

Received May 10, 2021, accepted June 9, 2021, date of publication June 15, 2021, date of current version June 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3089586

A New Malware Classification Framework Based on Deep Learning Algorithms

ÖMER ASLAN¹ AND ABDULLAH ASIM YILMAZ²

¹Computer Engineering Department, Siirt University, 56100 Siirt, Turkey

²Republic of Turkey Ministry of Agriculture and Forestry, 06800 Ankara, Turkey

Corresponding author: Ömer Aslan (omer.aslan@siirt.edu.tr)

ABSTRACT Recent technological developments in computer systems transfer human life from real to virtual environments. Covid-19 disease has accelerated this process. Cyber criminals' interest has shifted in a real to virtual life as well. This is because it is easier to commit a crime in cyberspace rather than regular life. Malicious software (malware) is unwanted software which is frequently used by cyber criminals to launch cyber-attacks. Malware variants are continuing to evolve by using advanced obfuscation and packing techniques. These concealing techniques make malware detection and classification significantly challenging. Novel methods which are quite different from traditional methods must be used to effectively combat with new malware variants. Traditional artificial intelligence (AI) specifically machine learning (ML) algorithms are no longer effective in detecting all new and complex malware variants. Deep learning (DL) approach which is quite different from traditional ML algorithms can be a promising solution to the problem of detecting all variants of malware. In this study, a novel deep-learning-based architecture is proposed which can classify malware variants based on a hybrid model. The main contribution of the study is to propose a new hybrid architecture which integrates two wide-ranging pre-trained network models in an optimized manner. This architecture consists of four main stages, namely: data acquisition, the design of deep neural network architecture, training of the proposed deep neural network architecture, and evaluation of the trained deep neural network. The proposed method tested on Malimg, Microsoft BIG 2015, and Malevis datasets. The experimental results show that the suggested method can effectively classify malware with high accuracy which outperforms the state of the art methods in the literature. When proposed method tested on Malimg dataset, 97.78% accuracy is obtained which is outperformed most of the ML-based malware detection method.

INDEX TERMS Malware, malware classification, malware detection, malware variants, deep neural networks, transfer learning, deep learning.

I. INTRODUCTION

Recent technological advances on computer systems and the Internet make a human life easier and convenient. These days, it is possible to do everything on the Internet including social interaction, monetary transactions, measurement of human body changes, etc. All of these developments lure cyber criminals into committing crimes in cyberspace rather than in real life. According to recent scientific and business reports, cyber-attacks cost trillions of dollars to the world economy [1]–[3]. Cyber criminals often use malware to launch cyber-attacks. Malware is any software which performs

unwanted and suspicious activities on victim machines. Malware can be categorized into various types such as virus, worm, Trojan, rootkit, ransomware, etc. Malware variants can steal confidential data, initialize distributed denial of service (DDoS) attacks, and perform disruptive damage to the computer systems. New malware variants use concealing techniques such as encryption and packing to remain invisible in the victim's system [2]. Those new variants spread by exploiting human trust as an infection vector. For instance, opening email attachments, downloading fake applications, visiting and downloading files from phony websites are well-known methods of malware spreading vectors.

To protect the computer systems, we have to detect malware as soon as it infects the systems. Malware detection is

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh¹.

the process of analyzing the suspicious file and identifying whether it is malware or benign. Malware classification is one step further. After the file is identified as malware, specifying the category or family of malware known as malware classification. Detecting malware requires 3 steps operations:

1. Malware files are analyzed with appropriate tools.
2. Static and dynamic features are extracted from the analyzed files.
3. Features are grouped in certain ways to separate malicious software from benign.

To increase the detection rate, different sciences and techniques including data science, machine learning, heuristic as well as technologies such as cloud computing, big data, and block chain are used in these processes. There are different malware detection approaches using the above techniques and technologies. These approaches are mainly signature-, behavior-, model checking, and heuristic-based detection [2], [4]. The names of these approaches vary according to the techniques and technologies that are used. Signature-based approach is effective for known and similar versions of the same malware. However, it fails to detect previously unseen malware. Although behavior-based, heuristic-based, and model checking-based detection approaches are effective in detecting some portions of the unknown malware, they cannot show the same performance when detecting complex malware variants which are using obfuscation and packing techniques.

Deep learning-based approach is starting to be used as a new paradigm to eliminate the shortcomings of existing malware detection and classification approaches. Deep learning has been used extensively in different areas including image processing, computer vision, human action recognition [5], driving safety [6], facial emotion recognition [7] and natural language processing. However, it has not been used sufficiently in the cyber security field, especially in malware detection. Deep learning is a subset of artificial intelligence which works based on artificial neural networks (ANN). Deep learning uses several hidden layers and learns from examples. To increase the model performance, there are several deep learning architectures used recently such as deep neural networks (DNN), deep belief networks (DBN), recurrent neural networks (RNN), and convolutional neural networks (CNN). Deep learning brings many advantages over traditional learning schema:

1. DL model can automatically generate high-level features from existing features.
2. DL reduces need for feature engineering.
3. DL can handle unstructured data efficiently.
4. DL can process very large datasets.
5. DL reduces feature space.
6. DL can perform unsupervised, semi-supervised and supervised learning efficiently.
7. DL reduces cost and increases accuracy.

