Malware Detection and Identification using Multi-View Learning based on Sparse Representation

Elham Velayati
Department of Information Technology
Sharif University, Tehran, Iran
d.evelayati@gmail.com

Seyed Mehdi Hazrati Fard
Department of Computer Science and Engineering
Shiraz University, Shiraz, Iran
hazrati@cse.shirazu.ac.ir

Received: 2020/04/09 Revised: 2020/06/06 Accepted: 2020/07/04

***Abstract*— With the widespread using Internet in any device and services, several homes and workplace applications have been provided to avoid attacks. Connecting a system or device to an insecure network can create the possibility of being infected by unwanted files. Detecting such files is a vital task in any system. Employing machine learning (ML) is the most efficient method to detect these penetrations. On the other hand, malware programmers try to design malicious files that are hard to detect. A file can hide from detection in a feature view, but concealing in all views would be very difficult.**

In this paper, inspiring Multi-View Learning (MVL), we proposed to incorporate some various features such as Opcodes, Bytecodes, and System-calls to achieve complementary information to identify a file. In this way, we developed a modified version of Sparse Representation based Classifier (SRC) to aggregate the effect of all modalities in a unified classifier. To show the efficiency of the proposed method, we used several real datasets. Experimental results show the high performance of the proposed approach and its ability to cope with the imbalanced conditions.

Keywords—Multiview Learning, Sparse Representation, Malware Detection, Malware Identification, Imbalanced Condition

1. INTRODUCTION

Malware stands for Malicious Software is a program to perform malicious purposes [1]. They can be divided into categories such as virus, Trojan, worm, rootkit, backdoor, and DoS. However, some files can be classified in more than one category. The easiest way for malware detection is to employ a prepared database from the signature of known malware. Nevertheless, this can not apply to zero-day attacks, and using machine learning (ML) findings can improve the model performance [2].

Multi-View Learning (MVL) is a notable ML approach that combines several distinct attributes of data to improve detection performance [3]. For example, in image processing, color and texture information are two different features, which can boost each other as two-view data. As well, in malware identification, static and dynamic features are two complement feature sets. The static view can be extracted from the files and includes Bytecodes, Opcodes, and format features as three basic views [2]. On the other hand, dynamic features need to run the executable file and observe their behaviors, such as system calls and access to the ports [1].

Malware programmers try to write malicious files in a way that seems like ordinary files. Then, extracting only a single feature view from malware may not be successful in detecting them. By analyzing multiple views, it can be more probable to reveal malfunctions. Because different feature views can provide complementary information about the actual payload of an executable file and lead to a better analysis [4].

It is an underlying assumption in ML that the instances of each class lie on a subspace/submanifold, then each sample can be reconstructed with the neighbor instances [5]. So, each sample can be reconstructed by a linear combination of the other samples in that class. Inspiring that, Wright *et al.* introduced an efficient classifier so-called Sparse Representation based Classifier (SRC) [5]. This idea has received a lot of attention in the realm of face recognition in the last decade [6]–[9].

A malware family members have several resemblances, e.g., using the same APIs for their functionality and frequent access to system resources. As the behavior of malicious files is also similar, it can be assumed that they lie on a subspace/submanifold, and a new file can be locally reconstructed with a linear combination of the files around it. SRC takes the advantages of sparsity; thus, only a few samples participate in reconstructing a new sample, i.e., out-of-sample. As reconstruction in a sparse environment is only based on some available samples, SRC is not sensitive to imbalanced data. Whereas most of the real security databases are imbalanced, SRC can be an excellent choice for them. Furthermore, this method can handle multi-class problems.

In this research inspired by MVL, we proposed a modified version of SRC using various feature sets such as Opcodes, Bytecodes, and System calls, extracted from files for malware detection and identification. The main advantage of this method is that we can consider the importance of all feature views in a unified classifier and decide regarding all modalities. Figure 1 demonstrates the stages of the proposed algorithm in a simple diagram. Evaluating the proposed method shows prominent results on various datasets in the fields of Windows, Linux, and Android files.

The main contributions of this paper are as follows:

- Proposing an efficient framework to combine the results of multi views to detect the actual payload of a malicious file.

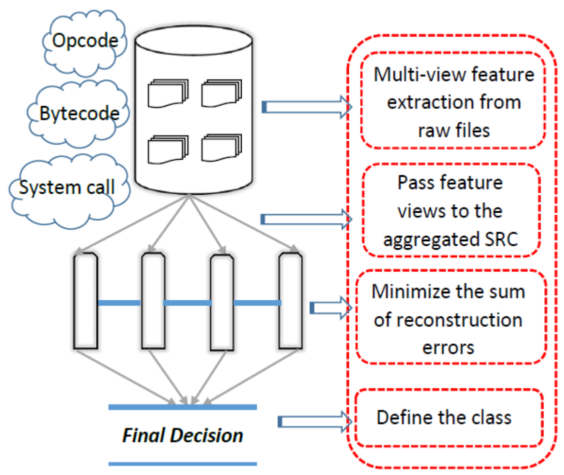
Opcode
Bytecode
System call
Multi-view feature extraction from raw files
Pass feature views to the aggregated SRC
Minimize the sum of reconstruction errors
Define the class
Final Decision

Fig. 1. Schematic view of the proposed method. From the top section, the features are extracted and then passed to the aggregated classifier. The final decision is held according to the minimum reconstruction error of all modalities.

- Modifying SRC, which is not sensitive to imbalanced data and can handle multi-class classification problems.

