Journal of Information Security and Applications 47 (2019) 377–389

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

A hybrid deep learning image-based analysis for effective malware detection

Sitalakshmi Venkatraman[a,*], Mamoun Alazab[b], R. Vinayakumar[c]

[a] Department of IT, Melbourne Polytechnic, Prahran Campus, VIC 3181, Australia
[b] Charles Darwin University, Darwin, NT 0810, Australia
[c] Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India

ARTICLE INFO

Article history:
Available online 24 June 2019

Keywords:
Malware detection
Similarity mining
Image analysis
Evaluation metrics
Machine learning
Deep learning architectures

ABSTRACT

The explosive growth of Internet and the recent increasing trends in automation using intelligent applications have provided a veritable playground for malicious software (malware) attackers. With a variety of devices connected seamlessly via the Internet and large amounts of data collected, the escalating malware attacks and security risks are a big concern. While a number of malware detection methods are available, new methods are required to match with the scale and complexity of such a data-intensive environment. We propose a novel and unified hybrid deep learning and visualization approach for an effective detection of malware. The aim of the paper is two-fold: 1. to present the use of image-based techniques for detecting suspicious behavior of systems, and 2. to propose and investigate the application of hybrid image-based approaches with deep learning architectures for an effective malware classification. The performance is measured by employing various similarity measures of malware behavior patterns as well as cost-sensitive deep learning architectures. The scalability is benchmarked by testing our proposed hybrid approach with both public and privately collected large malware datasets that show high accuracy of our malware classifiers.

1. Introduction

We are increasingly facing security breaches as we progress into the fourth industrial revolution (Industry 4.0), which is driven by advancements in Internet technologies and intelligent automation. It is important to develop defense systems against malicious software (malware) attacks that can disrupt businesses by penetrating their computing systems, data and applications without acquiring any user permission and authentication [1,2]. The term 'malware' covers a wide range of malicious code such as a computer virus, a worm and potentially unwanted programs (PUP) that can lead to a denial of service attack. Today, several malicious attacks are caused by unknown variants of existing malware, that obfuscate their behavior to evade from detection [3]. With the huge variety of open-source software and seamless installation of applications via the Internet on the rise, it is a major challenge to effectively detect such new obfuscated malwares that are being stealthily generated in largescale. Therefore, to combat this malware warfare, malware detection solutions are evolving in the field of cyber security and this paper takes a modest step forward in this research direction.

For several decades, anti-virus software solutions form the most commonly used commercial systems for the detection and mitigation of malware. Traditionally, such solutions were based on virus signatures that required human intervention to supervise and update the signature database when new malwares were encountered. More recently, machine learning systems are being developed which have the capability to alleviate the limitations of signature-based systems [4–6]. Such self-learning systems are capable of employing data mining and deep learning methods which can facilitate the learning of complex virus patterns to distinguish between the benign and malware binaries. However, different self-learning approaches exhibit degrees of variability in their capability to detect variants of existing malware or even an entirely new malware. Several research studies are being conducted to explore the suitability of different approaches of self-learning techniques for malware detection, and to compare the scalability and performance of such models with a variety of datasets [7,8]. It is important to first understand the fundamental approaches employed as well as the malware obfuscation techniques adopted by hackers in order to propose innovative solution models to match with this cyber security problem.

The most commonly applied malware detection approach falls under two main techniques: static and dynamic analysis [4–8].

* Corresponding author.
E-mail address: sitavenkat@melbournepolytechnic.edu.au (S. Venkatraman).

https://doi.org/10.1016/j.jisa.2019.06.006
2214-2126/

Static analysis uses the syntax and structural properties of a file by disassembling the program binary in order to extract the features. On the other hand, dynamic analysis of the file is required to be conducted during its running time in order to extract characteristic actions performed by the program. Previous studies have also combined static and dynamic approaches [9,10]. Many commercial systems of today have started to use a hybrid of static and dynamic analysis for malware detection. However, new approaches are required in order to improve the technique of detecting different obfuscations of malware that are being increasingly launched by the hackers and recent studies have compared static, dynamic and hybrid approaches arriving at various implications in the current paradigm [4].

It has become a common practice of malware writers to adopt several obfuscation techniques using metamorphic and polymorphic methods to generate variants of an existing malware family so as to evade detection [3,4]. In addition, the entire program binary could be obfuscated using packing methods to ensure that the code can only be analyzed at runtime [11]. Reverse engineering of such non-standard and custom–made packing is labor-intensive and requires the binary to be executed in a virtual environment for unpacking [12]. Hence, intelligent approaches such as machine learning that are capable of incorporating self-learning traits of a human expert are being developed.

Machine learning models for static, dynamic and hybrid analysis of malware have been investigated showing promising results to detect obfuscated malware and their implications have been studied, motivating further research in this direction [4,13–16]. A number of static malware detection approaches have differentiated their work by exploring different classifiers such as Support Vector Machine (SVM), Hidden Markov Models (HMM) k-Nearest Neighbor (KNN) and Naïve Bayes (NB) [17,18]. In general, for techniques based on dynamic analysis, various traces of the behavior patterns of malware are analyzed by executing it. In literature, two commonly used approaches for dynamic analysis are control flow analysis and API call analysis [3,6,13]. Overall, several feature-based approaches, including high-level API calls as well as low-level opcodes for n-grams based malware detection, have been explored in previous studies [18–20]. Semi-automated data analysis methods require the malware analysts to analyse and interpret intermediate results that can be time-consuming and hence self-learning and intelligent frameworks are proposed [21,22]. In this work, we have adopted machine learning and similarity mining approaches that have been effectively applied to both static and dynamic malware detection with image-based deep learning method as the focus.

Image-based techniques could provide sophisticated visual aids in detecting suspicious unknown malware in order to alert anomalous behavior patterns quickly. Visual representations of malware patterns have the advantage of providing a summarized picture of possible attacks. Visual analytics along with analytical reasoning of human experts could speed-up the malware detection process [23,24]. Similarity mining is a machine learning method based on the analysis of similarities of the distance measures. In visual analytics, similarity mining has been recently adopted to detect malware. This work has advanced further from previous studies by using image-based analysis for malware detection. In this paper, we propose a hybrid model by employing similarity mining and deep learning architectures for accurately detecting and classifying obfuscated malware into their malware families . An image comparison of different malware families as well as benign datasets are used to visually demonstrate the significant difference in the behavior patterns of the malware families. Further, deep learning techniques are employed to facilitate self-learning so as to achieve high accuracy in our proposed classifiers.

Overall, the main contributions of our proposed model are:

1. developing a hybrid deep learning model for malware detection and classification by employing image-based machine learning techniques that are computationally cost-effective and scalable.
2. conducting an experimental study by performing an in-depth analysis of various classical machine learning and deep learning techniques on different public and private datasets with a purpose to evaluate our proposed architectures in terms of their efficacy in dealing with large datasets of new malware families.

We have organized the overall structure of the paper as follows. Section 2 provides related work of image-based analysis in the area of computer security. The proposed hybrid model adopted for this study is presented in Section 3. Section 4 describes the experimental setup and the datasets used for the study. The results of the performance evaluation of our proposed model indicating high classification accuracy achieved using machine learning and deep learning architectures are presented in Section 5. Finally, we provide our conclusions, highlighting the limitations of the study and future research work in Section 6.

2. Related work

Many image-based analysis using similarity of patterns fall under two main categories: (1) projection-oriented or (2) semantic-oriented [25]. In the field of computer security, visualization tools have evolved over a period of time and they are becoming more useful for processing massive data with large files. Two-dimensional visualization of a similarity matrix is a traditional technique used to capture the relevant similarity measures between objects [26,27]. It provides three key properties: (i) once the similarity space is formed, the high-dimensionality of the data does not affect further processing; (ii) clusters of equal importance get formed, and (iii) clusters that are related to one another are shown adjacent to each other aiding in visualization of results [25]. It is a common practice to visually represent documents as points on either a 2D or 3D plane. The distance between each pair of points show how similar the two documents are, i.e., the closer they are, the more similar the two document contents are [28,29].

Recently, research studies have employed image-based analysis of network security attacks [30]. Some semi-automated techniques involved the visual investigation of secure shell (SSH) brute force attempts that were identified by different colors for the various anomalies detected along with the details of UserIDs and Internet Protocol (IP) addresses [31]. Visualization techniques were also employed to display an overview of large packets at a time. Such images show the relationships between network packets which helped security analysts to zoom into further details [32]. Another study used image-based analysis to explain the chronology of a malware attack such as a spear phishing attack with colors indicating which type of connections to the system were successful [33]. Fig. 1 shows the information on ‘what’, ‘where’ and ‘when’ of these connections and how the distances to other hosts could be estimated using their IP addresses [34,35]. Various types of alerts are shown as separate sectors of concentric rings in consecutive time intervals and the different possible attacks to the same host are depicted with color-coded connections. Most of these visual techniques had adopted text-based analysis of network parameters and logs to arrive at image representations with different colors for the semi-automated analysis and detection of malware attacks. Such research studies are quite restricted with a focus on network traffic and infiltration analysis [36,37]. Furthermore, these are time consuming in today’s world of Big Data, and various approaches based on image texture representations and their analysis are being explored [38].

