GRASE: Granulometry Analysis With Semi Eager Classifier to Detect Malware

Mahendra Deore[1]*, Manoj Tarambale[2], Jambi Ratna Raja Kumar[3], Sachin Sakhare[4]

[1] Department of Computer Engineering, MKSSS's Cummins College of Engineering for Women, Pune-411052 (India)
[2] Electrical Engineering Department, PVG's COET and GKP IOM, Pune- 411009 (India)
[3] Department of Computer Engineering, Genba Sopanrao Moze College of Engineering, Pune-411045 (India)
[4] Department of Computer Engineering, Vishwakarma Institute of Information Technology, Pune 411048 (India)

Received 26 February 2023 | Accepted 13 November 2023 | Early Access 12 December 2023

Abstract

Technological advancement in communication leading to 5G, motivates everyone to get connected to the internet including 'Devices', a technology named Web of Things (WoT). The community benefits from this large-scale network which allows monitoring and controlling of physical devices. But many times, it costs the security as MALicious softWARE (MalWare) developers try to invade the network, as for them, these devices are like a 'backdoor' providing them easy 'entry'. To stop invaders from entering the network, identifying malware and its variants is of great significance for cyberspace. Traditional methods of malware detection like static and dynamic ones, detect the malware but lack against new techniques used by malware developers like obfuscation, polymorphism and encryption. A machine learning approach to detect malware, where the classifier is trained with handcrafted features, is not potent against these techniques and asks for efforts to put in for the feature engineering. The paper proposes a malware classification using a visualization methodology wherein the disassembled malware code is transformed into grey images. It presents the efficacy of Granulometry texture analysis technique for improving malware classification. Furthermore, a Semi Eager (SemiE) classifier, which is a combination of eager learning and lazy learning technique, is used to get robust classification of malware families. The outcome of the experiment is promising since the proposed technique requires less training time to learn the semantics of higher-level malicious behaviours. Identifying the malware (testing phase) is also done faster. A benchmark database like malimg and Microsoft Malware Classification challenge (BIG-2015) has been utilized to analyse the performance of the system. An overall average classification accuracy of 99.03 and 99.11% is achieved, respectively.

Keywords

Malware, Semi Eager Classification (Semi-E), Granulometry Analysis, Static and Dynamic Analysis.

DOI: 10.9781/ijimai.2023.12.002

I. Introduction

Malicious software is baleful to all the devices connected to an internet, irrespective of the platform i.e., windows on laptop or android in a mobile. Presently android applications are growing exponentially to the scale of approximately 5 million apps in Google play as of May 2021 surpassing 2.99 million in the year 2020[1]. In parallel, malicious apps are also increasingly creating threats to mobile based financial transactions, taking control over mobile cameras, and misusing the same. According to the survey done by AV-TEST institute, there are approximately 1214.76 million malicious apps in the year 2021. Everyday AV-TEST registers approximately 350,000 new malicious apps and potentially unwanted applications[2]. To cope with the security threats various techniques for malware detection have been proposed by the researchers. It has been found that Machine Learning (ML) based detection technique is one of the efficient methods to opt for Malware Detection System (MDS). ML based MDS is comprehensive, detects malware accurately and less dependency on human experts which is normally required in traditional MD techniques. Thus, ML techniques are found to be more suitable for present scenarios where malicious software is increasing day by day.

Traditionally, the ML technique is feature vector based in which important characteristics of malware are extricated and used for identifying the same in a real time system. Static and dynamic are the two primary feature sources which describe malware characteristics. Both the analysis techniques can be applied to various kinds of executable files like PE, ELF, DEX etc., of different processors and operating systems (OS) such as Microsoft Windows, Linux, Android, etc.

The Static Analysis (SA) of malware software is done without the code being executed [1]. In SA, features are extracted after unpacking the executable in advance. Examples of static features are, OPCODE (Operation CODE) frequency distribution, control or data flow graph, syntactic library call, byte-sequence n-grams, string signature etc.

[1] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] https://www.av-test.org/en/statistics/malware/

* Corresponding author.
E-mail address: mdeore83@gmail.com

Please cite this article in press as:
M. Deore, M. Tarambale, J. R. R. Kumar, S. Sakhare. GRASE: Granulometry Analysis With Semi Eager Classifier to Detect Malware, International Journal of Interactive Multimedia and Artificial Intelligence, (2023), http://dx.doi.org/10.9781/ijimai.2023.12.002

Static features can be extracted from an image, sound, native code, and byte code.

In Dynamic Analysis (DA) run time behavior of malware executable is monitored. Due to this code is executed by using controlled environments like sandbox, emulator, simulator, virtual machine, etc. Dynamic features are extracted from sensitive function calls, variable value tracking, code execution path, log records and other behavior of the code when the same is being executed. Normally, analysis of the code is done by different tools like Process monitor (.pmon), Capture Bat, poison IVY, etc. The report generated from the tool is extensive and in depth, requiring human interpretation. Automated analysis of the report can be achieved but with huge computational complexities. Therefore, it is time consuming [2].

SA is preferred due to reliable detection efficiency, full code coverage, unperceived by malware code and simplicity to generate generic fingerprint of the malware code. In MDS one can extract SA features which will be input to machine learning algorithms for training the system. But due to the introduction of Deep Neural Network (DNN) architecture, huge amounts of data, maybe a vector matrix or an image, can be given as an input for training the network. The next section discusses MDS approach based on an image.

A. Visual Analytic Technique to Solve Challenges in MDS

The malware executable is a binary file, and it can be represented as strings of ones and zeros. A string being an array of hexadecimal values can be reshaped in matrix form and can be viewed as a grayscale image. The technique is Visual Analytic Techniques (VAT) and thus, MD can be put into an image recognition problem. VAT is mostly used for documents where the files are huge and for image analysis where the data is massive. Therefore, the technique is suitable to be used for computer security, as malware attacks are almost always in thousands at any given time.

There are four motivational factors to select VAT. Firstly, the image classification techniques are mature and furthermore it is faster [3]. The second point deals with the mindset of a malware developer. They work hard to hide the code and simultaneously come up with variants of malware; normally they just utilize the old code. Meghna Dhalaria et al. [4] use a robust set of features from static and dynamic malware analysis for creating two datasets i.e., binary, and multiclass (family) classification datasets. In such conditions, for a single malware family, the deviation between two or more gray scale images will be very less which opens a huge number of algorithms based on Similarity Mining Machine Learning (SMML). Thirdly, neither disassembly nor code execution is required for classification based on visualization. Finally, VAT for MD does not require code analysis and it is resilient to obfuscation techniques like polymorphism, packing and section encryption. Samples of malware images are given in Fig. 4. From the images we can conclude that the texture of images of malware families is different. So, texture-based analysis will be a more suitable solution for MD. The next section describes a texture-based approach which is explored in this paper.

B. Granulometry Based Texture Analysis of a Gray Scale Malware Image

Granularity is the random optical texture of an image. In an image, pixels are considered as *'Grains'*. Extracting spatial features such as shape and size from the image is more complex as compared to extracting textural features which does not require any type of segmentation. Researchers have proved that classification accuracy significantly increases if only textural information of an image is taken into consideration. Texture analysis methods are grey level co-occurrence matrix (GLCM), Markov random fields, Laplace filters, discrete wavelet transformation, fractal analysis and GRanulometric Analysis (GRA).

GRA is lesser known, but its significance was proved by Kupidura [5] and Skullmowska [6]. Morphological closing and opening operations as well as measuring the difference between successive images is the base of GRA. This characteristic permits the quantification of different size particles in an image [7]. Haas et al. [8] introduced this technique. Dougherty et al. [9] introduced methods of local analysis which allows assignment of texture values to individual pixels. This feature motivated researchers to use this technique for satellite image analysis. On the similar line we used GRA because in the gray scale malware image, each and every pixel is a malware code byte which is important for the malware family analysis. Thus, GRA provides pixel level analysis.

To the best of our knowledge GRA has not been used till date for analyzing malware images therefore motivating us to work on the same. In ML, feature extraction block and classifier block, both are of utmost importance. Therefore, after finalizing the textural feature for the proposed work the next part describes the classifier used for MDS.

C. Semi Eager (SemiE) Classifier

The task of the classifier is to accurately predict a malware family group of the malware input captured by the system. The learning process is the base of the classifier. Lazy and Eager learning are two types of techniques used in machine learning.

Conditional Random Field (CRF), Support Vector Machine (SVM) and Artificial Neural Network (ANN) are Eager Learning (EL) algorithms. The disadvantages of EL are as follows. The first is the high training time cost e.g., training time for SVM is $O(n^3)$, where n is the number of training instances. The second is the drifting and information loss, leading to over fitting or under fitting risk. The reason is that it computes global models after analyzing prediction query. Finally, there is impact of global distribution on full dataset instead of local behavior of unpredicted targets.

Lazy Learning (LL) or delayed learning is instance based where it memorizes present training examples and waits for the new instance to occur. Thus, in this method instead of estimating the entire instance space it estimates only the different and local instances. Locally Weighted Regression (LWR) and KNN calculate distance to each training example for predicting new instances, thus it follows LL approach. A. Zakai et al. [10] put forth that for the convergence of ML models, local behavior is important. LL can commit a plentiful set of hypotheses. The drawback of LL is, despite no training overhead, prediction time complexity is more i.e., $O(n)$, where n is the number of training examples. The SemiE learning algorithm overcomes the disadvantages of both the training techniques without any compromise of the advantages.

This paper proposes a GRASP MDS architecture consisting of a SemiE learning network for accurate detection and classification of malware families making use of image-based approaches. A benchmark database from Kaggle and malimg is utilized to assess the performance of the system. Features will be extracted from the greyscale image of different malware families and will be trained using SemiE learning.

The primary contribution of the research work is as follows:

1. To provide critical overview of related work based on VAT (image-based).
2. To introduce and extract texture-based feature i.e., GRanulometric Analysis (GRA).
3. To compute Similarity based statistical parameters.
4. Consideration of prominent static features e.g., string signature, byte-sequence, N-grams, OPCODE.
5. To introduce and apply SemiE based classifier.
6. To integrate feature and classifier to present proposed GRASE model which combines texture-based analysis with SemiE classifier for MD.

The above specified combination i.e., granulometric analysis with SemiE classifier, to the best of our knowledge, has not been assessed by the researchers.

This part provides the structured organization of the paper. Section II describes related work to the paper topic as well as GRA techniques used by other researchers in different domains. It also introduces the SemiE classifier. The section also explores varieties of techniques proposed by researchers which will help in formulating the problem statement. In Section III GRASE, the model of the proposed work, has been presented. Feature vector formation of granulometry feature and SemiE classifier mathematical model is described in this section. Section IV provides details about the experimental setup. Section V elaborates experimental results. The system performance of the proposed model is presented in Section VI. The overall conclusion is presented in Section VII.

II. Related Work

This section discusses various features and classification methods investigated by researchers in image-based techniques. It also describes the use of GRA for varieties of applications.

A. Image Based Method

Key benefits of representing malware executable as a 2D image are as follows. Firstly, once the similarity space has been formed, the data dimension does not affect processing. Secondly, it forms equally important clusters and finally, for the clear visualization one can display the similar clusters adjacent to each other [11] - [13].

Malware analysis using VAT with implementation of Self-Organizing Map (SOM) algorithm was proposed by Yoo [14]. S. Foresti [15] demonstrated usage of VAT to represent information like time ('when'), IP address ('where'), Data ('what') and estimated distances to other hosts. The first effort in the direction of visualization technique to visualize binary files for malware detection was done in 2008 [16]. Quist et al. used Ether Hypervisor framework to track and visually represent overall program flow by performing DA [17]. They named the DA framework VERA. Brute Force attack on Secure Shell (SSH) was identified using the VAT by Shiravi et al. [18] and N. Diakopoulos et al. [19]. They represented details of User IDs, Internet Protocol (IP) addresses and various anomalies with the help of different colours. Thus, large network packets were displayed using VAT with which security analysts were able to identify the minuscule details with the help of zoom option. Trinius et al. [20] proposed a novel concept of Malware Instruction SeT (MIST) for monitoring malware. They proposed the use of CW Sandbox for collecting information regarding performed actions and API calls. They used VAT to represent distance matrices of the features for the five malwares.