The rest of this paper is organized as follows: Section 2 reviews the related work on MVL and its applications in several malware detection tasks. As Deep Learning (DL) is also a challenging approach to the current decade, we also introduce some salient methods in this realm for the comparison. In Section 3, the proposed method is introduced in detail. In Section 4, experimental results are presented, and the proposed method is compared with the original SRC using each single feature set and the supervector of all features and the rival methods on some real datasets. Section 5 concludes the paper with a summary of the proposed work and discussions.

2. RELATED WORK

In this section, first, the concept of learning a model using multi-views is explained, and then, some worthy works in the realm of malware detection and identification are investigated.

2-1. Multi-View Learning (MVL)

Considering a problem from various standpoints can lead to a better understanding and has become a prevalent task in the real-world [3]. Recently, many learning methods regarding the diversity of different feature views have been proposed. The motivation of MVL is to solve the problem with data represented by multiple distinct feature sets [10].

The simplest solution for MVL is to concatenate all views into a unified vector, i.e. supervector, and apply a common learning algorithm directly [3]. The drawback of this approach is the curse of dimensionality that often leads to overfitting. In this case, employing some dimensionality reduction algorithms can be useful, but still, the specific statistical property of each view is ignored [3].

In recent literature, MVL methods are divided into three major categories: co-training, co-regularization, and margin consistency [3]. Co-training learners train alternately on distinct views. Co-EM [11], Co-testing [12], and Robust Co-training [13] are representatives of this family.

For Co-regularization algorithms, the disagreement between the discriminant functions of two views is considered as a regularization term in the optimization function. Sparse multi-view SVMs [14], multi-view TSVMs [15], multi-view Laplacian SVMs [16] and multi-view Laplacian TSVMs [17] are some representative examples of this family. Another work proposed by Taheri *et al.* has been used in the security fields to detect ransomware [18].

Margin-consistency algorithms use the latent consistency of classification results from multiple views [19]–[22]. This family is newer than the others and recently has been employed in security tasks. Multiview ensemble learning is the most famous method in this realm. In ensemble learning, multiple models, such as classifiers or experts, are combined to solve the problem. A significant application of ensemble learning is data fusion that improves the confidence of the decision made by the model [23]. Several studies have employed ensemble learning for malware detection tasks [4], [24]–[26].

2-2. Previous Work

Several applicable features can be extracted from files. Opcode n-grams are extracted from the code section of the portable executable (PE) files, which contain significant information [27]–[30]. Bytecode n-grams are the entire binary executable program that has no explicit semantic information, but considering their sequences can be meaningful to use in ML approaches [31]–[33]. Format features are also useful attributes. They contain explicit semantic information extracted from the PE header, section header, import, and resource section. They have been used in many malware detection projects [34]–[37].

Each feature set, as mentioned above, reveals some details about a file, but all together provide complementary information for better diagnosis. Bai and Wang used multiview ensemble learning for unknown malware detection and employed all three mentioned features together in their proposed method [4].

Other useful features that have been used several times in malware detection tasks are function-based. We can extract these features from the behaviors of an executable file that is running in a dedicated system or an isolated environment [1]. Menahem *et al.* employed function-based features besides PE features and bytes n-grams. Then, they used different feature extraction parameters to provide five datasets. They trained five classifiers: C4.5 Decision Tree, Naïve Bayes, KNN, VFI, and OneR based on the datasets above, respectively. Consequently, they used an ensemble of the mentioned classifiers for a final decision. The combination methods include majority voting, performance weighting, distribution summation, Bayesian combination, Naïve Bayes, stacking, and Troika. Experimental results showed that using multi-view features and ensemble methods improved the accuracy and outperformed each of the mentioned single view classifiers. The flaw of them was the high computational burden [25].

It is trivial to extract some features from a feature view, e.g., extracting 2-grams and term frequency-inverse document frequency (TF-IDF) from Opcodes. But, they are the derivations of a single-view and cannot boost each other significantly. Landage *et al.* made three different kinds of Opcode sequence representations of the instances. Then, trained three base classifiers with each representation and combined them with the veto-based and majority voting [38]. The final results did not show a significant improvement from the individual classifiers that can approve the advantages of combining some classifiers based on some distinct views.

API calls are the other feature view of the files that can be extracted while a file is running in a dedicated environment, so it is a dynamic view. Sheen *et al.* extracted features from PE header beside the API calls and ran different learning algorithms to construct a set of classifiers. Then selected the best subset of classifiers and combined them, and reached out to better results than individual classifiers [26]. Caruana *et al.* stated that combining a proper subset of base classifiers to constitute an ensemble may work better than using all of them [24]. They called this method Ensemble Selection (ES), which can achieve strong generalization performance with small-sized base classifiers.

Android malware is another family of malicious files that have been mentioned in past years. Ozdemir *et al.* extracted some different static and dynamic features from APK files and employed multiple learning algorithms to construct diverse base classifiers. Then selected a subset of base classifiers using a simple heuristic algorithm and combined them by majority voting. Experimental results show the superiority of this ensemble over the rival methods [39]. There were other proposed works in this realm that employed MVL in API calls of android executable files besides the other features and improved detection rate [40].