This study proposes a novel hybrid deep-learning based architecture for malware classification. In the proposed

method, malware data gathered from Microsoft BIG 2015, Maling and Malevis datasets. Malware samples are first converted into grayscale images and given to the DL system. After the image acquisition section is fulfilled, the proposed method extracted high-level malware features from malware images by using the convolution layers of the proposed hybrid architecture. Finally, the system is trained in a supervised manner. Overall, several comprehensive deep-learning models, which are relying on a transfer-learning method, are combined so as to produce a hybrid model in the suggested model. During the aforementioned processes, several hidden layers and Rectified Linear Unit (ReLU) function are applied. The test results presented that the proposed method can effectively extract distinctive features for each malware type and family for classification. Experiment results also showed that proposed DL method classifies distinct malware variants with high accuracy which outperforms the state of the art methods in the literature. The major contributions of the paper is listed as follows:

1. A novel hybrid deep learning-based malware classification method is proposed.
2. Suggested method uses a new hybrid layer that involves two pre-trained models instead of one model.
3. Distinctive features are extracted from malware data for various categories.
4. Proposed method reduces feature spaces significantly.
5. Proposed method is tested on three well-known malware datasets.
6. Measured accuracy rates are higher than those of known methods.

The remainder of this paper is organized as follows. Section II explains the malware analysis, feature extraction, and detection processes; and reviews the existing malware detection and classification methods in the literature. Section III describes a proposed hybrid deep learning architecture framework. Section IV presents experimental results and discussion. Section V explains the limitations of the proposed model and future research directions. Finally, section VI presents the conclusion.

II. RELATED WORK

Malware detection and classification is a long process. Various techniques and technologies are used in these phases. In order to detect malware, first it needs to be analyzed with the use of relevant tools. Second, tools results are logged and features are extracted manually or automatically. In this phase, data mining techniques are used to get meaningful features [2]. Then, the extracted features are selected according to certain criteria. Finally, selected features are trained by ML algorithms or rule-based learning techniques to separate malware from benign [2], [4]. Malware analysis, detection and classification processes can be seen in Figure 1. In order to better learn the content and purpose of the malware, a further classification can be made by detecting the types and classes of malware it belongs to [8], [9].

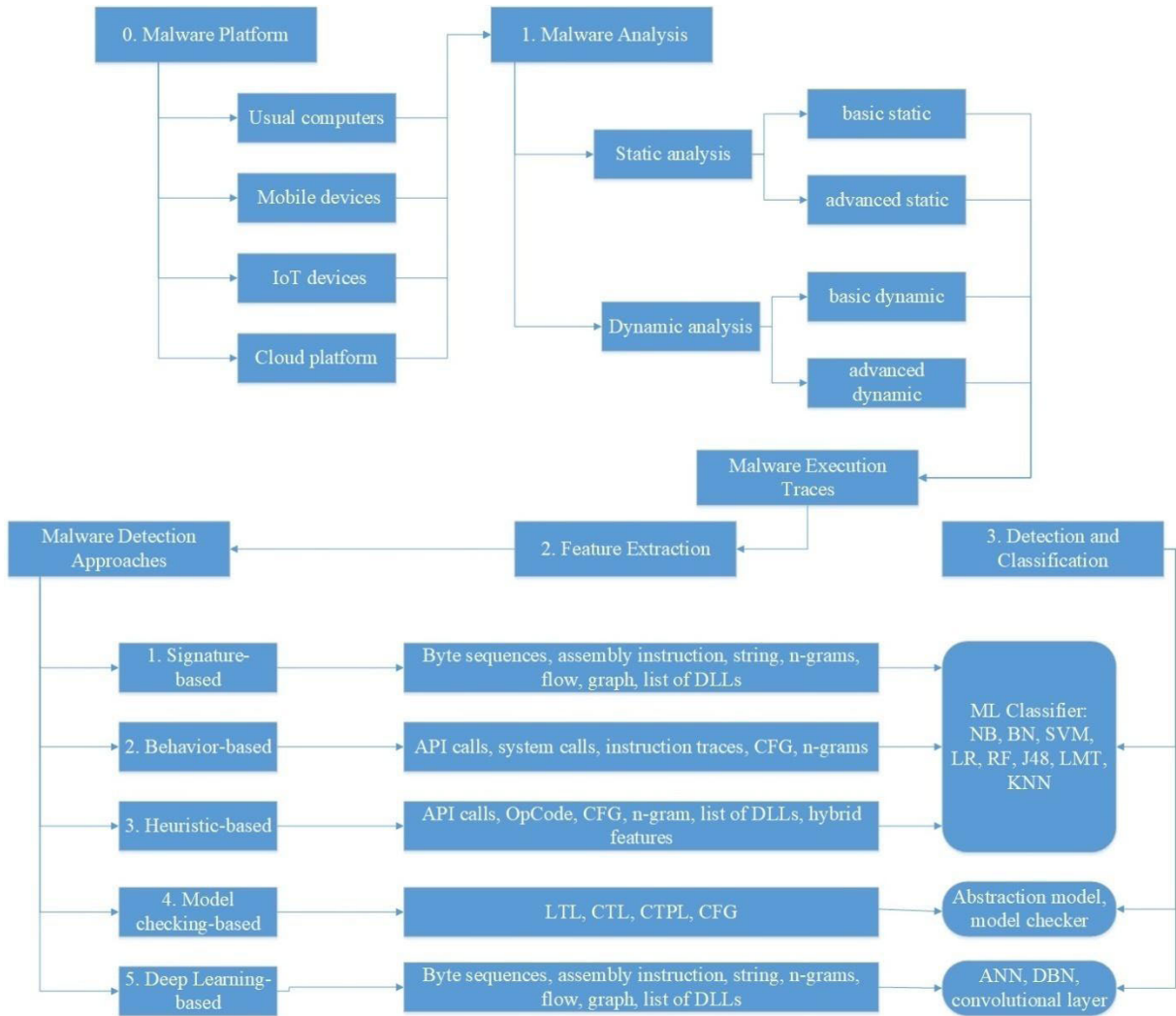


FIGURE 1. Malware analysis, detection and classification processes.

