 13015–13023, 2019.
[25] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Apr. 2013, pp. 40–44.
[26] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.
[27] J. Fu, J. Xue, Y. Wang, Z. Liu, and C. Shan, "Malware visualization for fine-grained classification," *IEEE Access*, vol. 6, pp. 14510–14523, 2018.
[28] Ö. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.
[29] O. Yamaguchi, E. Fukui, and K. Maeda, "Face recognition using temporal image sequence," in *Proc. 3rd IEEE Int. Conf. Autom. Face Gesture Recognit.*, Apr. 1998, pp. 318–323.
[30] K. Fukui and O. Yamaguchi, "Face recognition using multi-viewpoint patterns for robot vision," in *Proc. 11th Int. Symp. Robot. Res.*, vol. 15. Berlin, Germany: Springer, 2005, pp. 192–201.
[31] K. Fukui and A. Maki, "Difference subspace and its generalization for subspace-based methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2164–2177, Nov. 2015.
[32] Y. Iwashita, H. Sakano, and R. Kurazume, "Gait recognition robust to speed transition using mutual subspace method," in *Proc. 18th Int. Conf. Image Anal. Process. (ICIAP)*. Genoa, Italy: Springer, Sep. 2015, pp. 141–149.
[33] Y. Iwashita, M. Kakeshita, H. Sakano, and R. Kurazume, "Making gait recognition robust to speed changes using mutual subspace method," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017.
[34] K. Fukui, N. Sogi, T. Kobayashi, J.-H. Xue, and A. Maki, "Discriminant feature extraction by generalized difference subspace," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1618–1635, Feb. 2023.
[35] A. Qureshi, M. Marvi, J. A. Shamsi, and A. Aijaz, "eUF: A framework for detecting over-the-air malicious updates in autonomous vehicles," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 8, pp. 5456–5467, Sep. 2022.
[36] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.
[37] K. Griffin, S. Schneider, X. Hu, and T.-C. Chiueh, "Automatic generation of string signatures for malware detection," in *Recent Advances in Intrusion Detection*. vol. 5758. Berlin, Germany: Springer, 2009, pp. 101–120.
[38] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Secur. Privacy. (S&P)*, 2000.
[39] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–42, Feb. 2012.
[40] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2011, pp. 222–230.
[41] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Proactive detection of computer worms using model checking," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 4, pp. 424–438, Oct. 2010.
[42] M. Krcál, O. Švec, M. Bálek, and O. Jašek, "Deep convolutional malware classifiers can learn from raw executables and labels only," in *Proc. ICLR*, 2018, pp. 1–4.
[43] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *J. Comput. Virol. Hacking Techn.*, vol. 15, no. 1, pp. 15–28, Mar. 2019.
[44] K. Zhang, L. Zheng, Z. Liu, and N. Jia, "A deep learning based multitask model for network-wide traffic speed prediction," *Neurocomputing*, vol. 396, pp. 438–450, Jul. 2020.
[45] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *J. Inf. Secur. Appl.*, vol. 47, pp. 377–389, Aug. 2019.
[46] A. F. Agarap, "Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification," 2017, *arXiv:1801.00318*.
[47] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.
[48] J.-S. Luo and D. C.-T. Lo, "Binary malware image classification using machine learning with local binary pattern," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4664–4667.
[49] O. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021.
[50] Y. Dai, H. Li, Y. Qian, and X. Lu, "A malware classification method based on memory dump grayscale image," *Digit. Invest.*, vol. 27, pp. 30–37, Dec. 2018.
[51] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. de Geus, "Malicious software classification using VGG16 deep neural network's bottleneck features," in *Information Technology—New Generations*. Cham, Switzerland: Springer, 2018, pp. 51–59.
[52] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, "Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102166.