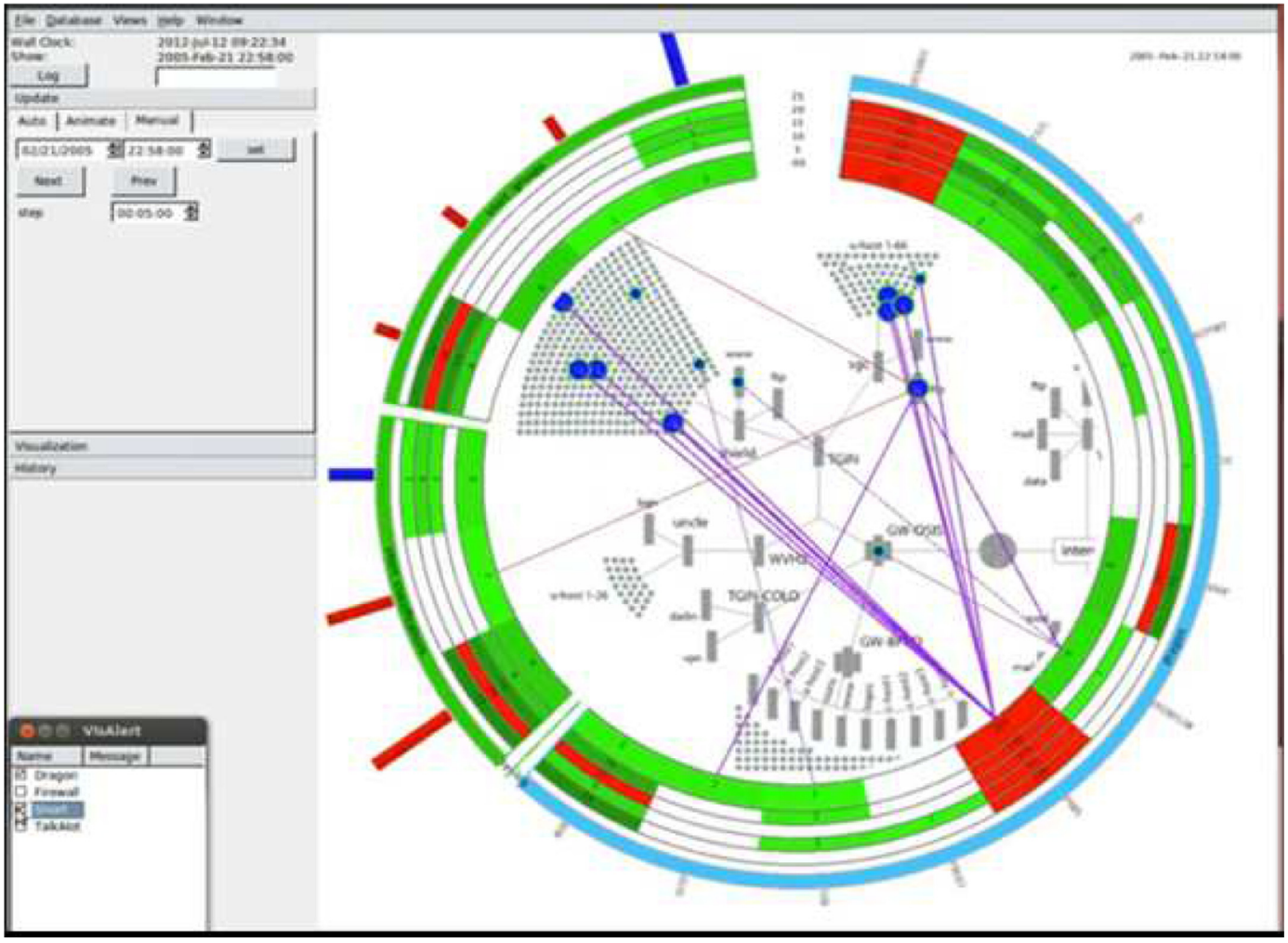


Fig. 1. Visualisation of network connection types (what), resources (where) attacked, and the time (when).

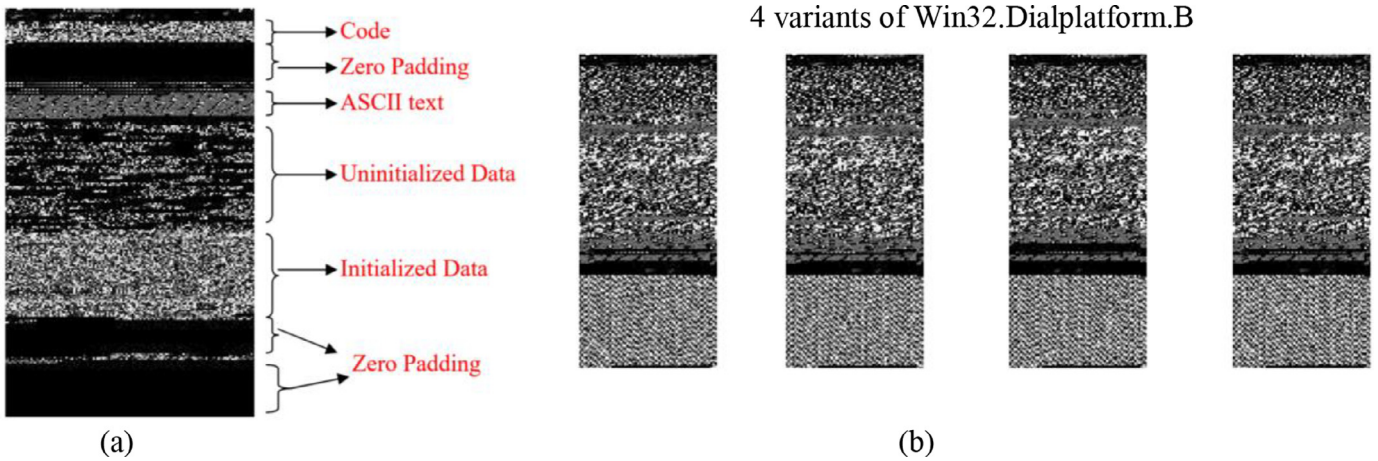


Fig. 2. Images of variants of a malware family showing similarity of binary sections.

Different sections of the executable binary are represented as unique image textures for a quicker analysis that can improve the productivity of a malware analyst. The advantage of such a texture-based image analysis is that they can give more information about the structure of the malware, displaying even small code changes while retaining the whole structure of the code. Fig. 2(a) shows different sections of the binary code displaying unique texture, which is useful in identifying similar patterns [38]. For instance, Fig. 2(b) shows similarity in the images of four malware variants belonging to the same malware family called Dialplatform.B.

The main limitation with texture-based image analysis of malware is that certain malware obfuscations cannot be easily analyzed. Malware could be packed using different packing methods and with different resolution. Hence, there is a need for an

improved solution for classifying malware using image-based analysis [39]. Recently, image features called gist descriptors have been analyzed in order to classify obfuscated malware and these techniques have been compared with deep learning techniques to evaluate their robustness [40]. Several such innovative techniques are being explored by researchers in order to address the major challenge of obfuscated malware detection.

In this paper, we propose a hybrid model of various supervised and unsupervised techniques to perform image analysis for detecting and classifying unknown malwares efficiently. By employing a variety of large public and private datasets, we conduct experimental studies to demonstrate how similarity mining combined with deep learning architectures could be employed effectively in the proposed classifier. Our proposed model is described in the next section.

3. Proposed hybrid model

Deep learning is a type of machine learning which has been used in various applications in different domains. This is primarily due to the reason that these methods have the capability to learn optimal feature representation implicitly by taking raw input samples. Recently, the applications of deep learning architectures are employed for malware detection. In image-based deep learning approaches, the malware binaries are converted into grayscale image representation and deep learning architectures are employed to learn the complex features (image patterns) [40–45]. Most commonly employed deep learning architectures are convolutional neural network (CNN) and long short-term memory (LSTM). The fundamental difference between these two methods is that CNN is capable of extracting spatial features, while LSTM is capable of learning the sequence information. Both CNN and LSTM can be combined to effectively learn both spatial and sequence information. A recent study used TensorFlow for deep learning experiments relying on transfer learning and were able to attain a testing accuracy of more than 98% [40]. In another work, a CNN based method was proposed for malware classification to handle the class imbalance problem achieving good results as compared to the existing well-known methods based on classical machine learning algorithms [41]. Another work had proposed a novel method to convert a bytes file into image representation and had employed a deep learning architecture for classification [42]. To avoid the class imbalance problem, their experiments had adopted random sampling as well as class rebalancing sampling methods. The class rebalancing methods had performed better than the existing classical machine learning based methods with highest F1-score. Some researchers proposed a malware classification framework by employing SimHash to convert the disassembled malware codes into grayscale images, and CNN for classification [43,44]. Another work had proposed static analysis of opcodes with malware classification performed based on the combination of RNN and CNN [45]. A recent study adopted cost-sensitive LSTM to handle class imbalance and the multi-class classification problem and the results show a better performance than the cost-insensitive LSTM [46].