Further improvement in malware detection was observed when researchers started presenting binary executable sections as grayscale images. These images were used to present detailed structure of malware and even capable of showing small changes in the code. L. Nataraj et al. [21] put forth that the texture of a grey scale malware image can be used to identify similar patterns of the binary code.

Conti et al. [22] presented 'Byte view' visualization where each byte corresponds to a 'single' pixel of an image. The idea is feasible as image pixel and code byte value has the same range i.e., 00 to FF hexadecimal corresponding to different levels of gray scale. So, if the base malware code sequence is the same then it will produce similar images. They also introduced 'Dot Plot' visualization for comparing two images. This presentation helps to identify the presence of similar byte sequences. L. Nataraj et al. [21] visually observed that malware grey scale images were distinct for the different malware families and there was similarity in images for single malware families. Therefore, they extracted image texture-based features (GIST) and used KNN classifiers to classify different malware families. They achieved good average accuracy along with increased speed of malware detection. On a similar line Kancherla et al. [23] used byte plot (image of executable) and achieved 95% accuracy using SVM classifier. Vasanet. al. [24] classified malware images using Convolutional Neural Network (CNN) and Narayan et. al. [25] used Deep Neural Network (DNN).

To detect Trojan, Tian et al. [26] proposed Function Length Frequency (FLF) algorithm. Variable Length Instruction Sequences (VLIS) with ML for malware detection was proposed by Zolkipli [27]. Static Analyzer for Vicious Executable (SAVE) and Disassembled Code (MEDiC) were the two models suggested by Shankarapani et al. [28] for malware detection. The techniques were robust to code obfuscation; thus, results were promising.

Kong et al. [29] used L1 regularized technique to select the best feature from the set of features like PE header, disassembly code and n-gram. They evaluated system performance by using varieties of classifiers like KNN, SVM, Naïve Bayes (NB) and decision tree. They also figured out that PE header features are more prominent in MD.

Santos et al. [30] worked specifically on OPCODE. They tried to relate each OPCODE and calculated OPCODE sequence frequency. They used the same four classifiers used by Kong [29] to evaluate system performance.

Based on the work done by Trinius [20], shaid et al., [98] proposed DA based MDS by observing behavior of malware. They collected behavioral patterns for the operating system resources and API call sequences; and presented the same using color map. They found similarities between these color images using statistical methods. But collecting behavioral patterns is time consuming. K. han et al. [31] proposed hybrid MDS which extracts API calls and OPCODE sequences only. Image matrices were prepared from OPCODE sequences and further given to the classifier for the training purpose. Execution traces were extracted dynamically to avoid binary transformation strategies. However, the method was good for the small-scale MD.

For large-scale malware detection, researchers focused on similarity. According to [32]-[33] similarity should be calculated between all the pairs of points based on Euclidean distance. Maximum similarity corresponds to minimum distance. Normally, similarity patterns can be checked using 2D VAT like projection and semantic orientation [34]-[36]. Windows PE binary file was converted to grayscale image by Han et al. [20]. They calculated entropy of each and every row of an image using the Entropy Graph Generator (EGG). MD was performed based on similarity of the present file with the original binary file. Arefkhaniet et al. [37] proposed an image processing technique i.e., Local Sensitive Hashing for classifying similar malware images having high probability. Grey scale image was prepared by disassembling binary executable into OPCODE sequences [38]. They first reduced dimensionality using PCA and then used KNN to classify the malware images.

S. Rezaei [39] used similarity measurements by comparing OPCODE strings of malware files and tried to reduce detection time. Colored based VAT for analyzing malware attack chronology was used by Venkatraman [40], Zhang [41] and Wylie Shanks [42]. Successful system connection was demonstrated by them. J. Zhang et al. [9] extracted local texture features of grayscale image as well as OPCODE instructions of disassembly file, to train Random Forest (RF) classification model. Shiqui et al. [43] extracted two features i.e., texture of malicious code and the frequency of instructions in the code. These features were used to train SoftMax classifiers and stacked auto encoders. Liu et al. [44] proposed enhancement of information density of malware images, where the '.text" section of malware code is visualized. To improve accuracy Wuechner et al. [45] proposed MDS based on data compression mining on the data flow graph.

The overall extract from the above discussion is that the malware grey scale image analysis is not affected by code obfuscation, therefore opted by many researchers. In addition to that, texture is an important parameter for the grey scale image which can be used to find out similarity between malware images.

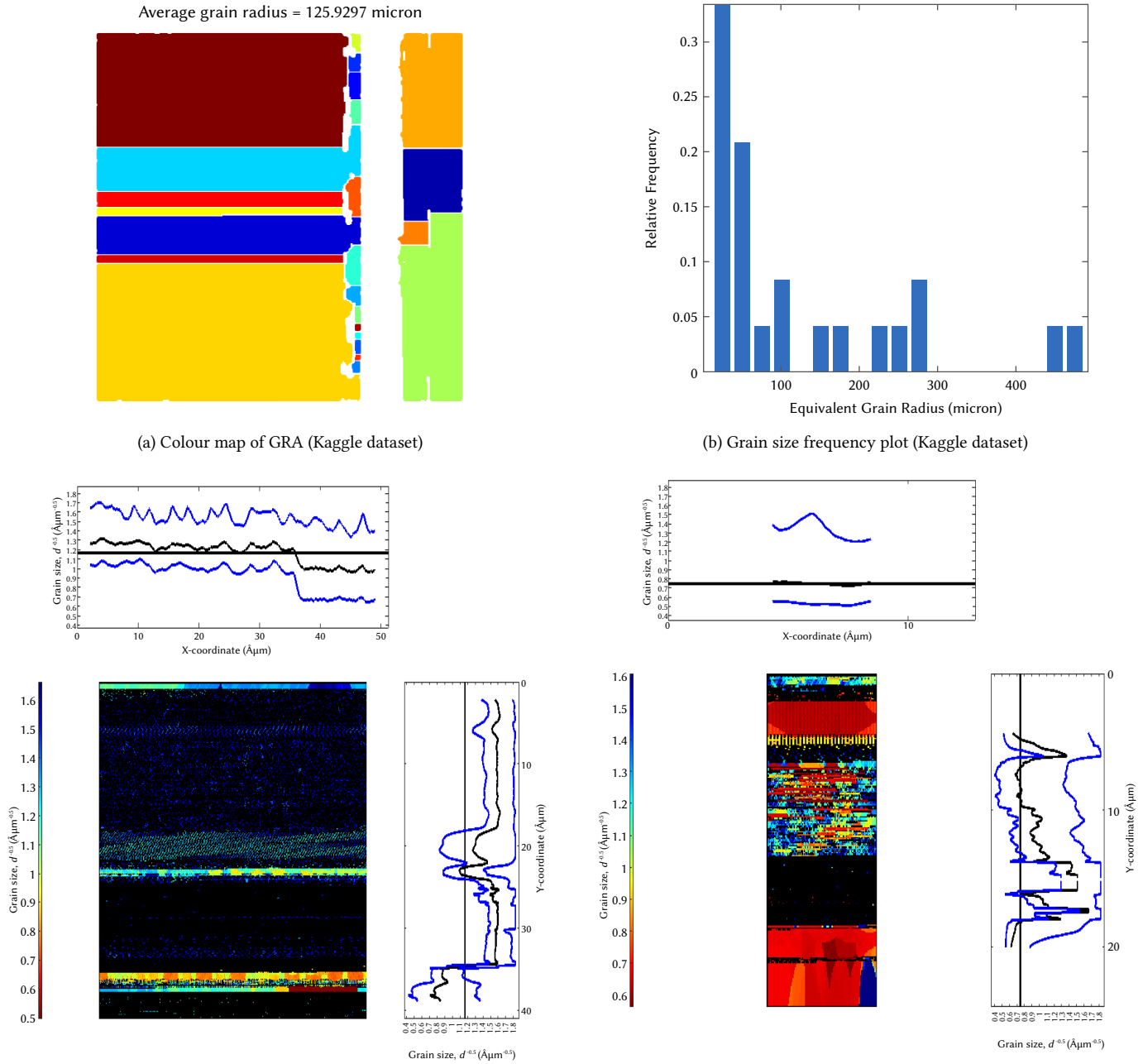
The next section provides related work done in Granulometry analysis and shows that the technique is versatile to image processing.

B. Granulometry Analysis

GRA concept was introduced in section I.B. This technique covers spatial as well as spectral characteristics of the malware grey scale image. GRA accuracy for classifying satellite images has been demonstrated by Kupidura et al. [5]. Basic GRA technique is based on morphological opening and closing. Extension of basic GRA is with a Multiple Structuring Element (MSE). Basic GRA and GRA with MSE have slightly different properties. So, it can give different results. GRA is also used to find the distribution of object sizes of an image [46]. We are proposing extraction of black patches from the grey scale image using successive closings by reconstruction.

GRA profiles (morphological profiles) can be used for image classification [47]. Thus, the technique is useful for malware family identification. Our main objective is to analyze the retrospective changes of the malware grey images for the family.

As GRA technique is relatively unknown, we are presenting two advantages of the system. The first advantage is multi-scality which is more suitable for malware detection. In this, we can obtain information about the texture grains of different sizes because of the possibility of successive application of increasing size of morphological opening and closing operations. Secondly, it is resistant to edge effects. In a



(a) Colour map of GRA (Kaggle dataset)

(b) Grain size frequency plot (Kaggle dataset)

(c) Malimg dataset - Adialer.C malware GRA [48]

(d) Malimg dataset - Dontovo. A malware GRA [48]

Fig. 1. GRA output for Kaggle Malware Family.

typical texture analysis process, these edges have low texture, but still, it will provide high value of texture. This is due to the fact that those methods are based on spatial frequency analysis and imagery edges have high spatial frequency, resulting in high texture.

But as GRA is not based on this principle, it analyses value and number of the removed image elements, resulting in normal texture value of the edges in an image. Thus, the GRA seems to be more suitable for the MDS. Fig. 1 depicts granulometry output for one malware family from the Kaggle dataset and two malware families from the Malimg dataset. It shows a colour map for the different grain sizes. It also represents the relative frequency of different grain sizes present in an image. The grain size is measured using a point-sampled intercept length method. Related information is elaborated in section – III.A The next section introduces the main block of ML i.e., classifier.

C. Classifier

This section covers the classifier used by researchers for malware detection. Feature extractions and classification techniques are two basic blocks of MDS. API calls, system calls, n-gram and OPCODE are features that are still being used extensively in MDS. Work carried on by researchers implementing any approach is unique and makes use of different types of bytes and features related to hex. In addition to that we are also introducing a GRA feature. Thus, feature vectors extracted from the malware grey scale image have to be trained using the SemiE network. After training the SemiE model, system performance is evaluated by applying real time malware data. The next part elaborates upon the selection of SemiE.

The role of the classifier is crucial as it lays the groundwork for precision and accuracy. A research problem must cover the core subject matter and at the same time it must lead to hitherto undiscovered knowledge. This goal entails not only an extensive literature survey, but also mandates interpreting the surveyed information accurately to attain an appropriate research path. To make it more feasible to extract the requisite data, we have included graphical presentations (Refer Fig. 2).