DL has been attracted much attention in recent years and has become a salient trend in ML. So, we introduce some outstanding research in this field to compare with the proposed method. Hashemi *et al.* [41] employed an image classification method that aims to extract micro-patterns of digital textural images, to detect malicious executable files. In this way, they converted executable files to digital images and then extracted visual features such as Local Binary Pattern (LBP). Finally, they used a Convolutional Neural Network (CNN) to distinguish malware and benign files. Their proposed method suffered from low accuracy. So, in another attempt [42], they created a graph of Opcodes within an executable file and then embedded this graph into eigenspace using the Power Iteration method. Consecutively, they represented an executable file as a linear combination of eigenvectors proportionate to their eigenvalues. It was beneficial to train ML classifiers such as KNN and SVM. Although this method has achieved good accuracy, the computational burden made this method hard to run.

Because of the recurring patterns in malware families, Recurrent Neural Network (RNN) has become increasingly popular for cyber-attack detection on different domains. Haddadpajouh *et al.* [43] have explored the potential of using RNN to detect IoT malware. Specifically, they used RNN to analyze ARM-based IoT Opcodes. They evaluated the trained models with three different Long Short Term Memory (LSTM) configurations. The configuration with 2-layer neurons performed better results to detect new malware samples. Homayoun *et al.* [44] identified three ransomware and benign families by combining sequential pattern mining for feature identification in a proposed RNN framework.

Most of the DL-based methods achieved considerable accuracies, while the bottleneck of such approaches is needing numerous samples for training and also high running complexity. So, we seek a low-cost framework with high accuracy to detect the zero-day attacks.

3. PROPOSED METHOD

MVL [3] is an applicable trend in ML, which considers some feature views simultaneously to reach a final decision. MVL combines some feature sets in one or more classifiers to obtain better results than could be reached from any of them [45]. The major strength of such methods depends on the diversity of feature sets. It is also essential to choose a fit base classifier for the problem in hand. Some prevalent methods employ decision trees and neural networks as base classifiers. Most methods use a single base algorithm to produce homogeneous base learners, while some others benefit different classifiers.

SRC is an applicable classifier [5] that has attracted much attention in the past decade. There are several improvements in this method that has been proposed recently [6]–[9]. In this paper, we modified SRC as a sparse-based classifier that is fit to the field of malware detection and identification in a low-cost manner.

The seminal work was proposed by John Wright [5]. The goal is to represent a test example as a linear combination of some selected samples. If there are sufficient training examples from each class, it is feasible to represent the test sample as a linear combination of just those samples from the same category [5]. For reconstructing an example based on other samples, the simple objective is:

$$\|s - Xw\| \qquad (1)$$

where $s \epsilon \mathbb{R}^d$ is a new example, $X \epsilon \mathbb{R}^{d\times n}$ contains the available samples, and $w \epsilon \mathbb{R}^n$ is the coefficient vector. We need to make w sparse enough to choose a few samples for reconstruction. Minimizing the cardinality by adding l_0*-norm* of w, forces many coefficients to be zero, but as the problem is *NP-hard* and intractable in the general case, according to the convex envelope, the constraint can be approximated with l_1*-norm* [46].

$$w^* = \underset{w}{argmin}\|s - Xw\|^2 + \lambda\|w\|_1 \qquad (2)$$

λ is the regularization parameter to specify the sparsity level. It can be determined regarding the size of the dictionary by an expert or using cross-validation [47]. There are several toolboxes, such as NESTA [48] and SPAMS [49] to solve this function in polynomial time, using coordinate descent [50].

For malware identification, suppose X is the matrix of labeled samples, each one is presented with d features in the rows. Consider n samples in the columns, n_1 belong to class 1, n_2 belong to class 2, and so on, like in Figure 2.

The calculated vector w contains n elements pertain to all available samples, n_1 coefficients according to samples of class 1, n_2 coefficients for samples of class 2, and so on. Only a few coefficients are considerable, and the rest of them are negligible. Consequently, the test sample can be reconstructed with the samples of each class separately. As it was a pre-assumption that the samples of each class lie on a subspace/submanifold, the new sample can be reconstructed by a locally linear combination of the samples in the same class. Figure 3 schematically depicts this motivation.

Since each class samples lie on a subspace/ submanifold, the new sample can be reconstructed through a linear combination of its neighbors. So, the class with the minimum reconstruction error can classify the test sample [5].

$$j^* = arg \min_{j\in\{1,2,\dots,J\}} \|s - X_j w_j^*\| \qquad (3)$$

j shows the label of each class through J classes, and j^* is the label of the class with the minimum reconstruction error. Equations 2 and 3 show the seminal method of SRC [5].

Inspired by the idea behind MVL, we modified SRC to combine the effect of several views. In the proposed framework, the new file is reconstructed with a linear combination of available samples. The objective is to minimize the reconstruction error according to all feature views. Considering K views, the best coefficient vector w^* is the one that minimizes the sum of errors according to all views. The modified function can be rewritten as Equation (4):

$$w^* = \underset{w}{argmin} \sum_{k=1}^{K} \|s^k - X^k w^k\|^2 + \lambda \|w^k\|_1 \qquad (4)$$

As the number of features in all views are not the same, it is necessary to exert a normalization on each reconstruction error term and divide each one to the number of features in that view:

$$w^* = \underset{w}{argmin} \sum_{k=1}^{K} \frac{\|s^k - X^k w^k\|^2}{d^k} + \lambda \|w^k\|_1 \qquad (5)$$

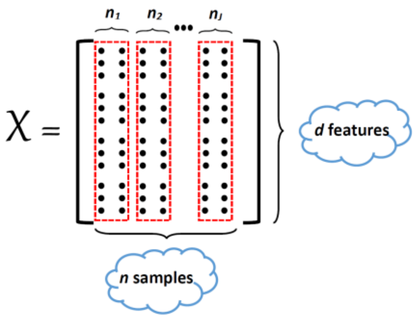

Fig. 2. The simple structure of a database for reconstructing an out-of-sample.