This work has advanced further from a previous work that proposed a hybrid deep learning-based framework for intrusion detection [47]. In the previous work, an optimal machine learning model was developed by conducting several experiments using various publicly available datasets. The results showed that the method had scaled well to handle large amount of network events using both image analysis and deep learning approaches to effectively detect and classify attacks in a real-time environment. This work employs a similar methodology with a distinct difference in the focus of classifying variants of malware into their malware families using deep learning image-based analysis. The proposed hybrid architecture using a combination of supervised and unsupervised learning models for image-based malware classification is shown in Fig. 3.

The proposed architecture uses self-learning system which is capable of detecting not only the known malware and the variants of known malware, but also unknown malware. More importantly, the self-learning system uses classical machine learning and contemporary deep learning models that capture the complex features of binaries which can be best utilized to distinguish between the benign and malware binaries. The architecture composed of three different subsystems in which one subsystem is based on unsupervised learning model and the other two are based on supervised learning models. Each pre-processing stage shown in Fig. 3 is used to convert the binary files into different feature representations in order to employ machine learning and deep learning models. To enhance the malware detection of obfuscated malware, this type of hybrid system can be used in real-time systems. All these subsystems use both classical and advanced machine learning models

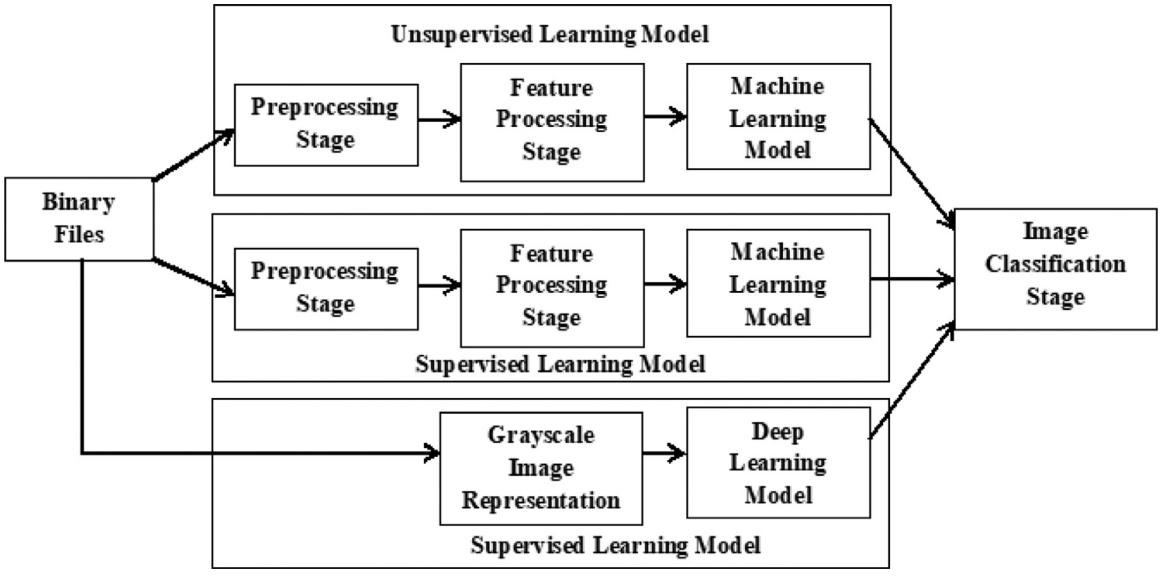


Fig. 3. Proposed Architecture.

with image-based analysis to effectively learn the complex system behaviors of malware binaries.

The proposed deep learning architecture for malware classification uses CNN and bi-directional pipeline. Miscalculation costs are included and in general, a malware family which contains more number of malware samples includes less cost, whereas a malware family which contains less number of samples includes higher cost. In the beginning, the cost matrix is unknown and genetic algorithms are employed to find out an optimal cost matrix. However, this method is time-consuming and leads to a high computational cost. Thus, to select an optimal value for each sample, let us assume that at least one category sample has equal cost. Let $c[i, i]$ denote the misclassification cost of the class i, which is generated using the class distribution as defined in Eq. (1) given below:

$$c[i,i] = \left(\frac{1}{n_i}\right)\left(\frac{1}{n_i}\right)^{\gamma} \tag{1}$$

where $\gamma \in [0, 1]$ is a trade-off parameter. When $\gamma = 1$, $c[i, i]$ is inversely proportional to the class size n_i and when $\gamma = 0$, the cost-sensitive LSTM reduces to the original LSTM, which is cost-insensitive. To find an optimal value for γ, we run 3 runs for the proposed architecture with values in the range 0 to 1. The proposed architecture performed well with $\gamma = 0.2$.

Convolutional neural network (CNN) is an improved model of classical neural network which has performed well in various long-standing artificial intelligence tasks in the field of computer vision [39]. Primarily, a CNN is composed of three different layers, convolutional, pooling and fully connected. The convolution layer contains a convolution operation that uses filters to slide over the image to capture features. By using Rectified Linear Units (ReLU), these features are mapped into non-linear space called feature maps. Since the dimension of the feature map can be very large, we use pooling to reduce the dimension. The most commonly used pooling operations are the max, min and average functions, and we employ max-pooling. The pooling features are passed into bidirectional Gated Recurrent Unit (GRU) which learns the sequential information of the byte sequences, and finally, the features are passed into the fully connected layer. We use a non-linear activation function for classification, and the most commonly used functions are sigmoid for binary and softmax for multi-class classification. To reduce the loss during training, we used categorical cross entropy, which is defined mathematically in Eq. (2) given below:

$$loss(p, e) = \sum_{x} p(x) \log\left(e(x)\right) \tag{2}$$

where x denotes an input, e and p denoting true probability distribution and predicted probability distribution respectively.

Image-based techniques make use of visual images of either binary data or behavior logs of the malware samples [48,49]. Feature-based techniques compare different malware samples based on extracted features [15]. Previous studies have considered a combination of both image-based and feature-based techniques for malware classification without execution or disassembly of malware code (e.g. Fig. 2) [38,50]. However, due to their limitations in operating with only selected file formats and packing methods, in this work, we propose new image-based analysis techniques to include similarity mining of behavior patterns of malware. In another related work, opcode sequences are converted into RGB pixels in an image matrix and the similarity of image matrices are computed [51]. Our approach is different in two ways based on enhancements from previous work [18–20,24,47]. Firstly, we make use of large datasets of about 52,000 malware samples collected privately in previous work as well as commonly used public datasets for benchmarking our study. Secondly, using similarity matrices and deep learning architectures, we adopt a hybrid approach of feature-based technique and image-based analysis for accurately profiling malware binaries. Such a hybrid deep learning architecture was not attempted before, and the purpose of this study is to evaluate the performance enhancements of the proposed approach.

Our proposed hybrid model for malware analysis consists of one unsupervised learning model and two supervised learning models as shown in Fig. 3. The pre-processing stage involves adoption of multiple techniques of packed binary detectors to automatically separate packed and unpacked files from the dataset [19,20]. In the pre-processing stage, a controlled execution environment is employed to retrieve the raw messages to arrive at the function calls executed that belong to the executables from the dataset. The pre-processing of such a privately collected dataset of about 52,000 binary samples indicates that about 77% of malware are packed and only 23% are unpacked. The feature processing stage involves applying feature extraction techniques effectively to conduct feature analysis using data mining techniques. All executable programs perform an action using API function calls, and a statistical analysis of the Windows API calling sequence reflects the behavior of a particular piece of code. Binary n-gram features were also extracted for analysis for performing n-gram statistical modeling to obtain the distribution of the executables for a range of n-values varying from 1 to 5. Extracting binary n-gram features to complement the API call features has uniquely helped to train the classifiers correctly. Overall, the first supervised learning model uses a private dataset to train, validate and test an array of machine learning classifiers, including support vector machine (SVM) methods. A similarity matrix is generated for each comparison of the these features and is passed through the similarity measure module to generate the similarity report.

Tables 1 and 2 provide an illustration of the similarity matrices for malware in the same family. They both represent malware variants of each family being compared. Table 1 shows the similarity matrix of the malware family Trojan.Downloader.Win32.Dadobra, while Table 2 shows the similarity matrix of the malware family Worm.Win32.Delf. The table columns are the binary file extensions representing the different versions or variants of the same malware family. For example, the columns ".aa" and ".aj" in Table 1 refer to two file extensions of the variants of the same malware family called TrojanDownloader:Win32/Dadobra. For the similarity score, we adopted the Cosine similarity metric, which is the standard distance measure to compute the similarity between two vectors as given in Eq. (3). Cosine similarity has been used successfully in information retrieval and malware detection.

$$\cos(a, b) = \frac{a.b}{\sqrt{|a|^2 |b|^2}} \tag{3}$$

The similarity scores, which range between −1 to 1, are calculated for developing the similarity matrices for various malware families. The values in each matrix are then given different color schemes based on the different distances from the threshold values. The image patterns developed are compared with other samples to identify groups or malware families.