The learning process is either Eager learning or Lazy learning (instance based). Key extract from the survey is both the techniques are at par. But the advancement in the Neural Network (NN) technique led to Deep Neural Network (DNN), which has opened up different segments altogether. But the major drawback of DNN is that the network is data hungry. This was the motivational factor where we thought of combining the advantages of lazy and eager and introduced Semi Eager learning. SemiE will reduce testing and training time, which is needed for malware detection methods, as it has to run in real time even as it detects malware in the fastest ways possible. This motivates us to choose the SemiE method with considerably low computational overhead and yet this technique has not yet been investigated for detection of malware. The next section elaborates the proposed model of GRASE.

III. Proposed Model: GRASE- GRanulometry Analysis With Semi Eager Classifier to Detect Malware

Fig. 3 shows the architecture of the GRASE model. The first step is to provide input to the model which is a malware file. There are three basic analysis techniques like SA, DA and hybrid, explored by researchers. While the SA method is the choice of many researchers, the hybrid technique is not so popular with them.

In step two, Malware Binary File (MBF) is read and features are extracted. The first feature set is SA based which has HEX dump-based features and disassembled file features. These features are common and must be used for malware prediction, therefore the same has been just specified and focus is given on the proposed technique.

The HEX dump-based features are n-gram, Meta-data (MD1), entropy [92] [93], Haralick and Local Binary Pattern (LBP) features. The disassembled file features are Meta-Data (MD2), symbol (SYM) [94], Register (REG) [97], Operation Code (OPCODE) [53], [95]-[96], DP and Section (SEC). Miscellaneous (MISC) feature should be done manually with the identification of keywords from the disassembled code. The Interactive Disassembler (IDA) tool can also be utilized for this purpose. Types of features that are extracted are number of imported DLLs, identifying strings viz. hkey_local_machine (it specifies access to specific paths of Windows registry), number of blocks in PE, etc. Hence, it is dependent on how experienced the MD software development engineer is.

Very few malware programs make use of the packing technique and hence they do not use API calls. Instead, they contain a few OPCODEs. Generally, such programs use assembler related directives like Define Byte (db), Define Word (dw) and Define Double Word (dd). This feature plays a significant role in the classification of the varieties of malware families.

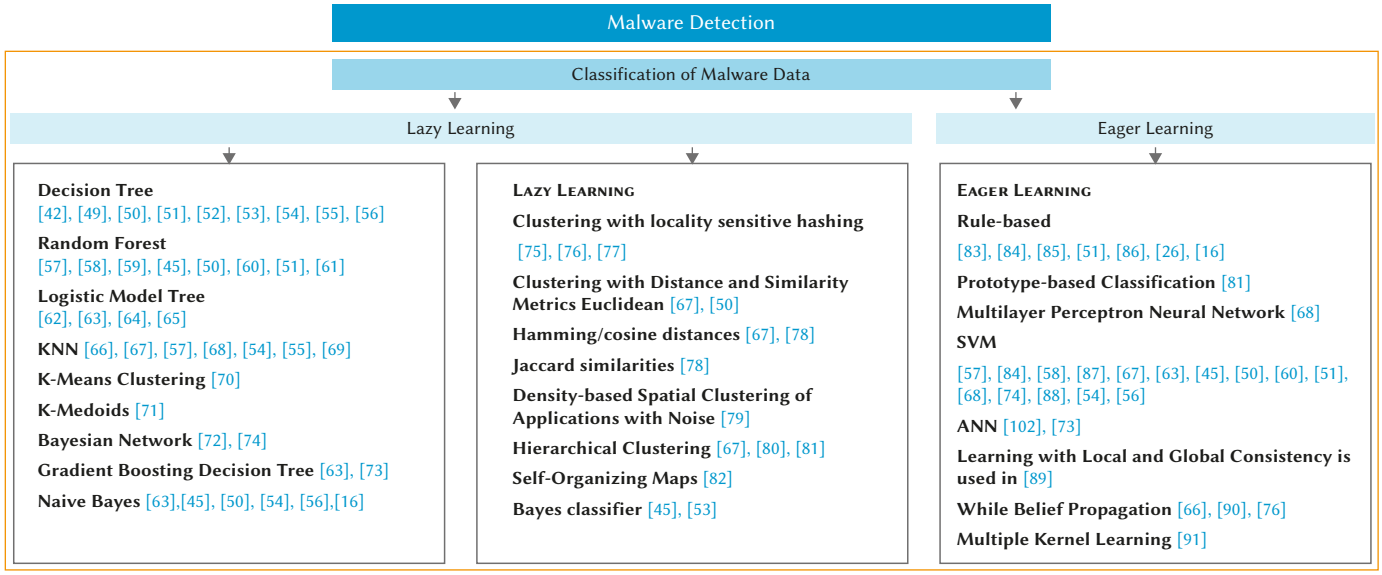


Fig. 2. Classifier based Literature Survey.

In step three, to extract GRanulometry features as well as Image Similarity based Statistical Parameter (ISSP), MBF is represented as a grayscale image. ISSP based features are Normalized Cross correlation (NCC), Average difference (AD), Maximum difference (MaxD), Singular Structural Similarity Index Module (SSIM), Laplacian Mean Square Error (LMSE), MSE and PSNR.

Sections III.A, III.B, III.C and III.D describe the proposed model shown in Fig. 3.

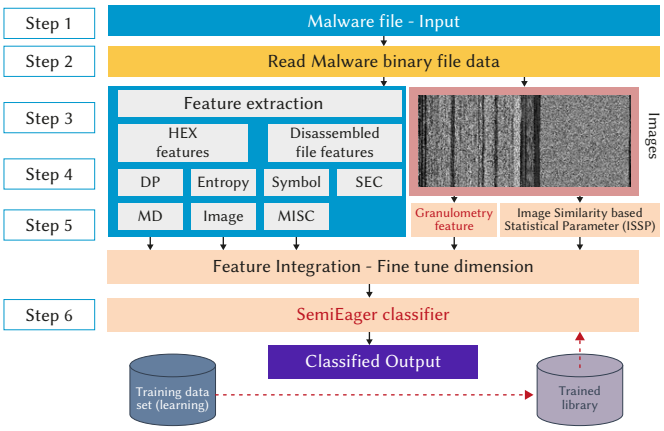


Fig. 3. Architecture Diagram: GRASE Model.

A. GRanulometry Analysis (GRA)

Step four is to generate granulometric profiles for each image pixel. GRA is based on the sequence of morphological opening and closing operations which are applied to gray scale image using set of known size and shape called the Structuring Element (SE). SE size is based on the pattern or a structure one would like to extract from the image. SE is normally a disk of size λ. In the process of closing by reconstruction, using SE will erase the dark spots of size less than λ during dilation process. Erased dark spots will not be recovered with multiple reconstructions, resulting in extraction of image structure having different sizes. Equation (1) represents granulometry density which describes the size of the image structures.

$$GRA_{\lambda} = \frac{|\emptyset_{\lambda}(I) - \emptyset_{\lambda-1}(I)|}{I}, \ \lambda \geq 1 \quad (1)$$

where, $\emptyset_{\lambda}(I) = Closing\ by\ reconstruction, \lambda - Radius\ of\ disk\ (integer\ value)$.

The operations are performed pixel wise. Granulometry profile may be written as

$$G(p) = [GRA_1(p), GRA_2(p), GRA_3(p) \ldots \ldots GRA_n(p)]$$

where, $n = Granulometrylevels.\ This\ parameter\ is\ configurable.$

GRA is used to measure the difference between two images by measuring the quantity of particles having different sizes by calculating Volume Weighted Average Grain Size (VWAGS). Refer Equation (2).

$$\text{VWAGS} = G_v = \frac{1}{V_I}\sum_{j=1}^{n} V_j * G_j \quad (2)$$

Where, V_I *is total image size,* V_j *is the volume of grains corresponding to the grain size* G_j.

VWAGS will always be larger than the average grain size as per Eq. (1). It has been observed that VWAGS is able to capture the influence of grain size distribution [48].

Researches have used this technique to analyse satellite images. As in satellite images each pixel (granule) is important, on similar lines in malware image analysis each pixel carries important information of malware property. GRA can also be based on Multiple Structure Element (MSE). There are two main advantages of GRA. Firstly, it has a property of multi-scalability. Due to a greater number of morphological operations the information obtained will have texture grains of varieties of sizes.

Secondly, the analysis is resistant to edge effect. Edge effect is observed in fairly all texture analysis techniques where edges of the object normally have low texture, but it will get high value. This effect is observed as texture analysis methods are based on spatial frequency analysis and normally edges have a high spatial frequency, exhibiting high texture. GRA is not based on this fundamental as analysis is based on value and number of removed image elements and therefore edges are not displayed as areas of high texture. This property will help analyze malware images more accurately.

B. ISSP

Step five focuses on the similarity between two images. A comparison of malware images from the *'x'* family with themselves and the rest of the families is undertaken, and a similarity parameter matrix is computed based on this. There are two input images, namely, Reference image (R_I) and Input image (I_I). If R_I is from the *'x'* family then I_I represents the rest of the images from the *'x'* family and the images from the other families. During the entire process of computing the similarity parameters, R_I will remain constant. Since the number of images for every family is in the thousands their mean value will be computed [2].

The NCC method is utilized fortemplate matching. This is a procedure utilized for finding incidences of a pattern or object within an image. Eq. (3), is used to calculate NCC.

$$NCC(R_I, I_I) = C_{R_I I_I}(\widehat{R_I}, \widehat{I_I}) = \sum_{[m,n]\in R} \widehat{R_I}(m,n)\widehat{I_I}(m,n) \quad (3)$$

where, $\widehat{R_I} = \frac{R_I - \underline{R_I}}{\sqrt{\sum (R_I - \underline{R_I})^2}}$, $\widehat{I_I} = \frac{(I_I - \underline{I_I})}{\sqrt{\sum (I_I - \underline{I_I})^2}}$

AD provides the average [2] of change regarding the input image and the reference image. AD can be represented as follows:

$$AD(R_I, I_I) = \frac{1}{mn}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_I(m,n) - I_I(m.n)] \quad (4)$$

MaxD provides the maximum of the error signal (i.e., the difference between the processed and reference image). *MaxD* is defined as follows:

$$MaxD(R_I, I_I) = \{[R_I(m,n) - I_I(m.n)]\} \quad (5)$$

SSIM is based on three factors [2], namely,luminance, contrast, and structure in order to be more in line with the workings of the human visual system. It is a perceptual metric that quantifies image quality degradation. This parameter is chosen as the malware developer alters the old code and comes up with the modified code. The modified code can be deemed to be the *'Noise'* element in an image. SSIM is defined as follows:

$$SSIM(R_I, I_I) = [l(R_I, I_I)^{\alpha} . c(R_I, I_I)^{\beta} . s(R_I, I_I)^{\gamma}] \quad (6)$$

where $l = luminance, c = contrast, s = structure$

The Laplacian error map [2] shows spatial error distribution across an image. The overall image quality is given by LMSE as follows:

$$LMSE(R_I, I_I) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}[L(R_I(m,n)) - L(I_I(m.n))]^2}{\sum_{m=1}^{M}\sum_{n=1}^{N}[L(R_I(m,n))]^2} \quad (7)$$

where $L((m,n))$ *is the Laplacian operator*

NAE measures [2] the numerical variance between the R_I and I_I. Additionally, the results that are closer to zero indicate that the image is highly similar to the original image and the results close to the value one mean that the quality of the image is very poor. NAE is calculated as follows:

$$NAE(R_I, I_I) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} |R_I(m,n) - I_I(m.n)|}{\sum_{m=1}^{M} \sum_{n=1}^{N} [R_I(m,n)]} \quad (8)$$

MSE and PSNR are used to compare the quality of image compression [2]. MSE represents the cumulative squared error between the R_I and I_I, whereas the PSNR represents a measure of the peak error. The lower the value of the MSE, the lower the error.

$$MSE(R_I, I_I) = \frac{1}{mn} \sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n)) - L(I_I(m.n))]^2 \quad (9)$$

$$PSNR = 10 * \frac{255^2}{MSE} \quad (10)$$

Once the ISSP has been computed, one more feature vector is generated which will be utilized to train the classifier.