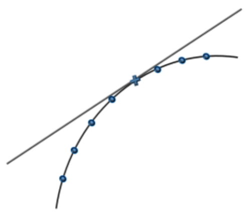

Fig. 3. The spanned space of the samples of a class can be considered locally linear, and each sample can be reconstructed with the linear combination of its adjacent on this path.

where d^k is the dimension of the k^{th} view and w^* is the best weight vector, including the coefficients of training samples according to all views. Finally, the class with the minimum sum of reconstruction errors regarding all feature sets, determine the label of the test sample:

$$j^* = arg \min_{j\in\{1,2,\dots,J\}} \sum_{k=1}^{K} \frac{\|s^k - X_j^k w_j^{k*}\|}{d^k} \qquad (6)$$

As the aim of this optimization function is to minimize the sum of reconstruction errors, all views participate in finding w^* as a shared vector. So, the effect of all feature sets can lead to finding the best samples for describing the out-of-sample. Algorithm 1 shows the stages of the proposed model to label an unknown file.

In the first step of Algorithm 1, we need to extract some useful features from each file. As the platform of any file can be different, regarding the ability of the host devices, we extract some relevant features. For example, extracting Opcodes and Bytecodes for Android files is more feasible than controlling their behavior. So, we can extract 2-grams and TF-IDF from Opcodes and Bytecodes to use in the algorithm. TF-IDF considers the repetition of each Opcode in a file individually vs. the presence of this Opcode in the other files, while 2-grams considers the importance of Opcode sequences. On the other hand, for Windows files, we can use a Sandbox to extract system calls as a dynamic feature. This can be a useful complementary feature set for Opcodes that has been discussed in detail in the Experimental Results Section.

In the next steps, a unified SRC decides based on the consensus of all views and finds the class with the minimum overall reconstruction error regarding all features. In conclusion, this algorithm aggregates the effects of several views in a unified objective function that leads to detect a file's payload.

Another advantage of the proposed method besides the accuracy is the insensitivity to imbalanced datasets. To reconstruct a new sample, we use a linear combination of available instances. So, it can be considered a locally linear reconstruction, and the rest of the samples do not participate in reconstruction, like in Figure 3. So, it is not dependent on all samples of a class and only needs the samples that lie on the subspace/submanifold near the test sample. To prove these assertions, we used several real datasets in the experiments.

```
Algorithm 1
Input: a raw file
output: the selected class
1.  Extract features of several views from all files.
2.  Find the best coefficient vector of $w^*$ that leads to a minimum reconstruction error based on all features:
        $w^* = \underset{w}{argmin} \sum_{k=1}^{K} \frac{\|s^k - X^k w^k\|^2}{d^k} + \lambda \|w^k\|_1$
3.  Choose the class which best describes the out of sample through all views:
        $j^* = arg \min_{j\in\{1,2,\dots,J\}} \sum_{k=1}^{K} \frac{\|s^k - X_j^k w_j^{k*}\|}{d^k}$
```

4. Experimental results

In this Section, several benchmarks and collected datasets on various platforms, e.g., Android and Windows, are investigated. Then, the effectiveness of the feature combination in the proposed method is illustrated and compared with the based method using the individual feature sets and some other rival methods.

The used datasets and their specifications are introduced briefly in Table 1. The first column shows the name of datasets, and the second one is according to the environments of the files. The third and fourth columns are due to the number of samples and classes, respectively. Also, other columns show the extracted feature sets from each dataset.

Whereas the proposed method does not need many samples from each class for reconstruction, first, some samples from each category are selected, and then, the required features are extracted. If we select 100 samples for each class, the computational burden of running the algorithm is less than a second. So, we exerted leave-one-out in each case to show the superiority of the proposed model.

The results of the proposed method are compared with SRC using each extracted feature set and the concatenation of all feature sets. Employing a supervector of all features was the seminal idea of MVL that was used in the previous work. We implemented the algorithm in Matlab 2016b and used a personal desktop equipped with a Core i7-3770 CPU and 32GB of memory for its running.

4-1. Evaluation Metrics

For evaluation, some metrics are required. Accuracy is a useful metric that has been used widely in the ML assessments and considers the rate of correct predictions to all. True Positive (TP) and True Negative (TN) are the malicious and benign files that have been classified truly, respectively. Also, False Negative (FN) and False Positive (FP) are the malicious and benign files that have been classified wrongly, respectively. So, accuracy will be the number of true classified samples out of all test samples that have been shown in Equation (7):

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (7)$$

True Positive Rate (TPR) is the percentage of actual positives that are correctly identified. So, it is the rate of detected malicious files out of all malware.

$$\text{TPR} = \frac{TP}{TP+FN} \qquad (8)$$

Also, False Positive Rate (FPR) is the rate of incorrectly rejected files, which is the benign files that are identified as malware out of all malicious files.

$$\text{FPR} = \frac{FP}{FP+TN} \qquad (9)$$

4-2. Internet of Things (IoT) Dataset

Nowadays, IoT devices become more and more prevalent, and consequently, many malware developers target IoT devices [51]. Thus, one of the selected datasets to challenge our method includes IoT files. In this dataset, 280 malware samples were collected from 32-bit ARM-based malware in the Virus Total Threat Intelligence platform. For compatibility of the malware and benign files, 271 standard files of the Linux Debian package repository were chosen [43]. The files were unpacked by the Debian installer bundle, and the Object-Dump tool was employed to decompile all samples.