For the image analysis, the image features are extracted from a pre-trained deep convolutional neural networks (CNN) model and then clustered in the image feature space using k-means clustering algorithm. In general, the extracted malware features from the CNN model is high dimensional. Hence these are passed into t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm, which uses the concept of principal component analysis (PCA) technique. The PCA reduces the high-dimensional features into two principal component features. Finally, these features are visualized by plotting the first feature on the X axis and second feature on the Y axis. Fig. 4 shows the distance from the centroid of the first two principal component features from various malware data points that are plotted over X and Y axes. Different colors shown

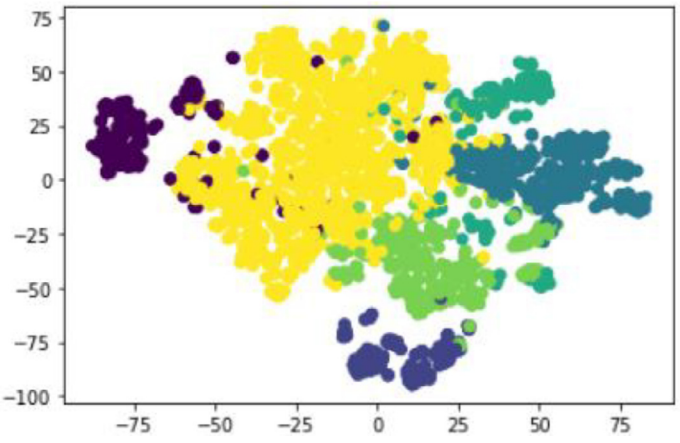


Fig. 4. Malware data with k-means clustering over two principal component features (X-Y axes).

in Fig. 4 correspond to the 6 different malware families obtained according to the k-means clustering algorithm.

While arriving at the similarity matrix, the classification methods require the training data to validate the threshold values that are formulated in these models. We adopt the k-fold cross-validation method to evaluate the results obtained from the statistical analysis. By having $k = 10$, 90% of the full dataset is used for "training" (and 10% for "testing) for each of the independent 10-folds. In order to achieve a higher accuracy of the predictive model for generalization, k-fold cross-validation approach was used and applied for test data, with $k = 10$. An evaluation of both feature selection and classification were done in a 10-fold cross-validation loop for all the malware and benign datasets. Then SVM was applied to the "training" dataset with the goal to produce a predictive model for the "testing" dataset. Different similarity mining metrics using eight different distance measures were employed for benchmarking of the results. The accuracies achieved for malware classifications were compared based on the following standard measures:

1. True Positive (TP): Number of correctly identified malicious code,
2. False Positive (FP): Number of wrongly identified benign code, when a detector detects benign file as a malware.
3. True Negative (TN): Number of correctly identified benign code.
4. False Negative (FN): Number of wrongly identified malicious code, when a detector fails to detect malware.

The efficiency of the proposed hybrid model was evaluated using the following performance measures:

Positive (P): The predicted attribute belongs to the right class.

$$P = TP + FN$$

Negative (N): The predicted attribute belongs to the wrong class.

$$N = FP + TN$$

Overall Accuracy giving the percentage of correctly classified binary is given by:

$$Overall\ Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P + N} \quad (4)$$

Though API calls have been analyzed to profile malwares and their families in previous research studies [18–20], in this study, we have enhanced the models from more recent research work [22,24,52]. The focus here is to develop a hybrid model of machine learning and deep learning architectures for image-based analysis in order achieve higher performance and scalability. Further, we have conducted validation of results with well-known benchmarked public datasets, such as Microsoft Malware Classification Challenge (BIG, 2015) and the Malimg. The experimental details and the findings of the study are presented in the subsequent sections.

4. Experimental setup, datasets and results

We conducted the experimental investigation of our proposed hybrid model using different datasets. Firstly, the similarity analysis was carried out by implementing distance measures and analysis of the various data mining algorithms in Python Programming Language. The experiment was run in three different processors, which aided in the effective malware classification and was evaluated using very large real-life malware dataset consisting of about 75,000 samples obtained through public databases such as VX Heavens [53]. More than two-thirds of the samples were malware and the remaining were benign samples. The similarity distance system developed in this research was able to automatically identify all malware variants. Tables 1 and 2 provide an illustration of the similarity matrices for malware in the same family. In these matrices, the similarity measures calculated are color-coded based on the distance from the threshold values. These distance measures can take values between 0 and 1, and highly positive correlations (closer to a value of 1) are displayed in blue color which represents high similarity, while low correlations (closer to a value of 0) are displayed in red color to represent low similarity. We scaled the distance measures (m) from the [0, 1] range to the similarity metric [−1, 1] range by transposing the values using the formula, ((1 - m) - 0.5)*2. Also, we made use of the color intensity and the size of the circle to be proportional to the correlation coefficients. We employed the R corrplot function to convert the correlation matrix into a graph Correlogram.

An image pattern analysis of the visual representations show that Table 1 has a close similarity to the known malware called Win32.Dadobra (a Trojan), and Table 2 is similar to the known malware called Win32.Delf (a Worm). Further, we conducted experimental comparisons between each pair of malware families to understand whether their behavior patterns were similar. As an example, Table 3 shows the similarity matrix between two different malware families Win32.Dadobra and Win32.Delf, and Table 4 shows the similarity matrix obtained for all the benign files.

From the experimental results obtained with similarity matrices, it was evident that the obfuscated malware or variants from the same family exhibited high similarity in the image patterns, while different families of malware exhibited distinctly different image patterns. Also, the experiments confirmed that there is no similarity among the different benign files, but they exhibit a similar image representation of the similarity matrix, which is distinctly different from that of a malware.

We performed further experimental studies to benchmark our proposed hybrid deep learning model by employing two publicly available datasets: i) Microsoft Malware Classification Challenge (BIG, 2015) dataset [42] and ii) Malimg dataset [54]. The first dataset contains 10,868 samples and 9 malware families of labeled training dataset that is publicly available on Kaggle. The detailed statistics across each malware family is shown in Table 5. The second dataset contains 9342 grayscale images of 25 malware families from which we prepared disjoint datasets of 70% for training and the remaining 30% for testing. To conduct the experimental study, we implemented the deep learning architectures using TensorFlow with Keras higher level API on the GPU enabled computers in a single NVidia GK110BGL Tesla k40 system [43,44]. A detailed anal-

Table 1
Similarity matrix of the malware familyTrojan.Downloader.Win32.Dadobra.

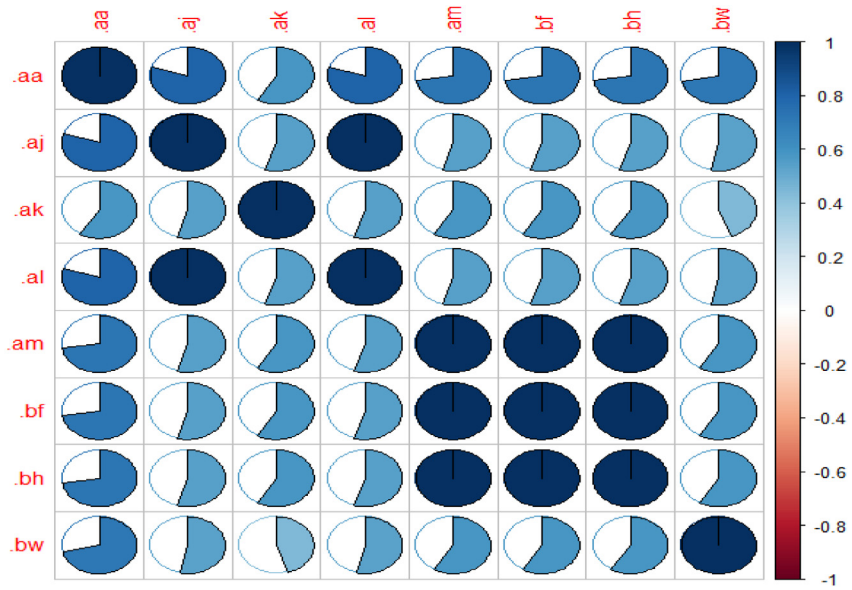
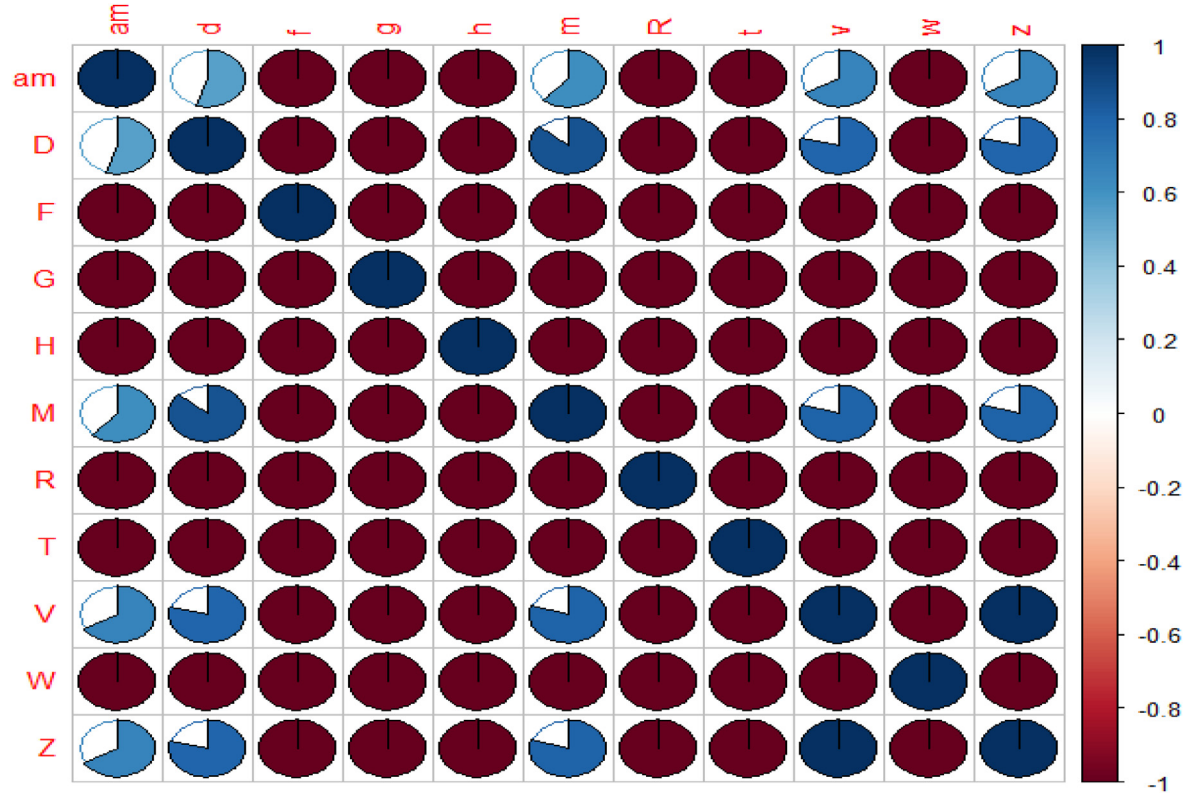