To compute various statistical parameters, to begin with, all the malware families were segregated into different folders for the Kaggle dataset. There are 9 malware families in the Kaggle dataset so 9 folders were created. Grey images were produced after processing malware files. These images will be used to compute ISSP parameters. Malimg dataset is already organized and has grey scale images so it is ready to act as an input to the following algorithm. Table I presents the algorithm.

C. SemiE Classification Module

Finally step six corresponds to a SemiE classifier, whose mathematical model is explained here. In the training process, SemiE stores only the Centre Point for each class. SemiE training time complexity is $O(n)$, where n is the number of training instances. $O(k)$ is the prediction complexity of space and time, where k represents the number of categories.

1. SemiE algorithm

Let's say, $X = Input\ space$.

It has *set of n dimension vector*, $X \subset R^n$.

$Y = Output\ space$.

It has set of class labels $\{c_1, c_2, \ldots\ \ldots\ \ldots, c_k\}$, *where* $c_i \in Z$

$P(X, Y) = Joint\ probability\ distribution\ over\ X\ and\ Y$

Assume that, *Feature vector* $x \in X$ and *corresponding label* $y \in Y$.

$T\{(x_1, y_1), (x_2, y_2), \ldots\ \ldots\ \ldots\ \ldots, (x_N, y_N)\}$, is the training data set,

where x_i *and* y_i *are instances of X and Y, respectively*

$PS = Partition\ of\ set\ T = \{(PS_1) \cup (PS_2) \cup \ldots\ \ldots\ \ldots\ \ldots \cup (PS_k)\}$,

Where $PS_j = \left\{(x_i, y_i)|_{y_i = c_j}\right\}$, *and* $\left(\frac{i}{j}\right) = 1,2, \ldots\ \ldots\ \ldots, N.$

Learning algorithms learns these two probabilities, namely Conditional Probability Distribution and Prior Probability.

$$Prior\ Probability = P(Y = c_j), where\ j = 1,2, \ldots., k$$

$$conditional\ probability\ distribution = P\left(X = x_i \mid Y = c_j\right)$$

$$\therefore\ PosteriorProbability = P\left(Y = c_j \mid X = x_i\right)$$

$$= \frac{P\left(X = x_i \mid Y = c_j\right) * P(Y = c_j)}{\sum_{m=1}^{k} P\left(X = x_i \mid Y = c_m\right) * P(Y = c_m)} \quad (11)$$

The algorithm, while predicting input x_i will provide the output class label based on maximum posterior probability.

$$C = arg\ arg \frac{P\left(X = x_i \mid Y = c_j\right) * P(Y = c_j)}{\sum_{m=1}^{k} P\left(X = x_i \mid Y = c_m\right) * P(Y = c_m)}$$

As the denominator part is independent of c_j, it is constant, resulting in

$$C = arg\ arg P\left(X = x_i \mid Y = c_j\right) * P(Y = c_j) \quad (13)$$

TABLE I. Statistical Parameter Computation

Input — Folder structure is as follows.
Main folder – contains sub folders equal to number of malware families ($i = 9$ for this case)
- Sub-folders (9 malware families)
- Each sub-folder has different number of images j

$R_I = Reference\ image$
$I_I = Input\ image$
$i = Number\ of\ malware\ families\ in\ main\ directory\ (folder)$
$j = Number\ of\ malware\ variants\ (images)\ of\ specific\ malware\ family\ in\ a\ subfolder$

// Initialize empty array
Parameter Array = {∅}
for $(\beta = 0\ ;\ \beta < i\ ;\ \beta\ ++)$
// Load reference image – first image of malware family
$R_I = \beta[0]$
***for*// select malware families one by one**
$(k = 0\ ;\ k < i\ ;\ k\ ++)$
// Get number of images present of a specific malware family
$j = size\ (k_{sub-folder})$
for $(local_{cnt} = 0;\ local_{cnt} < j;\ local_{cnt}\ ++)$
// Load Input image from malware family
$I_i = (k)[local_{cnt}]$
// calculate SSIM
$SSIM(R_i, I_i) = [l(R_i, I_i)^{\alpha} . c(R_i, I_i)^{\beta} . s(R_i, I_i)^{\gamma}]$
where $l = luminance,\ c = contrast,\ s = structure$
// Calculate MSE

$$MSE(R_I, I_I) = \frac{1}{mn} \sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n)) - L(I_I(m.n))]^2$$

// calculate PSNR

$$PSNR = 10 * \log_{10} \frac{255^2}{MSE}$$

// calculate Normalized Cross-Correlation (NK)

$$NCC(R_I, I_I) = C_{R_I I_I}\left(\widehat{R_I}, \widehat{I_I}\right) = \sum_{[m,n] \in R} \widehat{R_I}(m,n) \widehat{I_I}(m,n)$$

$$\widehat{R_I} = \frac{R_I - \overline{R_I}}{\sqrt{\sum (R_I - \overline{R_I})^2}},\ \widehat{I_I} = \frac{(I_I - \overline{I_I})}{\sqrt{\sum (I_I - \overline{I_I})^2}}$$

// calculate Normalized Absolute-Error (NAE)

$$NAE(R_I, I_I) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} |R_I(m,n) - I_I(m.n)|}{\sum_{m=1}^{M} \sum_{n=1}^{N} [R_I(m,n)]}$$

// calculate Maximum difference
$MD(R_I, I_I) = \max\{[R_I(m,n) - I_I(m.n)]\}$
// calculate Laplacian Mean Square Error (LMSE)

$$LMSE(R_I, I_I) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n)) - L(I_I(m.n))]^2}{\sum_{m=1}^{M} \sum_{n=1}^{N} [L(R_I(m,n))]^2}$$

where $L((m,n))$ *is Laplacian operator*
// Store all the values in an array
end
// Take average of an array an obtain single value
Parameter array(k)= [mean (SSIM); mean (MSE); mean(PSNR); mean(NCC); mean(NAE); mean (MaxD); mean (LMSE)]
end
end

2. Parameter Estimation

The Maximum Likelihood Estimation technique is used to estimate the parameters, therefore the indicator function and prior probability, both that may be computed as,

$$P(Y = c_j) = \frac{\sum_{i=1}^{N} I(y_i = c_j)}{N} \quad (14)$$

$j = 1, 2, \ldots\ \ldots, k,\ \ N = number\ of\ samples\ in\ trianing\ set$

$$I(u = v) = \begin{cases} 1, & if\ u = v \\ 0, & otherwise \end{cases}$$

Posterior probability is calculated using Central Limit theorem. In this theorem whenever a number of samples N crosses a threshold limit T_H, then the input element x_i in the data set T generally follows a normal distribution having mean μ and variance σ^2.

Thus, if $|PS_j|$ is greater than T_H (let $T_H = 30$), then following the Central limit theorem x_i will have normal distribution with variance σ^2 and centred around μ_j.

$\therefore$ *conditional probability distribution* is given by,

$$P\left(X = x_i \mid Y = c_j\right) = \frac{1}{\sqrt{2 * \pi * \sigma^2}} e^{\left(-\frac{1}{2*\sigma^2}(x_i-\mu_j)^2\right)} \tag{15}$$

By substituting Eq. (14) to Eq. (12), we get

$$C = arg\ arg \frac{1}{\sqrt{2 * \pi * \sigma^2}} e^{\left(-\frac{1}{2*\sigma^2}(x_i-\mu_j)^2\right)} \tag{16}$$

After discarding c_j independent term and constant values from the Eq. (15), we get,

$$C = arg\ arg e^{\left(-\frac{1}{2*\sigma^2}(x_i-\mu_j)^2\right)*P(c_j)} \tag{17}$$

$$C = arg\ \arg\ ln\ e^{\left(-\frac{1}{2*\sigma^2}(x_i-\mu_j)^2\right)*P(c_j)} \tag{18}$$

$$C = arg\ arg\left(lnP(c_j) - \frac{1}{2 * \sigma^2}(x_i - \mu_j)^2\right) \tag{19}$$

$$C = arg\ arg\left(\frac{1}{2 * \sigma^2}(x_i - \mu_j)^2 - ln\ lnP(c_j)\right) \tag{20}$$

$$C = arg\ arg\left((x_i - \mu_j)^2 - 2 * \sigma^2 * ln\ lnP(c_j)\right) \tag{21}$$

Condition – 1: If all the classes are having the same prior probabilities, then Eq. (21) can be written as

$$C = arg\ arg\left((x_i - \mu_j)^2\right), if\ P(c_i) = P(c_j)(i \neq j) \tag{22}$$

where $\mu_j = j^{th}$ class centre

Classification of instance x_i, is based on the class centers of every class,

Let, $x_i = (x_i^1, x_i^2 \dots\dots\dots, x_i^n)$, then.

$$\mu_j = \frac{\sum_i I(y_i = c_j)x_i}{\sum_i I(y_i = c_j)} = \frac{\sum_i I(y_i = c_j)(x_i^1, x_i^2 \dots\dots\dots, x_i^n)}{\sum_i I(y_i = c_j)} \tag{23}$$

Table II presents pseudo code of SemiE algorithm.

TABLE II. SemiE Classifier Pseudo Code

Training Phase

Mean calculation, $\mu = \frac{\sum_{i=1}^{N} x_i}{N}$ (24)

Variance calculation,

$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu\ )^2}{N}$ (25)

for $j = 1$ *to* N

  for $i = 1$ *to* k

  $P(Y = c_j) = \frac{\sum_{i=1}^{N} I(y_i = c_j)}{N}$ (26)

  $\mu_j \leftarrow \frac{\sum_i I(y_i = c_j)x_i}{\sum_i I(y_i = c_j)} = \frac{\sum_i I(y_i = c_j)(x_i^1, x_i^2 \dots\dots\dots, x_i^n)}{\sum_i I(y_i = c_j)}$

Regularization term $r_j \leftarrow -\sigma^2 * ln\ lnP(Y = c_j)$ (27)

  end

end

Testing Phase

Instance x_i *is to be classified*

$C \leftarrow arg\ arg\left((x_i - \mu_j)^2 + r_j\right)$ (28)

Significant properties of SemiE classifier are listed as follows. It performs incremental learning leading to training time complexity - $O(n)$ and prediction time complexity - $O(k)$. If the special case where the data point x_i is ready for classification but has equal distance from all the class labels is given, then the next level of decision making is done which is based on frequency. A regularization term will be assigned to the class label having the highest frequency.

IV. Experimental Setup

The system performance is analyzed on a benchmark database from Kaggle[3] as well as the 'malimg' dataset[4]. Detailed information about both databases is given in Table III and Table IV. The major difference between these two datasets is that in 'malimg' set directly grey scale images were given, so, for the Kaggle dataset one additional coding function is required to convert malware files to grey scale image.

TABLE III. Kaggle and Malimg Datasets Basic Information

Header	Kaggle dataset	Malimg dataset
Download	Microsoft malware classification challenge from Kaggle	Vision research Lab
ID	Twenty-character hash value for unique identification of file	Thirty-two-character hash value for unique identification of file
Number of malware families / Size	9 / 0.5 Tera byte uncompressed	25 / 1.09 GB uncompressed, in image form
RAW data	HEX representation of the file's binary content	HEX representation of the file's binary content
Class	Integer representing malware family	Integer representing malware family
Metadata manifest	Log of various metadata information e.g. Function calls, Strings etc. extracted from the binary using IDA disassembler tool.	----------

V. Results

This section presents the results achieved throughout the process of malware analysis. Initially the Kaggle dataset malware files were converted to grey scale images, as shown in Fig. 4. It has been clearly noted that the image for each of the families is unique. 'malimg' dataset images are not shown as it is readily available from the website.