The sequence of Opcodes in each sample was obtained from all files, and consequently, two feature sets were extracted from them: TF-IDF and 2-grams. Bytecode is another applicable view of the files. Then, two sets of features, according to Bytecode TF-IDF and 2-grams, were extracted as well. Eventually, we had four feature sets to learn classifiers.

For the first evaluation, 100 samples were selected randomly from each class, and then leave-one-out was exerted to reconstruct each sample with the rest of 199. Table 2 compares the results of the proposed method to the original SRC according to each set of features individually and using a supervector of all features. The results are due to the average of a 10-times random-selection of samples from the initial dataset. The first row of the table is, according to Haddadpajouh *et al.* [43], due to this dataset.

The accuracy and FPR of the proposed method outperform all other methods in Table 2, while the TPR is somehow near to Haddadpajouh *et al.* [43] and SRC using supervector of all extracted features.

To investigate the imbalanced effect on the proposed methods, we selected a subset of 100 benign and 30 of malware samples. Table 3 shows the results according to 10-times random-selection in the mentioned methods that show no significant changes compared to the balanced condition. This can prove the ability of the proposed method to handle

TABLE 1. THE INTRODUCED DATASETS AND THEIR SPECIFICATIONS. TO EVALUATE THE METHODS, TWO BINARY CLASS DATASETS INCLUDE MALWARE AND BENIGN FILES AND TWO MULTI-CLASS DATABASES CONSIST OF SOME MALWARE TYPES ARE EMPLOYED.

dataset	***environment***	***# samples***	***# classes***	***feature set 1***	***feature set 2***	***feature set 3***	***feature set 4***
IoT dataset	Linux	551	2	opcodes TF-IDF	opcode 2-grams	bytecode TF-IDF	bytecode 2-grams
VXHeaven	Windows	330	2	opcodes TF-IDF	opcode 2-grams	bytecode TF-IDF	bytecode 2-grams
Ransomware dataset	Windows	1627	4	System call	opcodes TF-IDF	opcode 2-grams	
Microsoft malware	Windows	10825	9	opcodes TF-IDF	opcode 2-grams	opcode 3-grams	

imbalanced conditions. To show the merit of the proposed method, our results are compared to the original SRC using Opcode TF-IDF as the best single feature set in Table 2 and also using supervector of all features, which is a type of MVL in the literature.

4-3. VXHeaven Dataset

VXHeaven is a benchmark dataset contains windows malware [52] and was used to evaluate several methods [41], [42], [53]. 1000 samples were picked randomly from the dataset and labeled as malware and benign. Then, TF-IDF and 2-grams, according to Opcodes and Bytecodes of the selected files, were extracted as four feature sets.

The results are presented in Table 4. The mentioned algorithms were applied on samples 10-times; each time, 100 samples were picked randomly from each class. Leave-one-out was exerted, while the runtime for each sample was less than a second. The first three rows of the table are due to some available methods that have evaluated this dataset, recently.

As the available samples in this dataset are due to the Windows environment, the files are more complicated, and the overall accuracies are a bit lower than the IoT samples. Nevertheless, the accuracy and TPR of the proposed method outperform the rivals. Also, the FPR of Hashemi et al. [42] is comparable to the proposed method.

As we faced to a binary class dataset, we examined imbalanced conditions again. To do so, a subset of 100 benign and 30 malware samples are selected randomly, 10-times. Table 5 implies the ability of the proposed methods to challenge the imbalanced conditions.

The results of the proposed method have not diminished meaningfully compared to the balanced condition. This is due to the ability of the proposed method to handle imbalanced conditions. To show the superiority of the proposed method, our results are compared to the original SRC using supervector of all features and also Bytecode 2-grams as the best single feature set in Table 4.

4-4. Ransomware Dataset

As another evaluation, the ransomware dataset has been employed [44]. The ransomware dataset contains sequences of activities according to some Windows PE ransomware samples reported as malicious, from Virustotal[1]. This dataset consists of three famous families of ransomware, namely Locky, Cerber, and TeslaCrypt. As ransomware samples are in the form of PE files, portable applications available online[2] are considered as benign samples. Table 6 shows the summary report of the samples in the dataset with three families of ransomware and a group of benign samples.

All samples were launched in a testbed to collect runtime behaviors of ransomware and normal files. The runtime behaviors were considered as system calls performed by the process of a monitored sample that leads to the first view of the samples. Moreover, TF-IDF and 2-grams were extracted from Opcodes.

TABLE 2 .COMPARING THE RESULTS OF THE PROPOSED METHOD ON IOT DATASET WITH SRC (BASED ON FOUR FEATURE SETS AND THE CONCATENATION OF ALL) AND HADDADPAJOUH *ET AL.* [55].

IoT (100/100)	*TPR (%)*	*FPR (%)*	*Accuracy (%)*
Haddadpajouh *et al.* [43]	98.6	2.1	98.1
SRC Opcode TF-IDF	96.1	3.2	95.3
SRC Opcode 2-grams	92.5	4.4	94.4
SRC Bytecode TF-IDF	91.3	5.2	92.6
SRC Bytecode 2-grams	93.1	3.8	90.3
SRC Supervector	97.9	2.8	96.7
Proposed Method	98.8	1.1	99.1

TABLE 3. THE AVERAGES RESULT FROM THE PROPOSED METHOD AND SRC USING THE BEST FEATURE SET AND SUPERVECTOR OF ALL EXTRACTED FEATURES ON THE IMBALANCED DATASET OF IOT FILES.