Table 2
Similarity matrix of the malware family Worm.Win32.Delf.



ysis of the results and observations of our experimental study are provided in the next subsections.

There are two types of metrics, namely micro-averaging and macro-averaging to identify the quality of the overall classification of a model. In macro-averaging, a metric is averaged over all classes and they are treated equally. On the other hand, micro-averaging is based on the cumulative True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) measures. In general, the micro-averaging types give importance to the classes that have more samples, while macro-averaging types give better indicators for a multiclass imbalance problem. Hence, in this study, due to the multiclass imbalance nature of the publicly available datasets, we have adopted macro-averaging as the metric for the experimental evaluation. The metrics used, namely Macro averag-

Table 3
Similarity matrix of two malware families Trojan.Downloader.Win32.Dadobra vs Worm.Win32.Delf.

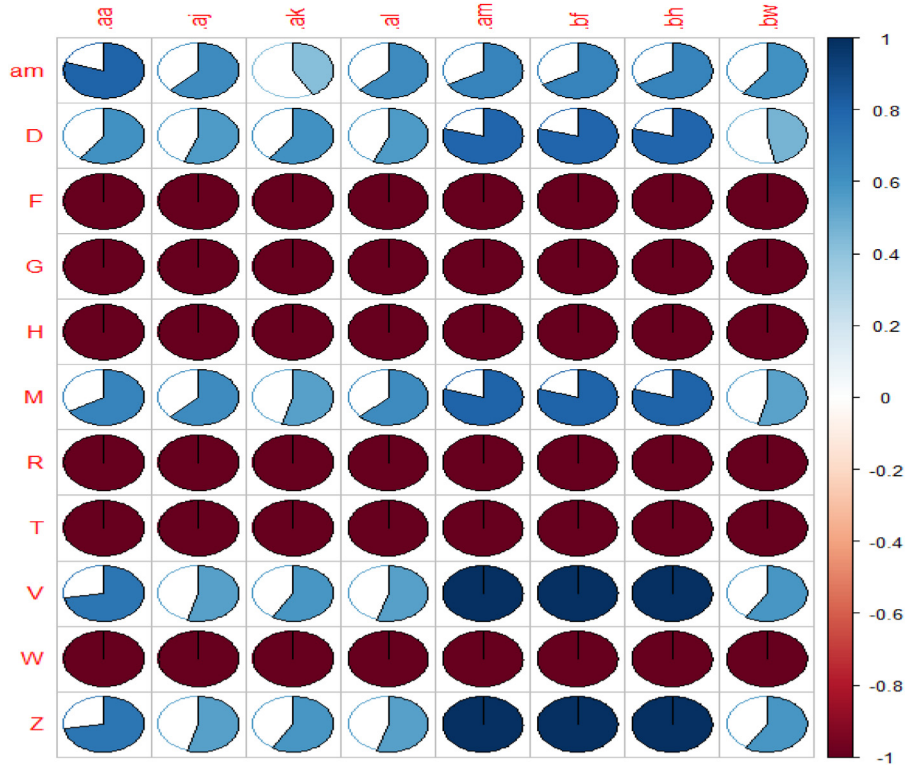


Table 4
Similarity Matrix of Benign Files.

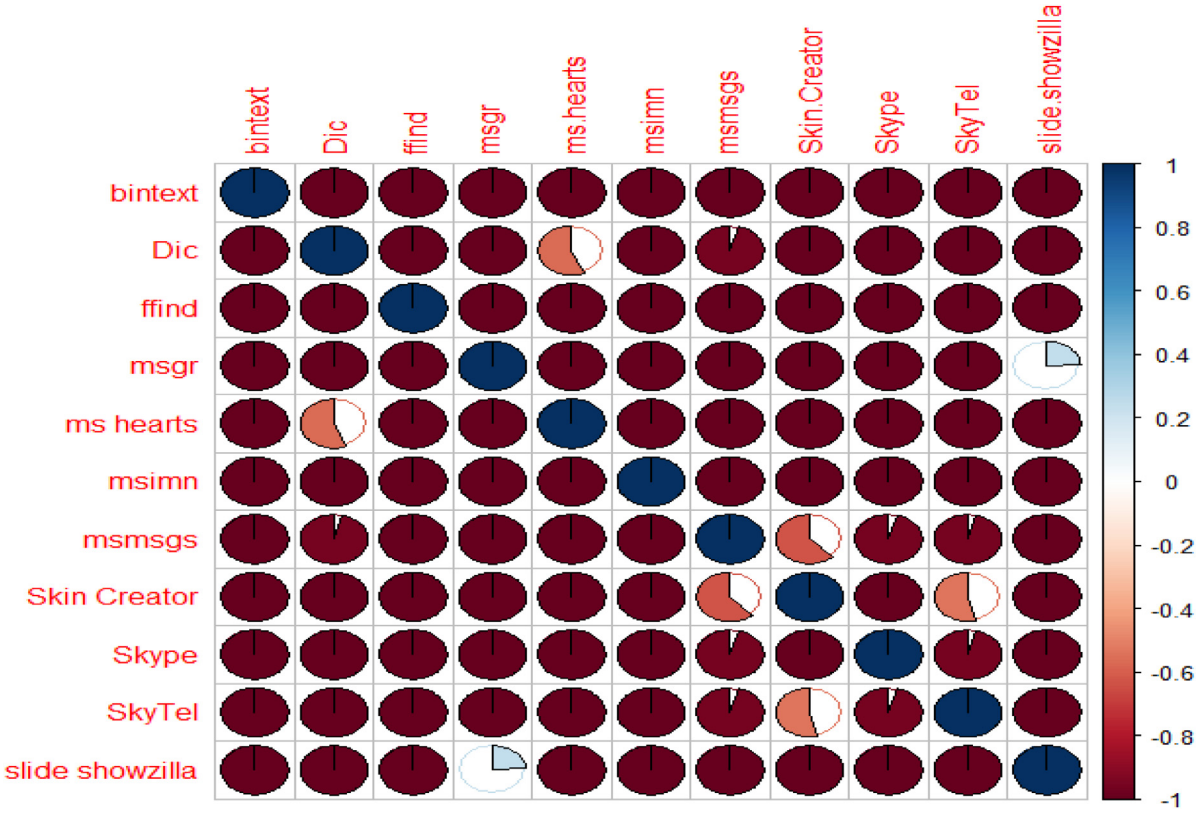


Table 5
Detailed statistics of dataset.

Malware Family	Number of Samples
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013

ing Precision, Recall and F1-score are defined in Eqs. (5)–(7) respectively:

$$Precision_M = \frac{\sum_i \frac{TP_i}{TP_i + FP_i}}{Number\ of\ Classes} \tag{5}$$

$$Recall_M = \frac{\sum_i \frac{TP_i}{TP_i + FN_i}}{Number\ of\ Classes} \tag{6}$$

$$F1 - score_M = \frac{2}{\frac{1}{Precision_M} + \frac{1}{Recall_M}} \tag{7}$$

Microsoft Malware Classification Challenge (BIG, 2015) Dataset Analysis using Deep learning Architectures:

The malware samples in the datasets consisted of binary files that were converted into image representations using the method proposed by [55]. This type of image representation preserves the sequential order of the byte code in the binary files. Since the recurrent neural network (RNN) architectures are well-known methods used for sequential data modeling tasks, we stack RNN with CNN for evaluating our proposed hybrid deep learning model. This achieved the best performance for our experimental study as compared to adopting CNN model alone. The learnable parameter details of the existing CNN architectures with the Unidirectional LSTM (UniLSTM), and Bidirectional LSTM (BiLSTM), as well as with the proposed Unidirectional GRU (UniGRU) and Bidirectional GRU (BiGRU) models are provided in Table 6. We measured the performance of the deep learning models using 3-fold cross validation as reported in Table 7. The parameter details for the best performed model are shown in Tables 8 and 9. The proposed method performed better than the existing cost-sensitive and cost-insensitive methods in all the experiments. More importantly, cost-sensitive models outperformed cost-insensitive models. The performance details of the best performed methods are provided in Tables 10 and 11. For all the malware families, the cost-sensitive models obtained best F1-score as compared to the cost-insensitive models. Most importantly the proposed cost-sensitive models showed per-

Table 6
Parameter Details.