To understand the malware detection process and the techniques that are applied more clearly, this section is divided into four subsections: malware detection devices and platforms, malware analysis, malware feature extraction, and detection and classification.

A. MALWARE DETECTION ON DIFFERENT DEVICES AND PLATFORMS

Malware detection and classification approaches can be performed on different devices and platforms including:

1. Desktop and laptop computers.
2. Mobile devices.
3. IoT devices.
4. Cloud computing platform.

At first, malware variants were written only for usual computers because there were no other devices. Timely, other devices such as smartphones, personal digital assistants

(PDAs), and Internet of things (IoT) have become recognized. Between 1990 to 2000, computer viruses became very popular [10]. From 2000 to 2010 first computer worm, then Trojan became familiar [10]. Since 2010 up to now, ransomware has become very popular malware [11]. From 1990 to 2010, most of the malware types were written for usual computers especially for Windows operating systems (OSs) because other OSs such as Unix, different suits of Linux and macOS were not as common as Windows. Thus, malware detection approaches were proposed for computers. However, after 2010 mobile devices such as smartphones, tablets, and PDAs became popular devices. Then, cloud computing environments and IoT devices become so popular.

According to recent studies, the number of smartphones has exceeded the number of computers and this difference is increasing every day. People use mobile apps. more than the web version of the program. The number of IoT devices

is growing as well. Since the usage of smartphones and IoT devices has become more popular than usual computers, cyber criminals' interests have shifted from usual computers to smartphones and IoT devices as well. Cybercriminals make changes to existing malware and create different versions of the same malware which can run on those devices.

According to a McAfee report, there is an enormous increase in banking Trojans, backdoors, and fake applications for mobile platforms [12]. In addition, the malicious attacks associated with cryptocurrency, social media, IoT devices, and cloud computing environments are on the rise as well. These reasons shift the malware detection landscape from computer to mobile-IoT devices, and cloud environment. Cloud computing brings many advantages to malware detection including easy access, more computational power and much bigger databases [13]. Thus, most of the recent papers on malware detection and classification methods are written for mobile and IoT devices using deep learning and implemented in cloud computing environments.

B. MALWARE ANALYSIS METHOD

Malware samples must be analyzed to find out the nature and behaviors of malware [14]. Malware analysis is an extremely important process. This is because during the analysis process, malware detection process takes place as well as several questions can be answered: the structure of the malware can be visualized, the infection and spread methods can be discovered and the given damage to the victim's machines can be evaluated [14]. Malware analysis can be divided into two main categories including static and dynamic. Malware analysis starts with basic static analysis and finishes with advanced dynamic analysis [15]. Analysis can be performed manually or automatically. Manual analysis requires domain expert knowledge, while automatic analysis requires advanced data science programming skills.

1) STATIC ANALYSIS

In static analysis, the structure of the malware sample is identified without running the actual malware codes [16]. It is divided into two parts: basic and advanced static analysis. In basic static analysis; file strings, header information, functions are examined by looking at the program without going into details [14]. To extract that information various tools can be used including PEiD, BinText, MD5deep and PEview [15]. Basic static analysis is the first step of malware analysis, to get more knowledge about malware, advanced static analysis should be performed. In advanced static analysis, the program commands are examined in detail. To do that disassembler is used to generate assembly codes from machine codes [8], [17]. For this purpose, IDA Pro packet splitter and its extension Hex-Rays decompiler are widely used for advanced static analysis. During the analysis, assembly instructions are examined deeply to find out characteristics of malware. Certain malware functionalities can be obtained as a result of reverse compilation and advanced static analysis. Advanced static analysis provides profound

knowledge about malware purpose and functionality. However, performing this analysis requires advanced expertise of domain knowledge about assembly code instructions and operating system concepts [14], [18].

2) DYNAMIC ANALYSIS

In dynamic analysis, program instructions are executed and the behaviors of the malware are evaluated. To protect the machines from the malware, the analysis is performed in closed environments such as sandbox or virtual machines. During the analysis; function calls, parameters, information flows, file-registry changes, and network activities are examined. Dynamic analysis presents malware real functionality more accurately than the static analysis. Dynamic analysis is divided into 2 parts: basic and advanced dynamic analysis. Basic dynamic analysis investigates the malware behaviors using monitoring tools such as Process Monitor, API Monitor, Process Explorer, Regshot, ApatеDNS, Wireshark and Sandboxes [17]. In advanced dynamic analysis, debugging tools such as OllyDbg, WinDbg are used. Debuggers allow malware analysts to execute each command individually for both viewing and changing the contents of variables, parameters and memory areas [14]. Debuggers work both at the user and kernel level. Using advanced dynamic analysis, most of the malware functionality and dynamic view of the program can be identified. However, the use of debuggers is difficult because it requires advanced domain knowledge about assembly level instructions and operating system concepts.

3) STATIC VERSUS DYNAMIC ANALYSIS

By using static analysis, it is easy and fast to get an overview of the programs and analyze malware that has been frequently seen before. However, it is almost impossible to accurately analyze the malware which uses obfuscation, packed, polymorphic, etc, techniques. Since the program codes are executed during dynamic analysis, malicious software which is using hiding techniques can be detected. However, some malware variants can be aware of being analyzed under sandboxes and virtual environments which results in hiding their real behaviors. Although static analysis performs faster and better for previously known malware, dynamic analysis performs more effective for unknown malware.