[53] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogleNet models for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 15, no. 1, pp. 29–37, Mar. 2019.
[54] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *J. Signal Process. Syst.*, vol. 93, nos. 2–3, pp. 265–273, Mar. 2021.
[55] A. Azab and M. Khasawneh, "MSIC: Malware spectrogram image classification," *IEEE Access*, vol. 8, pp. 102007–102021, 2020.
[56] (2015). *Microsoft Malware Classification Challenge (BIG 2015)*. Dataset. Accessed: Dec. 25, 2022. [Online]. Available: https://www.kaggle.com/c/malware-classification
[57] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, 2007, pp. 1–8.
[58] B. Sriperumbudur and Z. Szabó, "Optimal rates for random Fourier features," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.
[59] A. Singh, A. Handa, N. Kumar, and S. K. Shukla, "Malware classification using image representation," in *Proc. 3rd Int. Symp. Cyber Secur. Cryptogr. Mach. Learn. (CSCML)*, Beer-Sheva, Israel: Springer, 2019, pp. 75–92.
[60] S. A. Roseline, G. Hari, S. Geetha, and R Krishnamurthy, "Vision-based malware detection and classification using lightweight deep learning paradigm," in *Proc. 4th Int. Conf. Comput. Vis. Image Process. (CVIP)*, Jaipur, India: Springer, Sep. 2020, pp. 62–73.
[61] M. J. Awan, O. A. Masood, M. A. Mohammed, A. Yasin, A. M. Zain, R. Damaševičius, and K. H. Abdulkareem, "Image-based malware classification using VGG19 network and spatial convolutional attention," *Electronics*, vol. 10, no. 19, p. 2444, Oct. 2021.
[62] K. Tsuda, "Subspace classifier in the Hilbert space," *Pattern Recognit. Lett.*, vol. 20, no. 5, pp. 513–519, May 1999.
[63] E. Maeda and H. Murase, "Multi-category classification by kernel based nonlinear subspace method," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 2, Mar. 1999, pp. 1025–1028.
[64] E. Maeda and H. Murase, "Kernel-based nonlinear subspace method for pattern recognition," *Syst. Comput. Jpn.*, vol. 33, no. 1, pp. 38–52, Jan. 2002.
[65] Hex-Rays. *IDA Pro*. Accessed Dec. 2014. [Online]. Available: https://www.hex-rays.com/ida-pro/
[66] Kaggle. *Malimg Dataset*. [Online]. Available: https://www.kaggle.com/datasets/manmandes/malimg
[67] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*. Zurich, Switzerland: Springer, Sep. 2014, pp. 818–833.
[68] B. Schölkopf, P. Knirsch, A. Smola, and C. Burges, "Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces," in *Mustererkennung 1998*. Berlin, Germany: Springer, 1998, pp. 125–132.
[69] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," 2018, *arXiv:1802.10135*.

BENCHADI. DJAFER YAHIA M received the B.S. and M.S. degrees in telecommunication, networks, and multimedia from the University of Mohamed Khider Biskra, Algeria, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Computer Vision Laboratory (CVLAB), University of Tsukuba, Japan. His main research interests include computer vision, pattern recognition, and cyber security.

BOJAN BATALO received the M.S. degree in engineering from the University of Novi Sad, in 2019, and the Dr.-Eng. degree from the University of Tsukuba, in 2023. He joined the National Institute of Advanced Industrial Science and Technology (AIST), where he is currently a Postdoctoral Research Scientist with the Machine Learning Research Team, part of the Artificial Intelligence Research Center. His research interests include the theory and application of machine learning to computer vision and pattern recognition problems and the language of science communication.

KAZUHIRO FUKUI (Member, IEEE) received the M.E. degree in mechanical engineering from Kyushu University, in 1988, and the Ph.D. degree from the Tokyo Institute of Technology, in 2003. He joined the Toshiba Corporate Research and Development Center, where he was a Senior Research Scientist with the Multimedia Laboratory. He is currently a Professor with the Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests include the theory of machine learning, computer vision, pattern recognition, and their applications. He is a member of SIAM. He has served as a program committee member for many pattern recognition and computer vision conferences, including as the Area Chair for ICPR'12, ICPR'14, ICPR'16, and ICPR'18.