Architecture	#Learnable parameters
CNN [55]	105,569
CNN UniLSTM [55]	155,669
CNN BiLSTM [55]	268,949
CNN UniGRU (Proposed)	127,637
CNN BiGRU (Proposed)	212,885

Table 7
Results of 3-fold cross-validation.

Architecture	Type	F1-score (Macro)
CNN [55]	Cost-insensitive	0.598
CNN UniLSTM [55]	Cost-insensitive	0.664
CNN BiLSTM [55]	Cost-insensitive	0.671
CNN UniGRU (Proposed)	Cost-insensitive	0.669
CNN BiGRU (Proposed)	Cost-insensitive	0.678
CNN UniLSTM (Proposed)	Cost-sensitive	0.662
CNN BiLSTM (Proposed)	Cost-sensitive	0.687
CNN UniGRU (Proposed)	Cost-sensitive	0.675
CNN BiGRU (Proposed)	Cost-sensitive	0.711

Table 8
Configuration details of CNN BiLSTM [55].

Layer (type)	Output Shape	Parameter #
conv1d_1 (Conv1D)	(None, 9994, 30)	240
max_pooling1d_1	(MaxPooling1 (None, 1998, 30)	0
conv1d_2 (Conv1D)	(None, 1992, 50)	10,550
max_pooling1d_2	(MaxPooling1 (None, 398, 50)	0
conv1d_3 (Conv1D)	(None, 392, 90)	31,590
max_pooling1d_3	(MaxPooling1 (None, 78, 90)	0
bidirectional_1	(Bidirection (None, 256)	224,256
dense_1 (Dense)	(None, 9)	2313
Total parameters: 268,949 Trainable: 268,949 Non-trainable: 0		

Table 9
Configuration details of CNN BiGRU.

Layer (type)	Output Shape	Parameter #
conv1d_1 (Conv1D)	(None, 9994, 30)	240
max_pooling1d_1	(MaxPooling1 (None, 1998, 30)	0
conv1d_2 (Conv1D)	(None, 1992, 50)	10,550
max_pooling1d_2	(MaxPooling1 (None, 398, 50)	0
conv1d_3 (Conv1D)	(None, 392, 90)	31,590
max_pooling1d_3	(MaxPooling1 (None, 78, 90)	0
bidirectional_1	(Bidirection (None, 256)	168,192
dense_1 (Dense)	(None, 9)	2313
Total parameters: 212,885 Trainable: 212,885 Non-trainable: 0		

Table 10
Detailed 3-fold cross validation based on cost-sensitive CNN BiLSTM architecture.

Malware Family	Precision	Recall	F1-score
Ramnit	0.4883	0.977	0.5876
Lollipop	0.9842	0.6256	0.7329
Kelihos_ver3	0.9964	0.6648	0.7644
Vundo	0.7809	0.6512	0.6805
Simda	0.6087	0.5159	0.5284
Tracur	0.9122	0.6382	0.7163
Kelihos_ver1	0.9875	0.6377	0.7422
Obfuscator.ACY	0.9663	0.5825	0.6975
Gatak	0.9445	0.6437	0.7317
Average	0.897	0.685	0.713
Macro	0.852	0.66	0.687

Table 11
Detailed 3-fold cross validation based on cost-sensitive CNN BiGRU architecture.

Malware Family	Precision	Recall	F1-score
Ramnit	0.4924	0.9896	0.5954
Lollipop	0.9903	0.6461	0.74927
Kelihos_ver3	0.9993	0.6654	0.7656
Vundo	0.9007	0.6533	0.7255
Simda	0.756	0.5714	0.6253
Tracur	0.9482	0.6449	0.7355
Kelihos_ver1	0.9409	0.6499	0.7359
Obfuscator.ACY	0.9731	0.6115	0.7199
Gatak	0.970	0.6469	0.7435
Average	0.909	0.696	0.725
Macro	0.886	0.675	0.711

Table 12
Detailed test results.

Model	Type	Accuracy	Recall	Precision	F1-Score
CNN [56]	Cost-insensitive	0.943	0.892	0.898	0.893
CNN (Proposed)	Cost-sensitive	0.948	0.897	0.914	0.903
CNN BiLSTM [55]	Cost-insensitive	0.951	0.901	0.910	0.904
CNN BiLSTM (Proposed)	Cost-sensitive	0.958	0.907	0.909	0.908
CNN BiGRU (Proposed)	Cost-insensitive	0.960	0.912	0.918	0.914
CNN BiGRU (Proposed)	Cost-sensitive	0.963	0.915	0.918	0.916

Table 13
Class-wise performance: True Positive Rate (TPR) and False Positive Rate (FPR).

Family	Family Name	CNN [56]		CNN BiGRU (Proposed) Cost-sensitive	
		TPR	FPR	TPR	FPR
Dialer	Adialer.C	1.0	0.0	1.0	0.0
Backdoor	Agent.FYI	1.0	0.0	1.0	0.0
Worm	Allaple.A	0.9605	0.0146	0.9864	0.0083
Worm	Allaple.L	0.9916	0.0103	0.9937	0.0026
Trojan	Alueron.gen!J	1.0	0.0	0.9831	0.0
Worm:AutoIT	Autorun.K	1.0	0.0	1.0	0.0
Trojan	C2Lop.P	0.4833	0.0062	0.6833	0.0091
Trojan	C2Lop.gen!G	0.5909	0.0051	0.6363	0.0051
Dialer	Dialplatform.B	1.0	0.0011	0.9622	0.0004
Trojan Downloader	Dontovo.A	1.0	0.0011	1.0	0.0007
Rogue	Fakerean	1.0	0.0004	1.0	0.0004
Dialer	Instantaccess	1.0	0.0007	1.0	0.0004
PWS	Lolyda.AA 1	0.9219	0.0004	0.9844	0.0
PWS	Lolyda.AA 2	0.9818	0.0	1.0	0.0004
PWS	Lolyda.AA 3	1.0	0.0	1.0	0.0
PWS	Lolyda.AT	1.0	0.0011	1.0	0.0007
Trojan	Malex.gen!J	0.8537	0.0004	0.8780	0.0
Trojan Downloader	Obfuscator.AD	1.0	0.0	1.0	0.0
Backdoor	Rbot!gen	1.0	0.0	1.0	0.0
Trojan	Skintrim.N	0.875	0.0	0.9583	0.0
Trojan Downloader	Swizzor.gen!E	0.3947	0.0101	0.3947	0.0054
Trojan Downloader	Swizzor.gen!I	0.425	0.0076	0.5	0.0076
Worm	VB.AT	0.9508	0.0	0.9672	0.0007
Trojan Downloader	Wintrim.BX	0.8621	0.004	0.8621	0.0014
Worm	Yuner.A	1.0	0.0008	1.0	0.0

formance improvement of 0.0969 for F1-score as compared to the existing methods. The proposed architecture contains less number of parameters compared to the existing methods and hence, it can even reduce the computational complexity in both the training and testing stages.

Malimg Dataset Analysis using Deep learning architectures:

We evaluated the various deep learning models using Malimg dataset and the results are reported in Table 12. The class-wise performance of the best performed model us provided in Table 13. The proposed model performed better than the existing method [56], and the cost-sensitive model showed good performance over cost-insensitive models. The performance can be further enhanced by identifying optimal parameter values for deep learning architectures.

5. Performance evaluation

We performed three trails of experiments for testing our proposed architecture using various deep learning models. The experiments were run until 100 epochs with a batch size of 64, and by using adam optimizer with a learning rate of 0.001. The training and validation performance of the deep learning architectures in terms accuracy and training loss are shown in Figs. 5 and 6 respectively. All the models increased the accuracy and decreased the training loss gradually across the epochs. More importantly, the models achieved better performance once it reached 50 epochs. Additionally, all the models have gradually increased the performance after 50 epochs.