C. MALWARE FEATURE EXTRACTION TECHNIQUES

After the malware samples are analyzed, the execution traces are logged. These logs are processed by using data mining techniques to extract malware features. Data mining is the process of extracting previously unknown information from large datasets. At this stage, certain patterns in the data and previously unknown values are determined. Byte sequences, strings, assembly instructions, opcodes, API calls, system calls, and list of DLLs can be used when extracting malware features [2]. In recent years, data mining techniques such as n -gram, m -bag, k -tuple, and the diagram model have been widely used when determining malware

features [19], [20]. There are also several study-specific feature extraction techniques that are proposed for malware detection in the literature [19]–[26].

1) THE N-GRAM, M-BAG AND K-TUPLE TECHNIQUES

The n -gram feature extraction method is widely used in many fields as well as malware detection and classification processes. It can apply static and dynamic analysis attributes to create features. When extracting features from the malware actions, n -gram uses consecutive system calls [21] based on specified n values: 2, 3, 4, 6, etc. For instance, if sample program system calls are in the following order $P = \langle 1, 2, 3, 4, 5 \rangle$, 2-gram and 4-gram will be $\{(1, 2), \langle 2, 3 \rangle, \langle 3, 4 \rangle, \langle 4, 5 \rangle\}$, and $\{(1, 2, 3, 4), \langle 2, 3, 4, 5 \rangle\}$, respectively. Tuple and bag models are similar to n -gram. In the tuple method n can be at any distance while in the bag method the frequencies are more important than the order [19]. Although n -gram is an effective feature generation technique, the rapid increase in the number of features declines its performance. There are different models which are modifications of n -gram proposed in the literature to extract malware features as well [20]–[22]. Generally, those models generate less features than the n -gram.

2) GRAPH-BASED TECHNIQUE

Execution traces which are collected from the previous section used when extracting and representing features. Collected string values, program instructions or system calls are converted into graphs [23]–[25]. In the graph $G(V, E)$, V (nodes) represent system calls and E (edges) represent relationships among system calls. Since the size of the graph increases over time, there are sub-graphs to express the graph approximately [26]. The determination of the sub-graph is expressed as NP -complete in many studies. After sub-graphs are extracted, the detection phase can proceed.

3) VISION-BASED TECHNIQUE

In vision-based feature extraction, there are two main techniques to extract the features and visualize the malware. In the first technique, malware binaries are represented as an image. In this technique, no other tools are required such as Sandbox, Disassembly, API Monitor for feature extraction. Malware binary is converted to an 8 bit vector and then represented as a 2D array [27]. In the second technique, malware analysis is performed by using relevant tools such as Sandbox, API Monitor, Process Monitor, Bintext, and IDA Pro [28]. Then, execution traces are collected such as opcode, byte strings, API calls, system calls, etc. Finally, execution traces are used to visualize the images. During the visualization, different methods such as vectors, treemaps, and graphs are used. Based on the previous studies, it is found that malware types, which belong to the same family, have similar images [8], [27]–[29].

D. MALWARE DETECTION AND CLASSIFICATION APPROACHES

Extracted malware features are categorized by using ML algorithms or heuristic techniques to detect and classify the malware. We can divide malware detection and classification approaches to five distinct groups including: signature-, behavior-, heuristic-, model checking-, and deep learning-based. This number can be increased according to the environment and technologies that are used. Each approach and related literature studies are explained in details as the following:

1) SIGNATURE-BASED MALWARE DETECTION AND CLASSIFICATION APPROACH

Signature is a sequence of bits which uniquely identify the program structure. Since signatures are unique for each program, they are frequently used in malware detection [14], [17], [30]. During the signature extraction, initially the static features are identified from executable files. Afterwards, the signature creation engine generates signatures by using extracted features. Finally, signatures are stored in the database. When the suspicious sample file needs to be marked as malicious or benign, the signature of the file is extracted at the same as before and compared with the previously determined signatures. Based upon the comparison, the sample file is marked as malicious or benign. This detection process is called signature-based malware detection. This detection approach is fairly fast and effective in identifying known malware. However, it fails to detect zero-day malware. Furthermore, according to Scott [31] signature based malware detection is dead because it cannot detect new malware variants, it is not scalable, and it must depend on human interaction.

Automatic signature extraction method is presented by Griffin *et al.* [32]. The presented method automatically extracted the string signatures using a range of library identification techniques and diversity-based heuristics. According to the paper, generated signatures are mostly seen in malware files. Thus, the rate of false identification is reduced. Tang *et al.* [33] explained a simplified regular expression signature using bioinformatics technique for detecting polymorphic worms. The proposed technique produces exploit-based signatures and consists of three phases: identifying the most prominent sub-sequences using the array alignment technique, eliminating noisy sub-sequences, and making the simplified regular expression signature compatible with existing IDSs (Intrusion detection systems). Liu and Sandhu [34] proposed a fingerprint-based signature generation method that detects malware in hardware. The program, which runs according to the tamper-evident architecture, generates different cryptographic hash-based signatures. By using these signatures, Trojans which are embedded in hardware are detected.

2) BEHAVIOR-BASED MALWARE DETECTION AND CLASSIFICATION APPROACH

In the behavior-based detection approach, the behaviors of the sample program are monitored. Based on the behaviors that are gathered, the sample program is decided to be malware or benign. This approach consists of three parts: extracting behaviors, creating properties, deciding the analyzed program as malicious or benign by using ML algorithms [2]. When determining behaviors, system calls, API calls or changes in file, registry and computer network are used. In other words, behaviors are determined by examining the order or frequency of the system calls and file-registry operations. Behaviors are grouped, sequences are formed and properties are obtained using these sequences. Although the program source code changes overtime, the behaviors of the program will not change completely. Thus, several malicious software variants can be detected by using this approach. In addition, new malware, which has been previously unknown, can also be detected by this approach. The biggest disadvantage of behavior-based detection is that malware does not show all of its real behaviors in the protected environment such as virtual machines and sandboxes.