IoT (100/30)	*TPR (%)*	*FPR (%)*	*Accuracy (%)*
SRC Opcode TF-IDF	93.3	4.2	94.0
SRC Supervector	95.6	3.6	95.1
Proposed Method	96.9	1.9	97.1

TABLE 4. THE RESULTS OF THE PROPOSED METHOD ON VXHEAVEN SAMPLES COMPARED TO SRC (BASED ON FOUR FEATURE SETS AND THE CONCATENATION OF ALL) AND SOME RECENT OUTSTANDING METHODS.

VXHeaven (100/100)	*TPR (%)*	*FPR (%)*	*Accuracy (%)*
Farrokhmanesh *et al.* [53]	91.3	7.9	90.4
Hashemi *et al.* [41]	89.1	7.8	89.7
Hashemi *et al.* [42]	96.0	3.1	94.7
SRC Opcode TF-IDF	86.6	12.8	84.5
SRC Opcode 2-grams	85.5	14.4	87.4
SRC Bytecode TF-IDF	88.3	7.6	89.6
SRC Bytecode 2-grams	90.3	5.2	90.1
SRC Supervector	92.9	5.5	93.6
Proposed Method	96.6	3.1	96.3

TABLE 5. THE AVERAGES OF RESULTS ON THE IMBALANCED DATASET COMPARING THE PROPOSED METHOD AND SRC USING SUPERVECTOR OF ALL FEATURES AND THE FEATURE SET IN TABLE 4.

VXHeaven (100/30)	*TPR (%)*	*FPR (%)*	*Accuracy (%)*
SRC Bytecode 2-grams	89.4	6.5	89.6
SRC Supervector	92.1	5.9	93.0
Proposed Method	97.2	2.6	97.5

TABLE 6. THE NUMBER OF SAMPLES IN EACH FAMILY OF RANSOMWARE AND BENIGN.

Class	*number of samples*
Locky	450
Cerber	470
TeslaCrypt	507
Benign	200

[1] https://www.virustotal.com/
[2] https://portableapps.com/app

As few samples were required for the learning, and a multi-class classification problem was ahead, we selected 50 samples of each class randomly and exerted leave-one-out for assessment. These steps were repeated 10-times, and the results were summed up in Table 7. The first row of the table is the results of Homayoun *et al.* [44] that were reported in their paper according to the prepared database.

According to Table 7, the results of the proposed model, works better than all single classifiers, while the TPR of Homayoun *et al.* [44] is near to the proposed method.

Similar to the previous datasets, to show the eligibility of the proposed method in imbalanced conditions, we aggregated the samples of 3 malware families as a malicious class, including 150 samples vs. 50 benign instances on the other hand. Table 8 indicates the summary of the average results achieved on the imbalanced data based on 10-times random subsampling and leave-one-out. In this table, only the results of the best base classifier (Opcode 2-grams) are reported and compared to the results of the SRC supervector and the proposed approach.

It is evident from Table 8 that the results of the proposed method in imbalanced conditions are comparable to the results of Table 7.

4-5. Microsoft Malware Dataset

Another Windows-based malicious dataset that has been used in our evaluation is Microsoft malware collection that was presented in the Microsoft malware classification challenge from the Kaggle website [54]. This dataset contains more than 10000 samples from 9 families of malware variants that have been analyzed statically to obtain their Opcodes. Consequently, TF-IDF, 2-grams, and 3-grams of the Opcodes from 900 files of all classes were extracted.

We selected 40 samples of each class randomly 10-times and repeat our experiments for each subset. Table IX shows the average accuracies according to 10 runs. Whereas all three feature sets are extracted from Opcodes, in this case, the results of the ensemble methods are not significantly superior.

The available results in Table 9 can confirm the assumption of diversity according to the employed feature sets to build an MVL method. Also, one of the state-of-the-art in this field [38] had shown that the dependent feature sets could not boost each other in the ensemble and MVL methods.

We can reach better results by adding some complement feature sets to the available data to raise the results of the MVL-based methods. Diversity has a significant role in this area. So, it shows the importance of using partly independent views in the completion of each other and reveals the hidden virtue of the samples.

The runtime of the proposed method is proportional to the size of the selected samples for reconstruction. So, in this dataset with growing the number of samples, the runtime grows highly. For this reason, we used a random selection method in previous datasets to reach reasonable running time and used the method for online applications.

5. CONCLUSION AND FUTURE WORK

In this study, inspired by Multi-View Learning (MVL) and Sparse Representation based Classifier (SRC), we proposed a model for malware identification and classification. Whereas using various views of the files, e.g., Opcodes, Bytecodes, and System calls, help the classifiers to reveal the hidden dimensions of a file, we combined the reconstruction errors in a unified SRC. As the proposed method uses the sparse representation to reconstruct an out-of-sample, it can handle imbalanced conditions. Another merit of the proposed model is the ability to overcome multi-class problems without any extra computation.

The proposed methods outperform any individual based classifiers trained on a single feature set and show elegant results on several datasets that have been investigated in the experimental results. Also, we tested the imbalanced conditions for the available datasets and considered about three times more samples from a class.