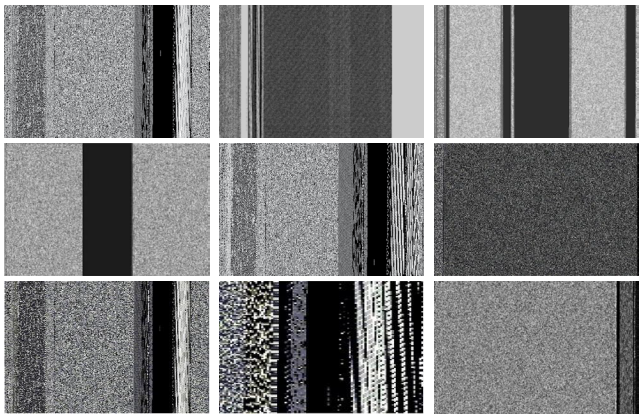


Fig. 4. Malware images of different malware families from the Kaggle dataset.

[3] https://www.kaggle.com/c/malware-classification/

[4] https://paperswithcode.com/dataset/malimg

TABLE IV. KAGGLE AND MALIMG DATASETS DESCRIPTION

Kaggle dataset		Malimg dataset						
Malware Family	**Malware category**	**Sample size**	**Malware Family**	**Malware category**	**Sample size**	**Malware Family**	**Malware category**	**Sample size**
Gatak	Backdoor	1013	Allaple.L	Worm	1591	Alueron.gen! J	Trojan	198
Obfuscator. ACY	obfuscated malware	1228	Allaple.A	Worm	2949	Malex.gen! J	Trojan	136
Kelihos_ver1	Backdoor	398	Yuner.A	Worm	800	Lolyda.AT	PWS	159
Tracur	TrojanDownloader	751	Lolyda.AA 1	PWS	213	Adialer.C	Dialer	125
Simda	Backdoor	42	Lolyda.AA 2	PWS	184	Wintrim.BX	Trojan Downloader	97
Vundo	Trojan	475	Lolyda.AA 3	PWS	123	Dialplatform.B	Dialer	177
Kelihos_ver3	Backdoor	2942	C2Lop.P	Trojan	146	Dontovo.A	Trojan Downloader	162
Lollipop	Adware	2478	C2Lop.gen! G	Trojan	200	Obfuscator.AD	Trojan Downloader	142
RAmnit	Worm	1541	Instantaccess	Dialer	431	Agent.FYI	Backdoor	116
			Swizzor.gen! I	Trojan Downloader	132	Autorun.K	Worm: AutoIT	106
			Swizzor.gen! E	Trojan Downloader	128	Rbot! gen	Backdoor	158
			VB.AT	Worm	408	Skintrim.N	Trojan	80
			Fakerean	Rogue	381			

TABLE V. CONFUSION MATRIX

Malware	**Malware Detection %**								
	Ramnit	**Lollipop**	**Kelihos_ver3**	**Vundo**	**Simda**	**Tracur**	**Kelihos_ver1**	**Obfuscator. ACY**	**Gatak**
RAmnit	99.61	0.00	0.06	0.06	0.00	0.06	0.06	0.06	0.06
Lollipop	0.04	99.80	0.00	0.12	0.00	0.00	0.00	0.00	0.04
Kelihos_ver3	0.03	0.00	99.90	0.00	0.00	0.00	0.00	0.07	0.00
Vundo	0.00	0.21	0.21	99.37	0.21	0.00	0.00	0.00	0.00
Simda	0.00	0.00	0.00	0.00	95.24	2.38	2.38	0.00	0.00
Tracur	0.00	0.00	0.00	0.00	0.00	99.73	0.13	0.13	0.00
Kelihos_ver1	0.00	0.25	0.25	0.00	0.25	0.00	99.24	0.00	0.00
Obfuscator.ACY	0.08	0.08	0.08	0.08	0.00	0.00	0.08	99.43	0.16
Gatak	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.20	99.70

The next step was to extract the ISSP feature set from the grey scale images (refer section III.C). Suppose in the dataset, there are *MF number of malware families* For Kaggle dataset set *MF* is 9 and for Malimg dataset it is 25. There are *S number of samples* per malware family. For ISSP computation we require two images namely '*Reference Image* (R_I)' *and* '*Input image* (I_I)', R_I will remain constant through the iteration and I_I will change. Initially, R_I will be selected from the 1st malware family, and 1st image in the folder. Remaining all the images from 1st malware families and all the images of the remaining malware families will be now 'I_I'. R_I and I_I will be used to compute the ISSP parameters. Computed ISSP parameters per I_I will be stored in respective array say for example PSNR_array, NK_array and so on, with respect to malware family. After iterating through all the images of any one family, the mean value of an array will be computed. Hence, for every family there will be just one single mean value. The mean value matrix will be plotted. The same is illustrated in Fig. 5. Since the 'malimg' dataset is massive and has a large number of malware families we present the result of this dataset.

Fig. 5(1) shows the MD value. It is 225 for the first family of malware, but for the rest of the families the value is 255. Therefore, there is high structural similarity with self-family, but with other families a higher MD value reflects slower similarity. We can decide a threshold of 145 to 250 for differentiating between malware families. SSIM, PSNR, MSE, NK and NAE plots are on a similar line to MD i.e., clear bifurcation between self-family and other families, but the threshold values will be different. However, for the AD parameter for the self-family value is near to '0' and for other malware families it is either a high positive or high negative value (refer Fig. 5(7)), so AD demands for the hysteresis-based threshold. SC parameter value reflects overlap i.e., defining proper threshold is difficult, so we discarded this parameter for the training purpose (refer Fig. 5(8)). The remaining parameters can be used to train the SemiE classifier. The grain size selected for GRA analysis is 5,7,10 and 13 with MSE.

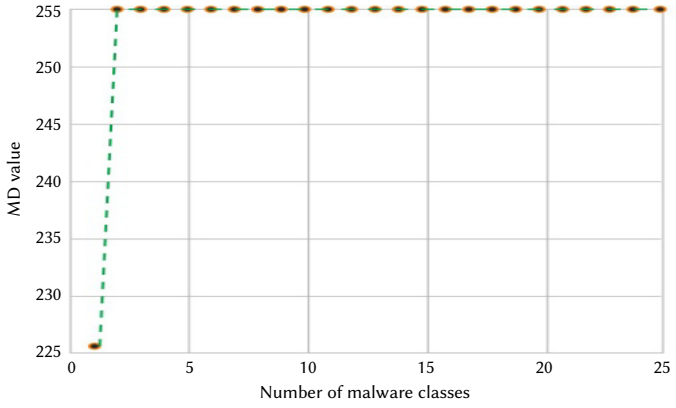
VI. PERFORMANCE EVALUATION

This section presents a confusion matrix and compares the results obtained from the proposed work with the state-of-the art methods. The proposed method was validated with the Big 2015 Kaggle and Malimg datasets and therefore results are compared with those research techniques that also have been validated with the same datasets.

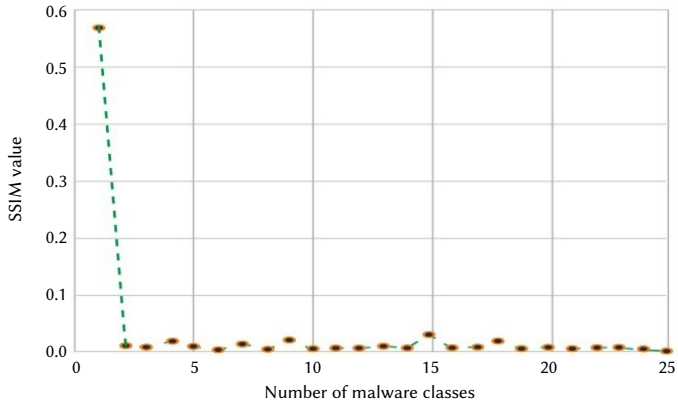
Accuracy is a significant performance parameter for the MD system. It specifies accurate classification of malware. Accuracy is computed on the basis of the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} X\ 100$$

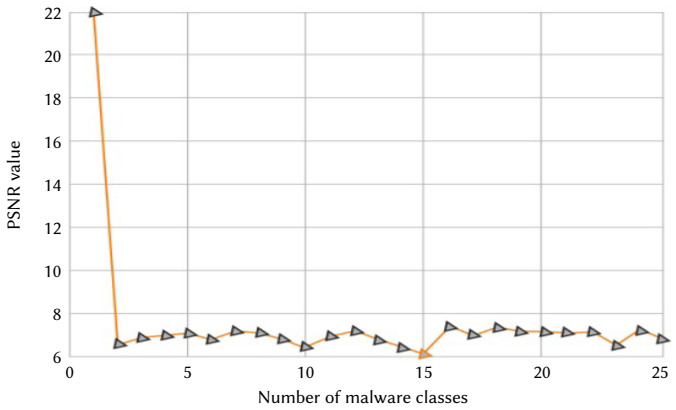
Table V shows the confusion matrix of the proposed MDS using Kaggle dataset.