Malware detection system using the graph model is explained by Kolbitsch *et al.* [35]. System calls are transformed into a diagram such that each node represents a system call and the edge represents a transition among the system calls. By giving the result of a system call as an input to the other systems, the connections among system calls are determined. The program diagram to be marked was extracted and compared with the existing diagrams. Based on the comparison, the sample file was marked as malware or benign. In addition, new behaviors, which are observed during the analysis, were dynamically added to the diagram. Lanzi *et al.* [21] proposed a system-centric behavioral model. According to the study, the way malicious software interacts with system resources including directories, files, registries, etc. is different from the interaction of benign software. By using the interaction differences, behavior sequences were created from the system calls. Later, by using these sequences, malware and benign categories were generated. In the proposed method, the n -gram technique was used, but it was difficult to distinguish malicious software from benign because there were too many sequences of behavior with the n -gram technique.

Chandramohan *et al.* [19] proposed the BOFM as a malware detection method, which decreases the number of properties immensely. In the proposed method, interrelated system calls were converted into behaviors in such a way that created behaviors became meaningful. Then, the properties were determined using these behaviors. At this stage, the less repetitive features were eliminated. A feature vector was created using properties and classification performed by applying ML algorithms. Singh *et al.* [36] detected malicious software using behavior-based multiple API system calls. In this method, multiple API sequences were created

using depth-first search and n -grams. Dice coefficient, Cosine Coefficient and Tversky index were used to determine similarities between software while determining multiple API sequences. The sequences created were classified using ML algorithms. Aslan *et al.* [37] proposed a behavioral-based SCBM model to detect malware. The paper captured semantically related features from the analyzed program samples. During the feature extraction, system paths as well as behaviors were taken into consideration. That way malicious behavior patterns were differentiated from benign. According to the paper, the proposed model created fewer features than the n -gram and other leading methods in the literature. Test results showed that the proposed model can handle both known and unknown malware efficiently based upon DR , FPR , f -score and accuracy respectively.

A novel hybrid approach based on dynamic analysis using cyber threat intelligence, ML, and data forensics is proposed in [38]. The paper mentioned that using the concept of big data forensics, IP reputation is predicted in its pre-acceptance stage. Then, associated zero-day attacks are categorized by using behavioral analysis by applying the decision tree algorithm. The proposed method is evaluated based on f -measure, precision and recall scores. According to test results, the obtained f -measure, precision and recall scores are quite satisfactory when comparing with other leading methods in the literature. A novel malware behavioral-based detection method called APTMalInsight is proposed in [39]. Proposed method identified and cognized Advanced Persistent Threats (APTs) which rely on the system call information and ontology knowledge framework. Paper mentioned that with respect to the obtained feature vectors, the APT malware could be detected and clustered with a high percentage.

3) HEURISTIC-BASED MALWARE DETECTION AND CLASSIFICATION APPROACH

Heuristic-based detection is a complex detection approach that uses different techniques together. This approach based on experience uses certain rules and ML techniques [40]. The heuristic approach may utilize both strings and behaviors related features to generate rules. Based upon those rules, signatures are created. It is mostly used to determine different forms of malware as well as previously unseen malware. First of all, the system is trained by using certain features. Then, using data for testing, anomalies are detected. Although the success rate in detecting new malware is high, the rate of false positive (FP) and false negative (FN) is high, too because of optimization issues.

Ye *et al.* [41] proposed an intelligent malware detection system. The purpose of the system is to detect polymorphic and previously unseen malware variants that cannot be detected by antivirus scanners. The system took the API sequences of the given program and then extracted appropriate rules using the FP-growth algorithm. Then, it decide whether the analyzed program files are malicious or benign

by using the classification algorithms. The proposed system worked on the Windows executable file format. According to the results stated in the article, although the proposed method performed better than some antivirus scanners, it did not perform as desired when detecting unknown malware. Canali *et al.* [42] explained a behavior-based signature method. The meaningful combination of the system calls generated behaviors. In this method, signatures were made up of atoms. Atoms could be any of the following system calls, system calls with its arguments, behaviors, and behavior groups with its arguments. Signatures were created by grouping the atoms with certain models. For this purpose, n -gram, k -tuple and m -bag models were used. According to the paper, the proposed method successfully detected malware.

Islam *et al.* [43] defined a detection system that combines static and dynamic properties. This system included three different types of features: frequencies of method lengths (in bytes), printable string information, and system calls and their parameters. The feature vector was created by combining these features and classified using ML algorithms. A dynamic heuristic method is proposed [44] to detect packed malware. First, API call frequencies are calculated. Then, the list of API calls which are strongly associated with malware is identified. Finally, Naive Bayes classifier and Levenshtein distance is used for training and classification. According to the paper, the proposed method produces satisfactory results for packed malware variants.

4) MODEL CHECKING BASED MALWARE DETECTION AND CLASSIFICATION APPROACH

In model checking-based detection approach, malicious and benign features are extracted and coded by using linear temporal logic (LTL) formulas to identify the certain features dependencies which are called specifications [2]. Program features are extracted by utilizing the flow relations among behaviors which use hiding, spreading, and injecting activities. In order to mark the sample program file as malware or benign, the properties that are obtained compared with the previously determined specifications. Based on the comparison, the file is identified as malware or benign. This approach is resistant to stealth and packing techniques and can detect some portion of the new malware variants.