As future work, we suggest learning the combination phase of the algorithms intelligently. To do so, we can learn each classifier individually and then learn a model for the best combination of them. Another suggestion is to extend these methods to the other ML tasks. Various approaches in the real world suffer from the nature of imbalanced data and can take advantage of the proposed method.

TABLE 7. COMPARING THE RESULTS OF THE PROPOSED METHOD ON RANSOMWARE WITH SRC (BASED ON THREE FEATURE SETS AND THE CONCATENATION OF ALL) AND HOMAYOUN *ET AL.* [44].

Ransomware (4 × 50)	***TPR (%)***	***FPR (%)***	***Accuracy (%)***
Homayoun *et al.* [44]	98.0	2.6	97.2
SRC System calls	86.3	12.1	85.2
SRC Opcode TF-IDF	89.5	9.7	92.9
SRC Opcode TF-IDF	91.5	7.0	90.3
SRC Supervector	96.2	4.9	94.9
Proposed Method	98.6	1.3	98.7

TABLE 8 .THE AVERAGES RESULT FROM THE PROPOSED METHOD AND SRC USING THE BEST FEATURE SET AND SUPERVECTOR OF ALL EXTRACTED FEATURES ON THE IMBALANCED DATASET.

Ransomware (150/50)	***TPR (%)***	***FPR (%)***	***Accuracy (%)***
SRC Opcode 2-grams	91.0	6.8	90.1
SRC Supervector	93.8	4.3	94.4
Proposed Method	95.9	2.2	97.0

TABLE 9. COMPARING THE RESULTS OF THE PROPOSED METHOD ON MICROSOFT MALWARE DATASET WITH SRC (BASED ON THREE FEATURE SETS AND THE CONCATENATION OF ALL).

Microsoft malware (9 × 100)	***TPR (%)***	***FPR (%)***	***Accuracy (%)***
SRC Opcode TF-IDF	91.7	8.8	92.9
SRC Opcode 2-grams	92.8	4.6	92.3
SRC Opcode 3-grams	94.7	5.6	93.4
SRC Supervector	93.2	6.1	94.5
Proposed Method	94.8	5.7	94.3

REFERENCES

[1] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *IKT - 5th Conference on Information and Knowledge Technology*, 2013, pp. 113–120.

[2] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, "Machine learning aided static malware analysis: A survey and tutorial," *Cyber Threat Intell.*, pp. 7–45, 2018.

[3] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Inf. Fusion*, vol. 38, pp. 43–54, 2017.

[4] J. Bai and J. Wang, "Improving malware detection using multiview ensemble learning," *Secur. Commun. Networks*, vol. 9, no. 17, pp. 4227–4241, 2016.

[5] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[6] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma, "Toward a practical face recognition system: Robust alignment and illumination by sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 372–386, 2012.

[7] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma, "Fast ℓ 1-minimization algorithms and an application in robust face recognition: A review," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010, pp. 1849–1852.

[8] J. Ma, J. Zhao, Y. Ma, and J. Tian, "Non-rigid visible and infrared face registration via regularized Gaussian fields criterion," *Pattern Recognit.*, vol. 48, no. 3, pp. 772–784, 2015.

[9] Y. Gao, J. Ma, and A. L. Yuille, "Semi-supervised sparse representation based classification for face recognition with insufficient labeled samples," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2545–2560, 2017.

[10] S. Sun, "A survey of multiview machine learning," *Neural Comput. Appl.*, vol. 23, no. 7–8, pp. 2031–2038, 2013.

[11] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 86–93.

[12] I. Muslea, S. Minton, and C. A. Knoblock, "Active learning with multiple views," *J. Artif. Intell. Res.*, vol. 27, pp. 203–233, 2006.

[13] S. Sun and F. Jin, "Robust co-training," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 07, pp. 1113–1126, 2011.

[14] S. Sun and J. Shawe-Taylor, "Sparse semi-supervised learning using conjugate functions," *J. Mach. Learn. Res.*, vol. 11, no. Sep, pp. 2423–2455, 2010.

[15] X. Xie and S. Sun, "Multi-view twin support vector machines," *Intell. Data Anal.*, vol. 19, no. 4, pp. 701–712, 2015.

[16] S. Sun, "Multi-view Laplacian support vector machines," in *International Conference on Advanced Data Mining and Applications*, 2011, pp. 209–222.

[17] X. Xie and S. Sun, "Multi-view Laplacian twin support vector machines," *Appl. Intell.*, vol. 41, no. 4, pp. 1059–1068, 2014.

[18] M. Taheri, H. Azad, K. Ziarati, and R. Sanaye, "A QUADRATIC MARGIN-BASED MODEL FOR WEIGHTING FUZZY CLASSIFICATION RULES INSPIRED BY SUPPORT VECTOR MACHINES," *Iran. J. Fuzzy Syst.*, vol. 10, no. 4, pp. 41–55, 2013.

[19] G. Chao and S. Sun, "Alternative multiview maximum entropy discrimination," *IEEE Trans. neural networks Learn. Syst.*, vol. 27, no. 7, pp. 1445–1456, 2016.

[20] G. Chao and S. Sun, "Consensus and complementarity based maximum entropy discrimination for multiview classification," *Inf. Sci. (Ny).*, vol. 367, pp. 296–310, 2016.

[21] L. Mao and S. Sun, "Soft Margin Consistency Based Scalable Multi-View Maximum Entropy Discrimination.," in *IJCAI*, 2016, pp. 1839–1845.