We have also conducted an experimental study using privately and publicly collected large dataset from VX Heavens [53] to evaluate the performance of four variations of a machine learning algorithm by comparing the accuracy of classification of malware and benign files. The experimental results of our hybrid model of feature-based and image-based analysis using similarity mining with eight different distance measures to detect and classify unknown malware have shown promising results. Similarity mining and deep learning architectures are effective to detect malware variants from the same family or different families of malware. Also, the experiments confirm that there is no similarity among the different benign files, but they exhibit a similar image representation of similarity matrix, which is uniquely different from that of a malware. In the classification algorithms, the training data and testing data were selected by making a partition of the malware and benign datasets for carrying out the experiments. We adopted the most common type of cross-validation namely, k-fold cross-validation that is a standard practice adopted in similar research studies adopted for many classifiers [57]. For the similarity mining, we adopted Sequential Minimal Optimization (SMO) algorithm in Support Vector Machine (SVM) method with 4 different kernels: i) SMO-Normalized Polynomial Kernel Function, ii) SMO-Polynomial

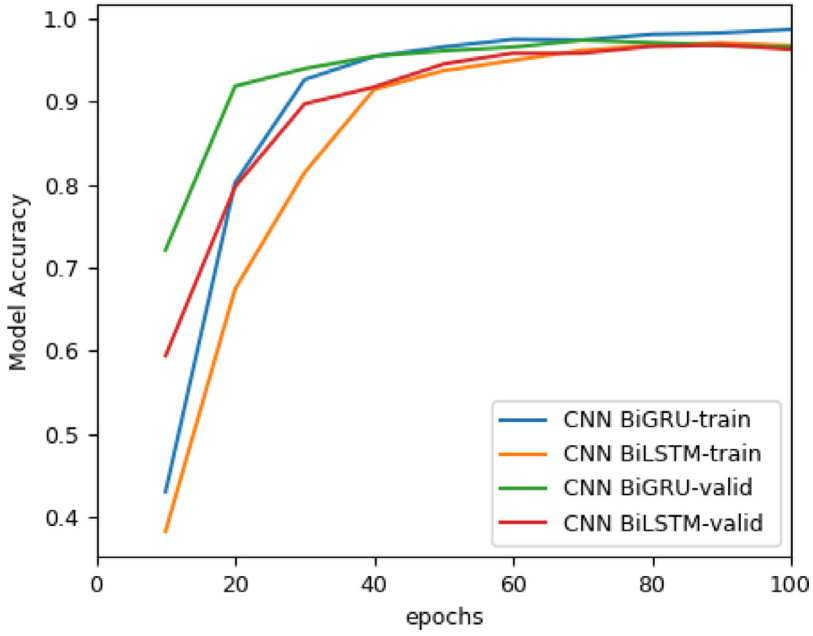


Fig. 5. Comparison of training accuracy among cost-sensitive deep learning models.

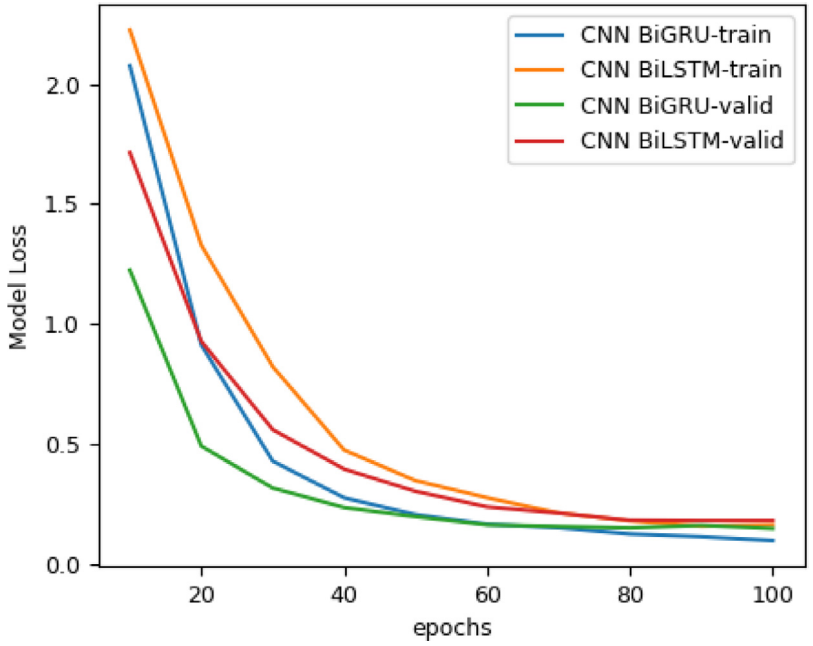


Fig. 6. Comparison of training loss among cost-sensitive deep learning models.

Kernel Function, iii) SMO-Radial Basis Function (RBF) and iv) SMO-Pearson VII kernel function (PUK). The advantage of SMO is its ability to solve the Lagrange multipliers analytically with fast implementation of SVM. Further, it is a popular supervised learning algorithm used for classification and regression problems. In Fig. 7, the overall accuracy rate for malware detection achieved using the four kernels of SMO for our experimental datasets are shown. Normalized Polynomial kernel provides the highest accuracy for all the k cross validations, with k={2,3,4,5,6,7,8,9,10}. In particular, with $k = 10$, we achieved about 98.6% accuracy for SVM based malware detection which is among the best so far reported in literature using large datasets.

Overall, image techniques are being adopted for an effective malware detection and deep learning approaches are becoming more popular. Various researchers are performing several studies in this direction. Recently, some research studies have employed novel malware analysis techniques such as robust hashing and transfer learning for image-based malware classification, reporting strong results of performance comparisons and benchmarking [58,59]. In this paper, we have proposed a hybrid model for image analysis using different similarity mining and deep learning architectures and have conducted a comparative study of their performance with large private and public datasets.

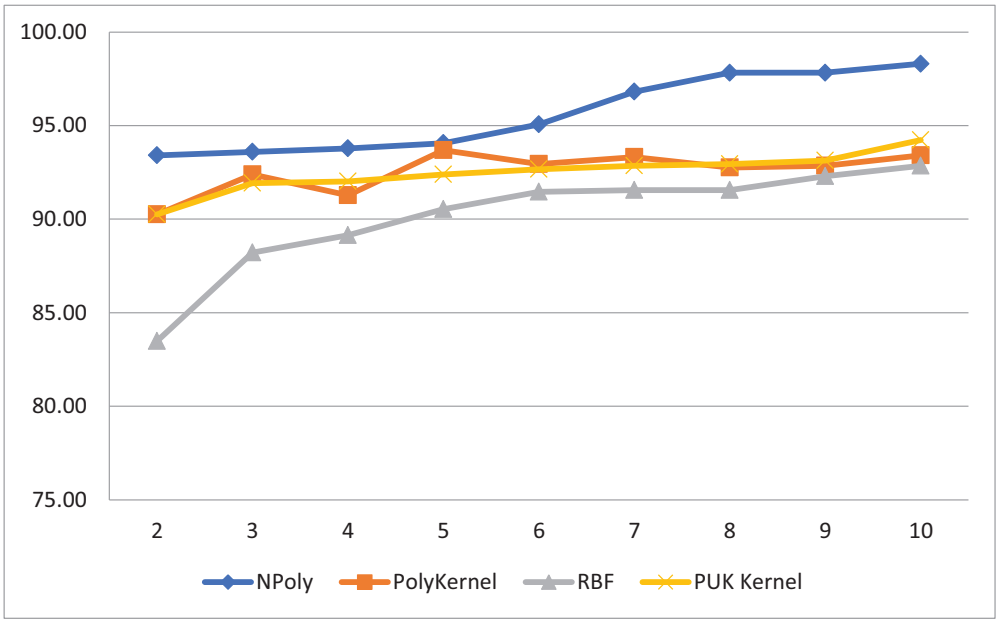


Fig. 7. Accuracy of malware classification using SMO with k cross validations ($k = 2$ to 10).

6. Conclusions

This paper proposed a new hybrid model for image-based analysis using similarity mining and deep learning architectures to identify and classify obfuscated malware accurately. We calculated the similarities between the malware variants using eight different distance measures to generate similarity matrices and to identify the malware family by adopting images of the distance scores. Further, benchmarking of deep learning architectures were performed resulting in high classification accuracies. We achieved almost 99% accuracy in the case of SMO-Normalized Polynomial kernel, and the performance of our proposed cost-sensitive deep learning architectures were comparable with some of the best architectures reported in literature. In summary, we envisage that our image-based approaches have effectively differentiated the behavior patterns of different malware families.

The advantage of the proposed method is that it required less computational cost as compared to the classical machine learning based methods. Further, the proposed cost-sensitive deep learning based model can be continuously trained in real-time to cope with the new malware of the future.

We anticipate to further enhance our proposed framework as future work. The same set of experiments could be run for more than 100 epochs to reach a better performance. Another scope for future research could consider the intersection of our proposed approach with those of other innovative image analysis techniques reported in the literature recently.

Declaration of Competing Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors wish to thank the reviewers and editors for their invaluable suggestions that helped in improving the quality of the paper. This work was supported by the Department of Corporate and Information Services, Northern Territory Government of Australia.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jisa.2019.06.006.