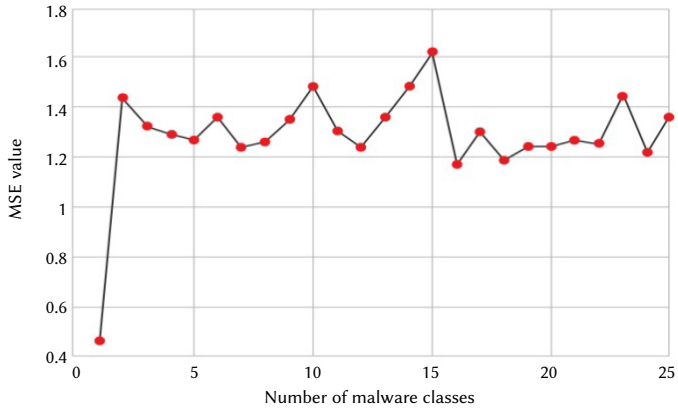
(1) MD Plot



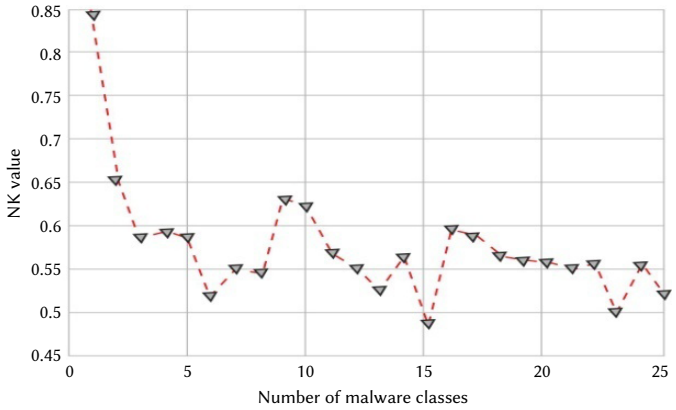
(2) SSIM



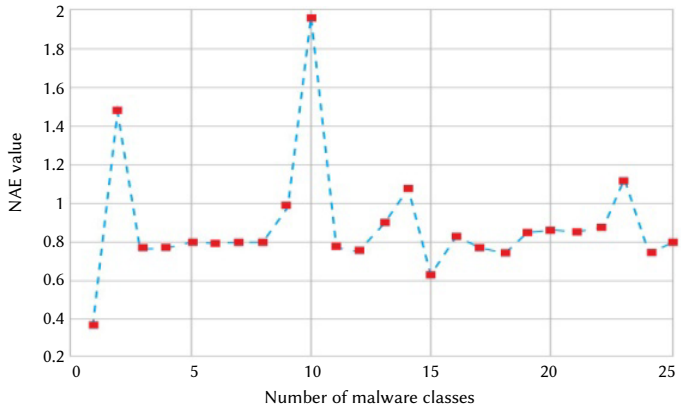
(3) PSNR



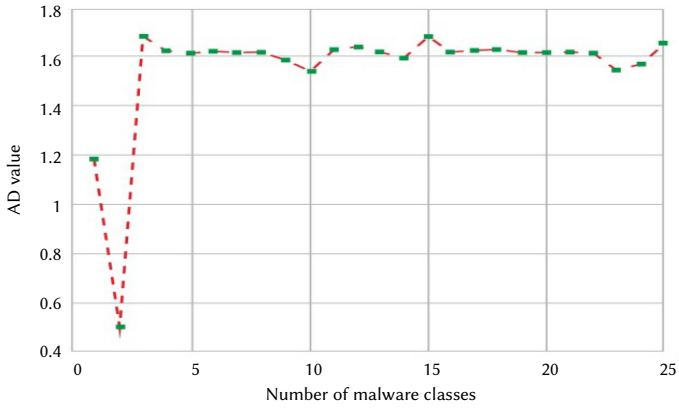
(4) MSE



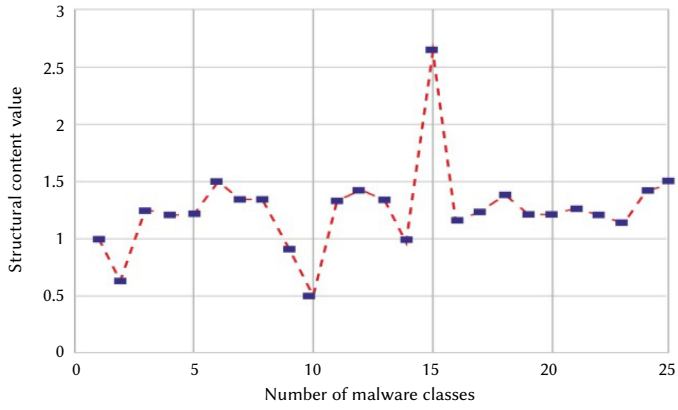
(5) NK



(6) NAE



(7) AD



(8) SC

Fig. 5. Parameter plots - (1) MD (2) SSIM (3) PSNR (4) MSE (5) NK (6) NAE (7) AD (8) SC.

Graphical plots of confusion matrix are illustrated in Fig. 6 and Fig. 7.

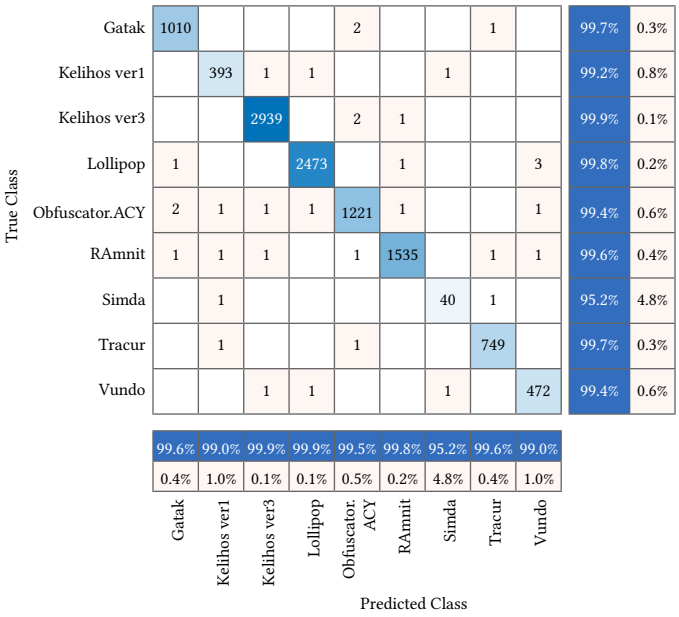


Fig. 6. Confusion Matrix Plot – Kaggle data set.

A. State of the Art Comparison

As SemiE classifiers are not used by other researchers, we compared the results with learning algorithm presented by researchers. Table VI shows a comparative of the performance of the proposed and other methods.

Malimg and BIG 2015 datasets are the datasets commonly used by researchers. It has been observed that researchers work on either of the datasets, but this paper explores both the datasets. This makes the proposed work more robust.

CNN technique is also common in researchers. This technique is the basis for deep learning. Few researchers used variants of CNN i.e. DenseNet, ResNet-50 etc. Minimum accuracy of 98.23% to maximum 99.3% is achieved for malimg dataset. The proposed technique provides 99.03% accuracy. BIG2015 dataset is from Kaggle. Minimum 96.9% and maximum 99.73% accuracy achieved for the BIG2015 dataset. The proposed work provides 99.11% accuracy. Thus, the proposed provides minimum 99% accuracy for both the dataset.

VII. Conclusion

Malware is a common attack on the internet. Developers of malware detection systems are continuously contending with cyber-attackers. For maintaining persistent pressure on cyber attackers, MDS developers should work out new strategies which can capture malware without any loss to the user and quarantine the same.

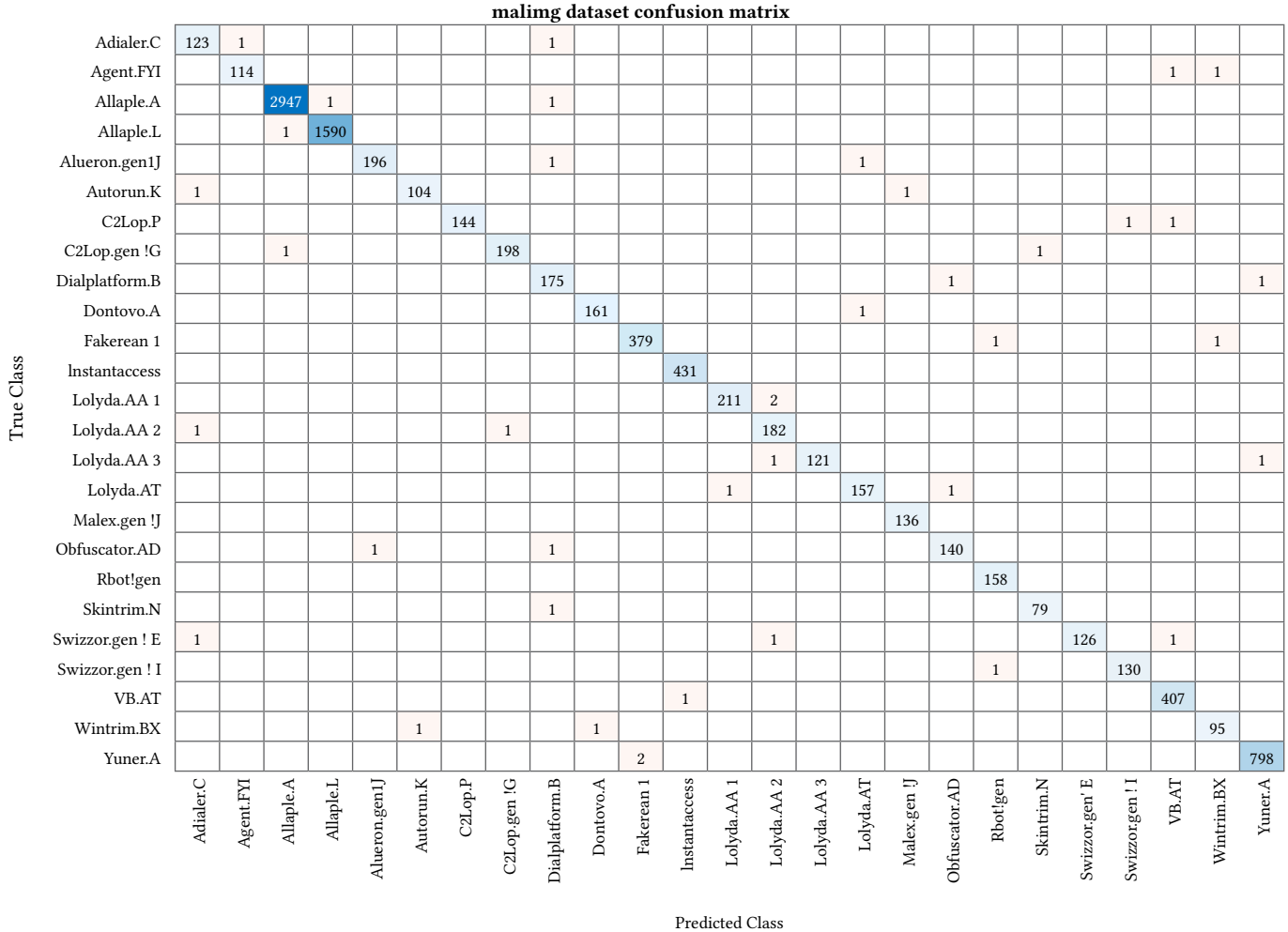


Fig. 7. Confusion Matrix Plot – Malimg data set.

TABLE VI. Proposed System Comparative Performance

Author	Dataset	Classifier	Accuracy (%)
J. Hemalatha et al., 2021 [99]	Malimg	Dense Net	98.23
V. Moussas et al., 2021 [100]	Malimg	Two level ANN	99.13
N. Marastoni et al., 2021 [101]	Malimg	CNN	98.5
M. Nisa et al., 2020 [102]	Malimg	CNN (AlexNet, Inception v3)	99.3
Ahmed Bensaoud et al, 2020 [103]	Malimg	Inception v3	99.24
Danish Vasan et al., 2020 [104]	Malimg	CNN	98.82
Hui Guo, et al., 2020 [105]	BIG 2015	ResNet50	99.73
Xianwei Gao et at., 2020 [106]	BIG2015	SSTL (Boost_ RNN)	96.9
S. A Roseline et al., 2020 [107]	BIG 2015	Deep random forest	97.2
Danish Vasan et al., 2020 [108]	ImageNet	CNN	98.82
Yuntao Zhao et al., 2020 [109]	BIG 2015	FRCNN with ImageNet	92.8
N. Bhodia et al., 2019[110]	malimg	DNN	94.8
Duc-Ly Vu et al.,2019 [111]	Author prepared dataset	Convolutional Transformation Network	99.14
Sung et al., 2021 [112]	BIG 2015	CNN	99.2
Proposed technique	**Kaggle**	**SemiE**	**99.11**
Proposed technique	**malimg**	**SemiE**	**99.03**

In this paper we have described a GRASE model which combines malware visualization, texture based GRanulometry (GRA) feature and Semi eager based classifier to classify malware images into different malware family classes.

System performance was evaluated using malimg and BIG-2015 Kaggle dataset. 'malimg' dataset is in the form of grayscale image, but BIG-2015 dataset required to be converted to gray scale images from the byte code of malware program. Each pixel in the gray scale malware image represents a code byte. Therefore, we applied a GRA technique where Granules' or pixels are the important input to compute features.

With GRA additional features like n-gram, MD1, MD2, entropy, OPCODE, Register, symbols, data define, and Sections were used for generating feature vectors. This kind of learning approach is more suited to MDS because both, real time learning can be implemented with less time, and also testing or generating output in the form of malware detection is desideratum.

SemiE classifier with image-based visualization of feature vector resulted in an enhanced performance for classifying nine classes of malware and offered an overall accuracy of 99.11% with the Kaggle dataset and 99.03 % accuracy was achieved with the malimg dataset (refer table VI).

Future scope for the proposed technique will focus on a diverse set of datasets to verify robustness of an algorithm. Presently a footprint of malware is available, but the objective of malware detection is customer security which should not be compromised. So, the task is to test a model if it can be used for predicting the new malware.

References

[1] G. Ekta, B. Divya, S. Sanjeev, "Malware Analysis and Classification: A Survey," Journal of Information Security, vol. 5, no. 2, pp. 56-64, 2014.