[22] S. Sun and G. Chao, "Multi-View Maximum Entropy Discrimination.," in *IJCAI*, 2013, pp. 1706–1712.

[23] R. Polikar, "{E}nsemble learning," *Scholarpedia*, vol. 4, no. 1, p. 2776, 2009.

[24] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 18.

[25] E. Menahem, A. Shabtai, L. Rokach, and Y. Elovici, "Improving malware detection by applying multi-inducer ensemble," *Comput. Stat. Data Anal.*, vol. 53, no. 4, pp. 1483–1494, 2009.

[26] S. Sheen, R. Anitha, and P. Sirisha, "Malware detection by pruning of parallel ensembles using harmony search," *Pattern Recognit. Lett.*, vol. 34, no. 14, pp. 1679–1686, 2013.

[27] D. Bilar, "Opcodes as predictor for malware," *Int. J. Electron. Secur. Digit. Forensics*, vol. 1, no. 2, pp. 156–168, 2007.

[28] R. Moskovitch *et al.*, "Unknown malcode detection using opcode representation," in *Intelligence and Security Informatics*, Springer, 2008, pp. 204–215.

[29] 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. (Ny).*, vol. 231, pp. 64–82, 2013.

[30] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Secur. Inform.*, vol. 1, no. 1, p. 1, 2012.

[31] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *J. Mach. Learn. Res.*, vol. 7, no. Dec, pp. 2721–2744, 2006.

[32] D. K. S. Reddy and A. K. Pujari, "N-gram analysis for computer virus detection," *J. Comput. Virol.*, vol. 2, no. 3, pp. 231–239, 2006.

[33] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based File Signatures for Malware Detection.," *ICEIS (2)*, vol. 9, pp. 317–320, 2009.

[34] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," *Sci. World J.*, vol. 2014, 2014.

[35] M. Z. Shafiq, S. Tabish, and M. Farooq, "PE-probe: leveraging packer detection and structural information to detect malicious portable executables," in *Proceedings of the Virus Bulletin Conference (VB)*, 2009, vol. 8.

[36] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "Pe-miner: Mining structural information to detect malicious executables in realtime," in *International Workshop on Recent Advances in Intrusion Detection*, 2009, pp. 121–141.

[37] M. Zakeri, F. Faraji Daneshgar, and M. Abbaspour, "A static heuristic approach to detecting malware targets," *Secur. Commun. Networks*, vol. 8, no. 17, pp. 3015–3027, 2015.

[38] J. Landage and M. P. Wankhade, "Malware detection with different voting schemes," *Compusoft*, vol. 3, no. 1, p. 450, 2014.

[39] M. Ozdemir and I. Sogukpinar, "An android malware detection architecture based on ensemble learning," *Trans. Mach. Learn. Artif. Intell.*, vol. 2, no. 3, pp. 90–106, 2014.

[40] S. Sheen, R. Anitha, and V. Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach," *Neurocomputing*, vol. 151, pp. 905–912, 2015.

[41] H. Hashemi and A. Hamzeh, "Visual malware detection using local malicious pattern," *J. Comput. Virol. Hacking Tech.*, pp. 1–14, 2018.

[42] H. Hashemi, A. Azmoodeh, A. Hamzeh, and S. Hashemi, "Graph embedding as a new approach for unknown malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 13, no. 3, pp. 153–166, 2017.

[43] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," *Futur. Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018.

[44] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Trans. Emerg. Top. Comput.*, 2017.

[45] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010.

[46] C. Ramirez, V. Kreinovich, and M. Argaez, "Why l1 is a good approximation to l0: A geometric explanation," *J. Uncertain Syst.*, vol. 7, no. 3, pp. 203–207, 2013.

[47] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1, no. 10. Springer series in statistics New York, NY, USA:, 2001.

[48] S. Becker, J. Bobin, and E. J. Candès, “NESTA: a fast and accurate first-order method for sparse recovery,” *SIAM J. Imaging Sci.*, vol. 4, no. 1, pp. 1–39, 2011.

[49] J. Mairal, “SPAMS: a SPArse Modeling Software, v2. 5.” 2014.

[50] S. J. Wright, “Coordinate descent algorithms,” *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.

[51] H. Naeem, B. Guo, and M. R. Naeem, “A light-weight malware static visual analysis for IoT infrastructure,” in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018.

[52] V. X. Heaven, “Computer virus collection,” *2007-09-14)[2010-05-28]. http://vx. netlux. org/vl. php*. 2014.

[53] M. Farrokhmanesh and A. Hamzeh, “A novel method for malware detection using audio signal processing techniques,” in *Artificial Intelligence and Robotics (IRANOPEN), 2016*, 2016, pp. 85–91.

[54] Microsoft, “Microsoft malware classification challenge,” *[Online]*, Available: https://www.kaggle.com/c/malware-classification., 2015.

Elham Velayati has received her M.Sc. in Information Technology engineering from Sharif University of Technology, Iran in 2012. Her research area is about Internet of things, Internet Security, Wireless Sensor Networks and its application.

Seyed Mehdi Hazrati Fard has received the Ph.D. degree in computer science from the Computer department at Electrical and Computer Engineering School, Shiraz University, Shiraz, Iran, in 2019. Since 2012, he has been an active member of Machine Learning Lab (MLL) and has done several projects on deep networks and sparse representation. Furthermore, he has been at the University of Waterloo as an international visitor in 2017. Currently, he is the assistant professor at Pishtazan Higher Educational Institute.