References

[1] Aycock J. Computer viruses and malware. Advances in information security). 1st Edn. New York: Springer-Verlag; 2006.
[2] Mohamed GAN, Ithnin NB. Survey on representation techniques for malware detection. Syst Am J Appl Sci 2017;14(11):1049–69.
[3] You I, Yim K. Malware obfuscation techniques: a brief survey. International Conference on Broadband, Wireless Computing, Communication and Applications; 2010.
[4] Damodaran A, Troia FD, Visaggio CA, Austin TH, Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. J Comput Virol Hack Tech 2017;13(1):1–12.
[5] Christodorescu M, Jha S. Static analysis of executables to detect malicious patterns. In: The 12th conference on USENIX Security Symposium USENIX Association, 12; 2003. p. 12. -12.
[6] Egele M, Scholte T, Kirda E, Kruegel C. A survey on automated dynamic malware analysis techniques and tools'. ACM Comput Surv 2012;44(2):1–49.
[7] Akour M, Alsmadi I, Alazab M. The malware detection challenge of accuracy. In: 2016 2nd International Conference on Open Source Software Computing (OSSCOM); 2016. p. 1–6.
[8] Nikolopoulos SD, Polenakis I. A graph-based model for malware detection and classification using system-call groups. J Comput Virol Hacking Tech 2017;vol.13(, 1):29–46.
[9] Alazab M, Huda S, Abawajy J, Islam R, Yearwood J, Venkatraman S, Broadhurst R. A hybrid wrapper-filter approach for malware detection. J Netw 2014;9(11):2878–91.
[10] Alazab M, Venkatraman S. Detecting malicious behaviour using supervised learning algorithms of the function calls. Int J Electron Secur Digit Forensics 2013;5(2):90–109.
[11] Brosch T, Maik M. Runtime packers: the hidden problem. Black Hat USA 2006.
[12] Martignoni L, Christodorescu M, Jha S. Omniunpack: fast, generic, and safe unpacking of malware. Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual. IEEE; 2007.
[13] Pektaş A, Çavdar M, Acarman T. Android malware classification by applying online machine learning. Computer and information Sciences ISCIS 2016 Communications in computer and information science, 659. Springer; 2016.
[14] Chowdhury M, Rahman A, Islam R. Malware analysis and detection using data mining and machine learning classification. In: Abawajy J, Choo K-KR, Islam R, editors. International conference on applications and techniques in cyber security and intelligence: applications and techniques in cyber security and intelligence. Cham: Springer International Publishing; 2018. p. 266–74.
[15] Malhotra A, Bajaj K. A hybrid pattern based text mining approach for malware detection using DBScan. CSI Trans ICT 2016;4(, 2–4):141–9.
[16] Souri A, Hosseini R. A state-of-the-art survey of malware detection approaches using data mining techniques hum. Cent. Comput. Inf. Sci. 2018;8(3):1–22.

[17] Bagga N, Troia F, Stamp M. On the effectiveness of generic malware models. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE 2018) - Vol. 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, pp. 442–50.
[18] Alazab M, Alkadiri M, Venkatraman S, Broadhurst R. Malicious code detection using penalized splines on OPcode frequency. In: Proceedings of 3rd Cybercrime and Trustworthy Computing IEEE Workshop, CTC 2012, Ballarat; 2012.
[19] Alazab M, Venkatraman S, Watters P, Alazab M. Zero-day malware detection based on supervised learning algorithms of API call signatures. In: Proceedings of AusDM2011 Ninth Australasian Data Mining Conference; 2011. p. 1–2. December 2011.
[20] Alazab M. Profiling and classifying the behaviour of malicious codes. J Syst Softw 2015;100(2):91–102.
[21] Venkatraman S. 'Autonomic framework for IT security Governance. Int Manag Inform Technol (IJMIT) 2017;vol. 9(3):1–14.
[22] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Venkatraman S. Robust intelligent malware detection using deep learning. IEEE Access 2019;7:46717–38.
[23] Bou-Harb E, Debbabi M, Assi C. Cyber scanning: a comprehensive survey. IEEE Commun Surv Tutor 2014;16(3):1496–519.
[24] Venkatraman S, Alazab M. Use of data visualisation for zero-day malware detection. Secur Commun Netw 2018;, 2018 :1–13.
[25] Cao N, Cui W. Introduction to text visualisation. Atlantis Press; 2016.
[26] Keim D. Information visualisation and visual data mining. IEEE Trans Vis Comput Graph 2002;8(1):1–8.
[27] Few S. Information dashboard design - the effective visual communication of data. Sebastopol, CA: O'Reilly; 2006.
[28] Diakopoulos N, Elgesem D, Salway A, Zhang A, Hofland K. Compare clouds: visualizing text corpora to compare media frames. In: Proceedings of IUI Workshop on Visual Text Analytics; 2015.
[29] Diakopoulos N, Elgesem D, Salway A, Zhang A, Hofland K. Compare clouds: visualizing text corpora to compare media frames. In: Proceedings of IUI Workshop on Visual Text Analytics; 2015.
[30] Shiravi H, Shiravi A, Ghorbani A. A survey of visualisation systems for network security. IEEE Trans Vis Comput Graph 2012;18(8):1313–29.
[31] Balakrishnan WB. Security data visualisation. SANS Institute Inc; 2014.
[32] Zhang TY, Wang XM, Li ZZ, Guo F, Ma Y, Chen W. A survey of network anomaly visualisation. Sci China Inform Sci 2017;60(12):121101.
[33] Shanks W. Enhancing intrusion analysis through data visualisation. SANS Institute,Inc; 2015.
[34] Foresti S, Agutter J, Livnat Y, et al. Visual correlation of network alerts. IEEE Comput Graph 2006;26:48–59.
[35] Wagner M, Sacha D, Rind A, Fischer F, Luh R, Schrittwieser S, Keim DA, Aigner W. Visual Analytics: foundations and experiences in malware analysis. In: Othmane Lotfi ben, Jaatun Martin Gilje, Weippl Edgar, editors. CRC/Taylor and francis in book: empirical research for software Security: foundations and experience. Publisher: CRC/Taylor and Francis; 2017. p. 139–71.
[36] Conti G. Security data visualisation - graphical techniques for network analysis. San Francisco: No Starch Press; 2007.
[37] Marty R. Applied security visualisation. Upper saddle river. NJ: AddisonWesley; 2009.
[38] Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: visualisation and automatic classification. In: Proceedings of the 8th international symposium on visualisation for cyber security. ACM; 2011. p. 4.
[39] Yue S. Imbalanced malware images Classification: a CNN based approach. CoRR abs/1708.08042, 2017.
[40] Yajamanam S, Selvin VRS, Di Troia F, Stamp M. Deep learning versus gist descriptors for image-based malware classification. 2nd International Workshop on Formal Methods for Security Engineering (ForSE 2018) Funchal, Madeira, Portugal; 2018. January 22–24.
[41] Cui Z, Xue F, Cai X, Cao Y, Wang GG, Chen J. Detection of malicious code variants based on deep learning. IEEE Trans Ind Inf 2018;14(7):3187–96.
[42] Ronen R, Radu M, Feuerstein C, Yom-Tov E, Ahmadi M. Microsoft Malware Classification Challenge. CoRR, abs/1802.10135, 2018.
[43] Abadi M, et al. Tensorflow: a system for large-scale machine learning. In OSDI 2016;16:265–83.
[44] Gulli A, Pal S. Deep learning with keras. Packt Publishing Ltd.; 2017.
[45] Ni S, Qian Q, Zhang R. Malware identification using visualization images and deep learning. Computers & Security; 2018.
[46] Sun G, Qian Q. Deep learning and visualization for identifying malware families. IEEE Trans Dependable Secure Comput 2018 –1.
[47] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access 2019;7:41525–50.
[48] Long A, Saxe J, Gove R. Detecting malware samples with similar image sets. In: Proc. 11th Workshop on Visualisation for Cyber Security. VizSec, ACM; 2014.
[49] KS Han, Lim JH, Im EG. Malware analysis method using visualisation of binary files. In: Proc. Research in Adaptive and Convergent Systems. RACS; 2013. p. 317–21.
[50] Han KS, Lim JH, Kang B, Im EG. Malware analysis using visualized images and entropy graphs. Int J Inform Secur 2015;14(1):1–14.
[51] Han K, Kang B, Im EG. Malware analysis using visualized image matrices. Sci World J Hindawi 2014;15:1–15.
[52] Venkatraman S, Alazab M. Classification of malware using visualisation of similarity matrices. In: Cybersecurity and Cyberforensics Conference (CCC), 21-23 Nov 2017. London IEEE Explore; 2017. p. 3–8.
[53] VX Heavens. (n.d.). Retrieved from http://vx.netlux.org/lib.
[54] Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: visualization and automatic classification. In: Proceedings of the 8th international symposium on visualization for cyber security. ACM; 2011. p. 4.
[55] Le Q, Boydell O, Namee BM, Scanlon M. Deep learning at the shallow end: malware classification for non-domain experts. Digi Investig 2018;26(1):S118–26.
[56] Cui Z, Xue F, Cai X, Cao Y, Wang GG, Chen J. Detection of malicious code variants based on deep learning. IEEE Trans Ind Inf 2018;14(7):3187–96.
[57] Yan J, Qi Y, Rao Q. Detecting malware with an ensemble method based on deep neural network. Secur Commun Netw 2018;2018:1–16. 7247095.
[58] Huang W-C, Troia F, Stamp M. Robust hashing for Image-based malware classification. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE 2018) - Vol. 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, pp. 451–9.
[59] Bhodia N, Prajapati P, Di Troia F, Stamp M. Transfer Learning for Image-Based Malware Classification, CoRR abs/1903.11551, 2019.