[2] M. Deore, U. Kulkarni, "MDFRCNN: Malware Detection using Faster Region Proposals Convolution Neural Network," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 7, no. 4, pp. 146-162, 2022.

[3] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber Scanning: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, Third Quarter, 2014.

[4] M. Dhalaria, E. Gandotra, "A Hybrid Approach for Android Malware Detection and Family Classification," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 6, no. 6, pp. 174-188, 2021.

[5] P. Kupidura, "Wykorzystanie granulometrii obrazowej w klasyfikacji treści zdjęć satelitarnych," Prace Naukowe Politechniki Warszawskiej. Geodezja, 2015.

[6] P. Kupidura, M. Skulimowska, "Morphological profile and granulometric maps in extraction of buildings in VHR satellite images," Archives of Photogrammetry, Cartography and Remote Sensing, pp. 83–96, 2015.

[7] P. Kupidura, P. Koza, J. Marciniak, "Morfologia Matematyczna w Teledetekcji," PWN: Warsaw, Poland, 2010.

[8] A. Haas, G. Matheron, J. Serra, "MorphologieMathématique et granulométriesen place. Ann. Des Mines," vol. 12, pp. 768–782, 1967.

[9] E.R. Dougherty, J.B. Pelz, F. Sand, A, Lent, "Morphological Image Segmentation by Local Granulometric Size Distributions," Journal of Electronic Imaging, vol. 1, pp. 46–60, 1992.

[10] A. Zakai and Y. Ritov, "Consistency and localizability," *Journal of Machine Learning Research (JMLR)*, vol. 10, pp. 827–856, 2019.

[11] N. Cao and W. Cui, Introduction to Text Visualization, Atlantis Press, Paris, 2016.

[12] D. Keim, "Information visualization and visual data mining," IEEE Transactions on Visualization and Computer Graphics, vol.8, no.1, pp.1–8, 2002.

[13] S. Few, "Information Dashboard Design - The Effective Visual Communication of Data, Sebastopol," CA: O'Reilly, 2006.

[14] I. Yoo, "Visualizing windows executable viruses using self-organizing maps," VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 82-89. 10.1145/1029208.1029222, 2004.

[15] S. Foresti, J. Agutter, Y. Livnat, S. Moon, and R. Erbacher, "Visual correlation of network alerts," IEEE Computer Graphics and Applications, vol. 26, no. 2, pp. 48–59,2006

[16] M. G. Schultz, E. Eskin, F. Zadok, S. J. Stolfo, "Data mining methods for detection of new malicious executables," in: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, Oakland, CA, USA, pp. 38-49, 2001.

[17] D. A. Quist and L. M. Liebrock, "Visualizing compiled executables for malware analysis," 6th International Workshop on Visualization for Cyber Security, Atlantic City, NJ, pp. 27-32, 2009.

[18] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 8, pp. 1313–1329, 2012.

[19] N. Diakopoulos, D. Elgesem, A. Salway, A. Zhang, and K. Hofland, "Compare clouds: visualizing text corpora to compare media frames," in Proceedings of IUI Workshop on Visual Text Analytics, 2015.

[20] P. Trinius, T. Holz, J. Göbel and F. C. Freiling, "Visual analysis of malware behavior using tree maps and thread graphs," 6th International Workshop on Visualization for Cyber Security, Atlantic City, NJ, pp. 33-38, 2009.

[21] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B, "Malware Images: Visualization and Automatic Classification," Proceedings of the

8th International Symposium on Visualization for Cyber Security, Article No. 4, 2011.

[22] Conti, G.; Bratus, S.; Shubina, A.; Lichtenberg, A.; Ragsdale, R.; Perez-Alemany, R.; angster, B.; Supan, M.A,"Visual Study of Binary Fragment Types," Black Hat: San Francisco, CA, USA, 2010.

[23] K. Kancherla and S. Mukkamala, "Image visualization-based malware detection," IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, pp. 40-44, 2013.

[24] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture,"Computer Networks, vol. 171, 107138, 2020.

[25] B.N. Narayanan, V.S.P Davuluru, "Ensemble Malware Classification System using Deep Neural Networks,"Electronics,vol. 9, no.5, 721, 2020.

[26] R. Tian, L. M. Batten, S. C. Versteeg, "Function length as a tool for malware classification," in: Malicious and Unwanted Software, MALWARE 2008. 3rd International Conference on, pp. 69-76,2008.

[27] Z. M. Fadli, A. Jantan, "An approach for malware behavior identification and classification," Computer Research and Development (ICCRD) 2011 3rd International Conference on, vol. 1, 2011.

[28] M. Shankarapani, S. Ramamoorthy, R Movva, S. Mukkamala, "Malware detection using assembly and api call sequences," Journal of Computing Virology, vol. 7, pp. 107-119, 2011.

[29] D. Kong, G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in: KDD '13: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, pp. 1357-1365, 2013.

[30] I. Santos, J. Devesa, F. Brezo, J. Nieves, P.G. Bringas, "OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection," In: Herrero, Á., et al. International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions. Advances in Intelligent Systems and Computing, vol 189. Springer, Berlin, Heidelberg, 2013.

[31] K. Han, J. H. Lim, and E. G. Im, "Malware analysis method using visualization of binary files," in Proceedings of the the2013 Research in Adaptive and Convergent Systems, pp. 317–321, Montreal, Quebec, Canada,2013.

[32] J. Jacobs and B. Rudis, "Data-driven security analysis, visualization, and dashboards," in Indianapolis, John Wiley & Sons, 2014.

[33] N. Cao, L. Lu, Y.-R. Lin, F. Wang, and Z. Wen, "Social Helix: visual analysis of sentiment divergence in social media," Journal of Visualization, vol.18, no.2, pp. 221–235, 2015.

[34] Dübel, Steve &Röhlig, Martin & Schumann, H. & Trapp, Matthias, "2D and 3D presentation of spatial data: A systematic review," 2014 IEEE VIS International Workshop on 3DVis (3DVis), Paris, France, 2014, pp. 11-18, doi: 10.1109/3DVis.2014.7160094.

[35] T. Songqing, "Imbalanced Malware Images Classification: a CNN based Approach," CoRR abs/1708.08042, 2017.

[36] W. B. Balakrishnan, "Security Data Visualization," SANS Institute Inc, 2014.

[37] M. Arefkhani and M. Soryani, "Malware clustering using image processing hashes," 9th Iranian Conference on Machine Vision and Image Processing (MVIP), Tehran, 2015, pp. 214-218, 2015.

[38] Q. Wu, Z. Qin, J. Zhang, H. Yin, G. Yang, K. Hu, "Android Malware Detection Using Local Binary Pattern and Principal Component Analysis," In: Zou B., Li M., Wang H., Song X., Xie W., Lu Z. (eds) Data Science. ICPCSEE 2017. Communications in Computer and Information Science, vol. 727, Springer, Singapore, 2017.

[39] S. Rezaei, A. Afraz, F. Rezaei, M. R. Shamani, "Malware detection using opcodes statistical features," in 2016 8th International Symposium on Telecommunications (IST), pp. 151–155, 2016.

[40] V. Sitalakshmi and M. Alazab, "Use of Data Visualization for Zero-Day Malware Detection,"Security and Communication Networks, vol. 2018, pp. 1728303:1-1728303:13, 2018.

[41] T. Y. Zhang, X. M. Wang Li, Z. Z. Li, F. Guo, Y. Ma, and W. Chen, "Survey of network anomaly visualization," Science China Information Sciences, vol. 60, no. 12, 121101, 2017.

[42] W. Shanks, "Enhancing Intrusion Analysis through Data Visualization," SANS Institute, Inc, 2015.

[43] L. Shiqi, T. Shengwei, S. Hua, and Y. Long, "Research on malicious code classification algorithm of stacked auto encoder," Application Research of Computers, vol. 35, no. 1, pp. 261–265, 2018.

[44] Y. Liu, Z. Wang, Y. Hou, and H. Yan, "Visualization and automatic classification of malicious code with enhanced information density,'' *J. Tsinghua Univ. (Natural Sci. Ed.)*, vol. 59, no. 1, pp. 914, 2019.

[45] T. Wuchner, M. Ochoa, A. Pretschner, "Robust and effective malware detection through quantitative data flow graph metrics," in: Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, pp. 98-118, 2015.

[46] R.W. Conners, C.A. Harlow, "A Theoretical Comparison of Texture Algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 3, pp. 204-222, 1980.

[47] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 674-693, 1989.

[48] Lehto P, Romanoff J, Remes H, Sarikka T. "Characterisation of local grain size variation of welded structural steel," Weld World, vol. 60, pp. 673 678, 2016, http://dx.doi.org/10.1007/s40194-016-0318-8

[49] S. Srakaew, W. Piyanuntcharatsr, S. Adulkasem, "On the comparison of malware detection methods using data mining with two feature sets," Journal of Security and Its Applications, vol. 9, no. 3, pp. 293-318, 2015.

[50] D. Uppal, R. Sinha, V. Mehra, V. Jain, "Malware detection and classification based on extraction of api sequences," in: ICACCI, IEEE, pp. 2337-2342, 2014.

[51] R. Islam, R. Tian, L. M. Batten, S. Versteeg, "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications, vol. 36, no.2, pp. 646-656, 2013.

[52] S. Nari, A. A. Ghorbani, "Automated malware classification based on network behavior," in: Computing, Networking and Communications (ICNC), 2013 International Conference on, IEEE, pp. 642-647, 2013.

[53] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," Information Sciences, vol. 231, pp. 64-82, 2013.

[54] I. Firdausi, C. Lim, A. Erwin, A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in: ACT '10, IEEE, pp. 201-203, 2010.

[55] F. Ahmed, H. Hameed, M. Z. Shafiq, M. Farooq, "Using spatio-temporal information in api calls with machine learning algorithms for malware detection," in: Proceedings of the 2nd ACM workshop on Security and artificial intelligence, ACM, pp. 55-62, 2009.

[56] J. Z. Kolter, M. A. Maloof, "Learning to detect and classify malicious executables in the wild," Journal of Machine Learning Research, pp. 2721-2744, no. 7, pp. 2721-2744, 2006.

[57] M. Ahmadi, G. Giacinto, D. Ulyanov, S. Semenov, M. Tromov, "Novel feature extraction, selection and fusion for effective malware family classification," CoRR abs/1511.04317, 2016

[58] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, T. Dumitras, "The dropper effect: Insights into malware distribution with downloader graph analytics," in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, pp. 1118-1129, 2015.

[59] W. Mao, Z. Cai, D. Towsley, X. Guan, "Probabilistic inference on integrity for access behavior-based malware detection," in: International Workshop on Recent Advances in Intrusion Detection, Springer, pp. 155-176, 2015.

[60] P. M. Comar, L. Liu, S. Saha, P. N. Tan, A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in: INFOCOM, 2013 Proceedings IEEE, pp. 2022-2030, 2013.

[61] M. Siddiqui, M. C. Wang, J. Lee, "Detecting internet worms using data mining techniques," Journal of Systemic, Cybernetics and Informatics, vol. 6, no. 6, pp. 48-53, 2009.

[62] M. Graziano, D. Canali, L. Bilge, A. Lanzi, D. Balzarotti, "Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence," in: USENIX Security '15, pp. 1057-1072, 2015.

[63] J. Sexton, C. Storlie, B. Anderson, "Subroutine based detection of APT malware," Journal of Computer Virology and Hacking Techniques, vol. 12, pp. 225-233, 2016.

[64] G. E. Dahl, J. W. Stokes, L. Deng, D. Yu, "Large-scale malware classification using random projections and neural networks," in: Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3422-3426, 2013.

[65] S. Palahan, D. Babi_c, S. Chaudhuri, D. Kifer, "Extraction of statistically significant malware behaviors," in: Computer Security Applications Conference, ACM, pp. 69-78, 2013.