Holzer *et al.* [45] suggested a verification system to detect malicious software. In the suggested system, malicious behaviors are formulated by using the specification language CTPL (computation tree predicate logic), and the finite state model is extracted from the disassembled executable files. If the model controller correctly determined the specification, the analyzed sample is marked as malicious, otherwise benign. In this system, malware has been detected in families using the same attack types. In addition, new malware variants which show similar behaviors are also detected. According to the study, a model checking-based approach determined malware semantic properties more accurately than traditional detection approaches, and this increased

the accuracy of the detection. A proactive malware detection method, which is based on the model checking-based approach, was proposed by Kinder *et al.* [46]. The suggested method can detect different forms of computer worms without signature update. The proposed method extracted control flow charts from the executable files and automatically validated them using the previously specified specifications. For this, they used a new specification language, CTPL. According to the paper experiment results, the suggested method could detect various forms of worms with low *FPR* (false positive rate).

A pushdown model checking-based method is suggested by Song and Tayssir [47] to detect malware. The suggested method reduces the model checking problem to the control of a Büchi pushdown system with symbolic variables. First, executable software codes are transformed into pushdown systems (PDS). Then, by using the SCTPL (stack computation tree predicate logic), the malware behaviors are determined. Finally, the software is determined by comparing the PDSs with the SCTPL specifications. According to the study, the proposed method is resistant to stealth techniques and detects malware with high accuracy. However, the proposed method worked effectively only when data in the stack could not be modified by direct memory access.

5) DEEP LEARNING BASED MALWARE DETECTION AND CLASSIFICATION APPROACH

Deep learning is a subfield of artificial intelligence which learns from examples and inherits from artificial neural networks (ANNs). Deep learning has been widely used in fields such as image processing, driverless cars and voice control, but it has not been used enough for malware detection as well as classification. The deep learning-based detection approach works with high performance and greatly reduces the feature dimension, but is not resistant to evasion attacks [2]. In addition, creating hidden layers takes a lot of time, and building extra hidden layers slightly improves the performance. Deep learning has not been used excessively in malware detection and classification approach yet, thus more academic works are needed to accurately evaluate this approach. The deep learning-based malware detection methods which are used in the literature are summarized as follows.

Deep neural network-based malware detection system using two-dimensional software features is proposed by Saxe and Berlin [48]. The proposed system consists of three main sections: In the first section, four different complementary features have been extracted from malicious and benign samples. In the second section, a deep neural network consisting of an input layer, two hidden layers and output layer is built. In the third section, the outputs of the neural network are identified by using the calibrator score. At this stage, the estimation of whether the file is malware or not is identified. According to the study, the proposed system works with 95% *DR* with low *FPR*. Although the performance of the proposed system is achieved with high accuracy when using the cross validation method, the performance dropped rapidly when

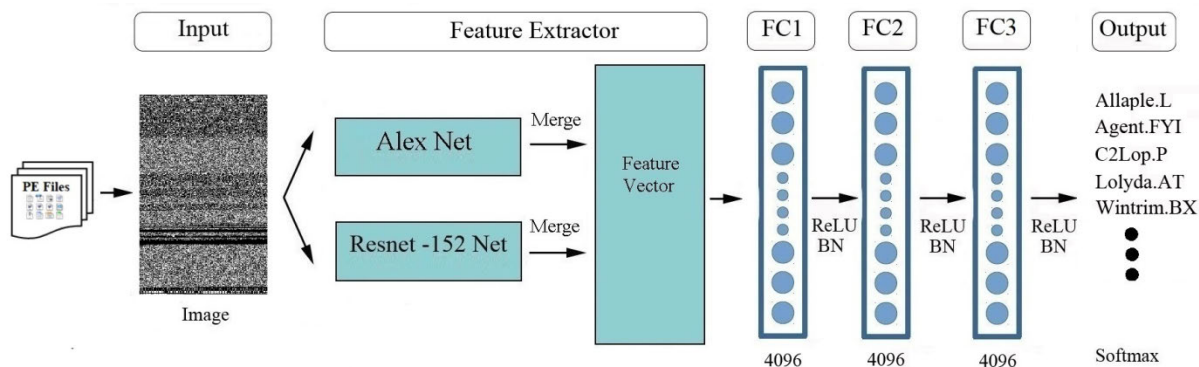


FIGURE 2. Proposed malware classification methodology.

split validation was used. This situation can be eliminated by using de-obfuscation.

Huang and Stokes [49] explained the MtNet architecture, which is multitasking learning for malware classification. In the proposed architecture, malicious and benign samples were trained with data obtained by dynamic analysis. Multitasking learning enables hidden layers to learn better even at low levels. Additionally, the MtNet architecture used the ReLU activation function to halve the number of epochs and reduce the error rate. The article claimed that MtNet performed well when it is compared to a usual neural network architecture. However, in the proposed architecture, the performance of the model could not be increased by adding an additional hidden layer, and it could not be resisted against evasion attacks. Ye *et al.* [50] proposed a heterogeneous deep learning system to detect zero-day malware. The proposed system works using multi-layer constrained Boltzmann machines and associated memory. It consists of two phases: pre-training and fine-tuning. In the pre-training phase, pre-learning is done by learning multi-layered features from labeled and unlabeled files. At this stage, the characteristics of each file were determined. Then, in the fine-tuning phase, supervised learning was performed to separate malware from benign. According to the study, the proposed method increased performance when compared with traditional shallow learning methods.