[66] E. Raff, C. Nicholas, "An alternative to ncd for large sequences," lempel-zivjaccard distance, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 1007-1015, 2017.

[67] A. Mohaisen, O. Alrawi, M. Mohaisen, "Amal: High-fidelity, "behavior based automated malware analysis and classification," Computers & Security, vol. 52, pp. 251-266, 2015.

[68] D. Kong, G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in: ACM SIGKDD '13, nKDD '13, ACM, New York, NY, USA, pp. 1357-1365, 2013.

[69] T. Lee, J. J. Mody, "Behavioral classification," in: EICAR Conference, pp. 1-17, 2006.

[70] K. Huang, Y. Ye, Q. Jiang, "Ismcs: an intelligent instruction sequence-based malware categorization system," in: Anti-counterfeiting, Security, and Identification in Communication, 2009, IEEE, pp. 509-512, 2009.

[71] Y. Ye, T. Li, Y. Chen, Q. Jiang, "Automatic malware categorization using cluster ensemble," in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 95- 104, 2010.

[72] M. Eskandari, Z. Khorshidpour, S. Hashemi, "Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection," Journal of Computer Virology and Hacking Techniques, vol. 9, pp. 77-93, 2013.

[73] Z. Chen, M. Roussopoulos, Z. Liang, Y. Zhang, Z. Chen, A. Delis, "Malware characteristics and threats on the internet ecosystem," Journal of Systems and Software, vol. 85, pp. 1650-1672, 2012.

[74] J. Yonts, "Attributes of malicious files," Tech. rep., The SANS Institute, 2012.

[75] J. Upchurch, X. Zhou, "Variant: a malware similarity testing framework," in: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, pp. 31-39,2015.

[76] A. Tamersoy, K. Roundy, D. H. Chau, "Guilt by association: large scale malware detection by mining file-relation graphs," in: Proceedings of the 20th ACM SIGKDD, ACM, pp. 1524-1533, 2014.

[77] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, Scalable, "behavior-based malware clustering," in: NDSS, Vol. 9, pp. 8-11, 2009.

[78] M. Polino, A. Scorti, F. Maggi, S. Zanero, "Jackdaw: Towards Automatic Reverse Engineering of Large Datasets of Binaries," in: Detection of Intrusions and Malware, and Vulnerability Assessment, Lecture Notes in Computer Science, Springer International Publishing, pp. 121-143, 2015.

[79] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, M. Antonakakis, "Measuring and detecting malware downloads in live network traffic," in: Computer Security ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 556-573, 2013.

[80] J. Jang, D. Brumley, S. Venkataraman, "Bitshred: feature hashing malware for scalable triage and semantic analysis," in: Computer and communications security, ACM, pp. 309-320, 2011.

[81] K. Rieck, P. Trinius, C. Willems, T. Holz, "Automatic analysis of malware behavior using machine learning," Journal of Computer Security, vol. 19, no. 4, pp. 639-668, 2011.

[82] S. Attaluri, S. McGhee, M. Stamp, "Profile hidden Markova models and metamorphic virus detection," Journal in Computer Virology, vol. 5, pp. 151-169, 2009.

[83] G. Liang, J. Pang, C. Dai, "A behavior-based malware variant classification technique," International Journal of Information and Education Technology, vol. 6, no. 4., pp. 291-295, 2016.

[84] Z. Feng, S. Xiong, D. Cao, X. Deng, X. Wang, Y. Yang, X. Zhou, Y. Huang, G. Wu, "Hrs.: A hybrid framework for malware detection," in: Proceedings of the 2015 ACM International Workshop on Security and Privacy Analytics, ACM, pp. 19-26, 2015.

[85] M. Ghiasi, A. Sami, Z. Salehi, "Dynamic VSA: a framework for malware detection based on register contents," Engineering Applications of Artificial Intelligence, vol. 44, pp. 111- 122, 2015.

[86] M. Lindorfer, C. Kolbitsch, P. M. Comparetti, "Detecting environment sensitive malware," in: Recent Advances in Intrusion Detection, Springer, pp. 338-357, 2011.

[87] C.T. Lin, N.J. Wang, H. Xiao, C. Eckert, "Feature selection and extraction for malware classification," Journal of Information Science and Engineering, vol. 31, no. 3, pp. 965-992, 2015.

[88] B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, "Graph-based malware detection using dynamic analysis," Journal in Computer Virology, vol. 7, no. 4, pp. 247-258, 2011.

[89] I. Santos, J. Nieves, P. G. Bringas, "Ch. Semi-supervised Learning for Unknown Malware Detection," International Symposium on Distributed Computing and Artificial Intelligence, Springer Berlin Heidelberg Berlin, Heidelberg, pp. 415-422, 2011.

[90] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, C. Faloutsos, "Polonium: Tera-scale graph mining for malware detection," in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 131-142, 2010.

[91] B. Anderson, C. Storlie, T. Lane, "Improving malware classification: bridging the static/dynamic gap," in: Proceedings of the 5th ACM workshop on Security and artificial intelligence, ACM, pp. 3-14, 2012.

[92] D. Baysa, R. Low, and M. "Stamp. Structural entropy and metamorphic malware," Journal of Computer Virology and Hacking Techniques, vol. 9, no. 4, pp. 179–192, 2013.

[93] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," IEEE Security and Privacy, vol. 5, no. 2, pp. 40–45, 2007.

[94] A. Moser, C. Kruegel, and E. Kirda "Limits of static analysis for malware detection," In Computer Security Applications Conference, 2007. ACSAC 2007, Twenty-Third Annual, pp. 421–430, 2007.

[95] D. Bilar. "Statistical structures: Finger printing malware for classification and analysis," InBlackhat, 2006.

[96] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. "Evasion attacks against machine learning at test time," in H. Blockeel, K. Kersting, S. Nijssen, and F. Železný (editors), Machine Learning and Knowledge Discovery in Databases, vol. 8190 of Lecture Notes in Computer Science, pp. 387–402, Springer Berlin Heidelberg, 2013.

[97] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. "Semantics-aware malware detection," In Security and Privacy, 2005 IEEE Symposium on, pp. 32–46, 2005.

[98] S.Z.M. Shaid, M.A. Maarof, "Malware behaviour visualization," Jurnal Teknologi, vol. 70, no. 5, 2014.

[99] J. Hemalatha, S.A. Roseline, S. Geetha, S. Kadry, R. Damaševičius, "An Efficient DenseNet-Based Deep Learning Model for Malware Detection," Entropy, vol. 23, no. 3, 344, 2021.

[100] V. Moussas, A. Andreatos, "Malware Detection Based on Code Visualization and Two-Level Classification," Information, vol. 12, no. 3, 118, 2021, https://doi.org/10.3390/info12030118

[101] N. Marastoni, R. Giacobazzi, M. Dalla Reda, "Data augmentation and transfer learning to classify malware images in a deep learning context," Journal of Computer Virology Hacking Techniques, vol. 17, no. 279-297, 2021.

[102] M. Nisa, J.H. Shah, S. Kanwal, M. Raza, M.A. Khan, R. Damaševičius, T. Blažauskas, "Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features," Applied Sciences, vol. 10, no. 14, 4966, 2020.

[103] A. Bensaoud, N. Abudawaood, J. Kalita, "Classifying Malware Images with Convolutional Neural Network Models," ArXiv, abs/2010.16108, 2020.

[104] D. Vasan, M. Alazab, S. Assan, H. Naeem, B. Safaei, Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," Computer Networks, Vol. 171, 107138, 2020.

[105] H. Guo, S. Huang, C. Huang, F. Shi, M. Zhang, Z. Pan, "Binary File's Visualization and Entropy Features Analysis Combined with Multiple Deep Learning Networks for Malware Classification," Security and Communication Networks, vol. 20, 8881760, 2020.

[106] X. Gao, C. Hu, C. Shan, B. Liu, Z. Niu, H. Xie, "Malware classification for the cloud via semi-supervised transfer learning," Journal of Information Security and Applications, vol. 55, 102661, 2020.

[107] S. A. Roseline, S. Geetha, S. Kadry and Y. Nam, "Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm," in IEEE Access, vol. 8, pp. 206303-206324, 2020.

[108] D. Vasan, M. Alazab, S. Wassan, B. Safaei, Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," Computers & Security, Vol. 92, 101748, 2020.

[109] Y. Zhao, W. Cui, S. Geng, B. Bo, Y. Feng and W. Zhang, "A Malware

Detection Method of Code Texture Visualization Based on an Improved Faster RCNN Combining Transfer Learning," in IEEE Access, vol. 8, pp. 166630-166641, 2020.

[110] N. Bhodia, P. Prajapati, F. Di Troia, M. Stamp, "Transfer Learning for Image-Based Malware Classification," doi: 10.5220/0007701407190726, 2019.

[111] D. -L. Vu, T. -K. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Massacci and P. H. Phung, "A Convolutional Transformation Network for Malware Classification," 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, 2019, pp. 234-239.

[112] K.-S. Sung, W. Na, "A study on the implementation of a system providing reliable malware information service," International Journal of Electrical Engineering & Education, vol. 58, no. 2, pp. 517-530, 2021.

Mahendra Deore

M. Deoreis working as an Asst. Professor in Computer Engineering Department atMKSSS's Cummins College of Engineering for Women, Pune 411051, India. He was awarded his Master of Technology Degree from Bharati Vidyapeeth Deemed University College of Engineering, Dhankawadi, Pune. He received doctoral degree from Swami Ramanand Teertha Marathwada University, Nanded, Indiain 20022. His areas of interest are big data, Security, Computer Networks and Machine learning. He has Fourteen years' experience in teaching.

Manoj Tarambale

He received graduate degree (B.E.) in Electrical Engineering from University of Pune (SPPU), India in 1992, post graduate degree (M.E.) in Control System from Shivaji University, Kolhapur, India in 2002 and completed research work (PhD) in Electrical – Biomedical Image Processing from PACIFIC University, Udaipur, India in 2018. He has one-year industrial experience and thirty years teaching experience. At present, he is Associate Professor of electrical engineering department and Principal of PVG's College of Engineering and Technology & G K Pate Institute of Management, Pune-09, India. His main research interests are control system engineering, electrical vehicle technology, robotics & automation, bio-medical image processing, electronic instrumentation, and medical diagnosis (AI, ML & DS based).

Jambi Ratna Raja Kumar

Prof (Dr) Ratna Raja Kumar Jambi Completed his PhD (CSE) degree from Maharishi University of Information Technology, Lucknow, Uttar Pradesh in 2019, Master of Technology (CSE) from Pondicherry Central University in 2007. He has 17 years of Teaching and Research Work. He is having Patents at National, international level and has published Papers in artificial Intelligence and Machine Learning. He has received the award as "Innovative Leader" form World Education Summit & Awards in 2019 at New Delhi.

Sachin Sakhare

Dr. Sachin R. Sakhare is working as a Professor and Head of the Computer Engineering Department at Vishwakarma Institute of Information Technology, Pune, India. He has 27 Years of experience in engineering education. He is recognized as PhD guide by Savitribai Phule Pune University and currently guiding 8 PhD scholars. He is a life member of CSI, ISTE and IAEngg. He has Published 51 research communications in national, international journals and conferences, with around 393 citations and H-index 7. He has authored 6 books which is published by Springer Nature, CRC Press and IGI Global. He worked as a reviewer of journals published by Elsevier, Wiley, Hindawi, Springer, Inder science, and IETE. He worked as a reviewer for various conferences organized by IEEE, Springer, and ACM. He worked as a member of the Technical and Advisory Committees for various international conferences. Dr. Sachin has Delivered invited talks at various Conferences, FDP's and STTP's as well as to PG and PhD students. He has guided 26 PG students. He has filed and published 07 patents out of which 01 Indian, 03 Australian and 02 south African patents are granted.