Roseline *et al.* [8] suggested intelligent vision-based malware detection and classification method. The proposed method is based on the layered ensemble which mimics the characteristics of deep learning. First, program executables are converted into 2D images. Then, based on the image patterns that are gathered, malware variants are classified into their corresponding classes. In this phase, they used a deep forest approach which includes sliding window scanning and cascade layering motivated by CNN concept. According to the paper, the suggested method did not require backpropagation and hyper-parameter tuning. Test results showed that the proposed method successfully detected and classified malware variants on Malimg, BIG 2015, and MaleVis datasets. Hybrid malware classification method, which is using deep convolutional neural network features, is proposed in [9].

They used pre-trained AlexNet and Inception-v3 models to extract features. Then, they used segmentation-based fractal texture analysis of images which represented the malicious code. Malware variants were divided into 25 classes. Extracted features from malware images were classified based on support vector machine, decision tree and k-nearest neighbor. According to the paper, the proposed method achieved a high accuracy rate on Malimg dataset.

Even though the deep learning detection approach is quite effective when detecting malware, it can be deceived by evasion attacks, which leads to misclassification. In adversarial attack, attackers provide deceptive inputs to bypass the detection system by only shifting a few specific bytes in malware binaries. In the proposed malware classification model, necessary contributions are applied in order to reduce the effect of evasion attacks.

III. PROPOSED MODEL

This section presents our proposed malware classification framework based on deep learning methodologies. This framework provides a hybrid deep neural network architecture for malware classification. The methodology of the proposed system, illustrated in Figure 2, is comprised of three main steps. Firstly, the collection of the malware data is accomplished by utilizing several exhaustive datasets. Secondly, the extraction of the low and high level malware features is performed using pre-trained networks. Finally, the training phase belonging to our deep neural network architecture is performed according to a supervised learning method.

This section is divided into two main sub-sections: malware visualization and model overview. In the malware visualization sub-section, malware variants in binary file form are represented as gray scale images. In the model overview section, the proposed malware classification framework methodologies are explained in great detail.

A. MALWARE VISUALIZATION AS AN IMAGE FRAME

There are generally several ways to convert binary code into images [51]. In our work, we used the visualization of executable malware binary files [27]. Our aim is to visualize

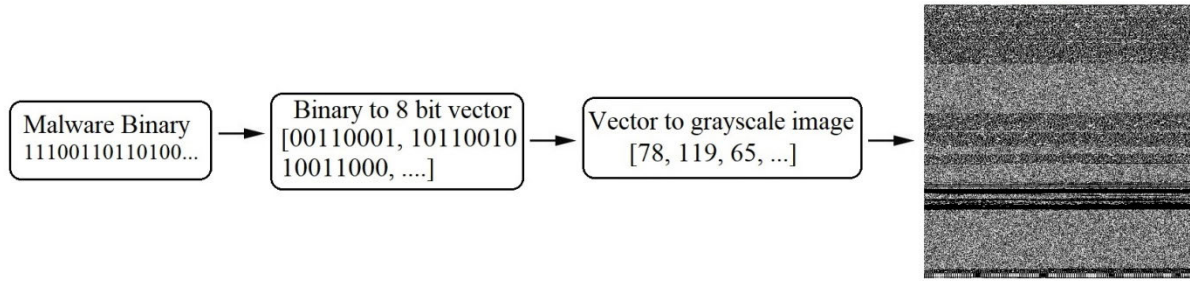


FIGURE 3. Overview of malware visualization process.

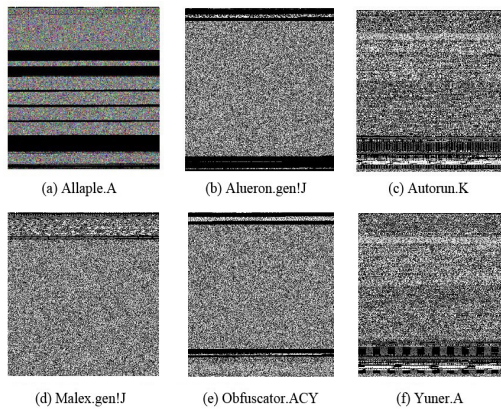


FIGURE 4. Malware grayscale images from different malware families. (These images are obtained from Maling [27], Microsoft BIG 2015 [52] and Malevis Datasets [53]).

binary files as a grayscale image. Figure 3 shows the process of converting malware binary files to grayscale images. Based on Figure 3, first malware binary file is read in a vector of 8-bits unsigned integers. After that, the binary value of each component $A = (a_7+a_6+a_5+a_4+a_3+a_2+a_1+a_0)$ is converted into its equivalent decimal value using equation (1). Finally, the resulting decimal vector is reshaped to a 2D matrix and then interpreted as a grayscale image. Choosing width and height of the 2D matrix basically depends upon the malware binary file size. Here, the spatial resolution provided by Nataraj *et al.* [27] was used to reshape the decimal vectors. Figure 4 illustrates resulting examples of malware visualization image frames from various malware families.

$$A = (a_0 * 2^0 + a_1 * 2^1 + a_2 * 2^2 + a_3 * 2^3 + a_4 * 2^4 + a_5 * 2^5 + a_6 * 2^6 + a_7 * 2^7) \quad (1)$$

B. PROPOSED MODEL OVERVIEW FOR MALWARE CLASSIFICATION

Suggested model provides an optimized framework for malware classification. This framework is designed as a hybrid deep neural network architecture. The methodology of the proposed framework, illustrated in Figure 2, comprises four phases respectively: collection of malware data, design of deep neural network architecture, training phase,

and evaluation stage. In addition system flowchart, which is illustrated in Figure 5, shows a more detailed definition of those stages. Herein, four stages are defined in three parts. In Figure 5, the pretrained networks in the pre-training section function as feature extractors. Additionally, in the training section, the first three layers demonstrate fully connected layers for the learning process and the final layer shows a softmax classifier for the classification process.

Initially, malware data is collected from various datasets including Maling [27], Microsoft BIG 2015 [52] and Malevis [53]. The details of these malware classification datasets are explained in the next part. Then, the proposed deep neural network architecture is designed. Here, two pre-processing stages are performed: Firstly, the process of predicting a suitable DL architecture in order to employ in malware classification processes has been carried out. So that, as it was uncovered in pre-experiments that a hybrid module [54] can provide preferable overall precision. A hybrid module, which contains ResNet-50 and AlexNet architectures, was created utilizing pre-trained architectures.

The ResNet-50 [55] architecture, shown in Figure 6, is a winning model in the ILSVRC 2015 and COCO 2015 competitions and a convolutional neural network that is 50 layers deep. In this network model, five convolutional blocks are used which comprise 1×1 , 3×3 , and 1×1 convolution layers.

Besides, the ResNet-50 network contains two pooling operations, softmax layer and a fully connected layer. The ResNet-50 architecture consists of 25.6 million parameters. AlexNet [56] is one of the prominent convolutional neural networks presented in the ‘‘ImageNet Large Scale Visual Recognition Challenge’’. AlexNet has a basic architecture, which employs 8-layers; the first five are convolutional layers and the last three are fully connected layers. Besides, the AlexNet network contains two normalizations, three pooling, seven ReLU layers, and a softmax layer with regard to the learning and classification processes, as shown in Figure 7.

Next, transfer learning approaches have been investigated in order to overcome the different challenging conditions encountered in the classification process such as time constraint, extreme dataset dimension, etc. In the transfer learning approach, firstly, the feature extraction process is

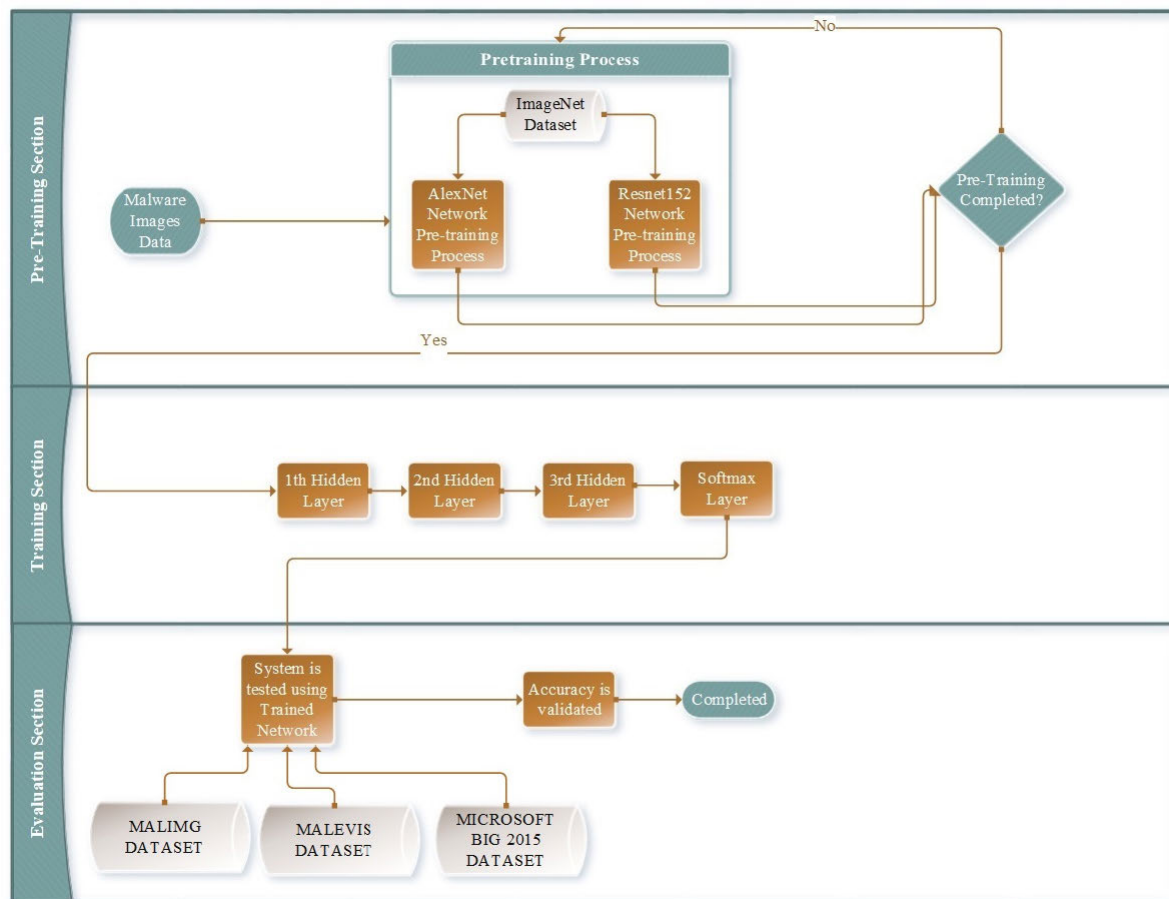


FIGURE 5. Flowchart of proposed deep learning architecture for malware classification.

performed using pre-trained networks. Then, as the last step, the classification process is carried out with a general classifier such as support vector machine, softmax. This approach is tailored for the proposed architecture so as to cope with challenging conditions aforementioned.

Suggested model combines two pre-trained networks with implementing an equal weighting operation in order to create a feature vector. Then, training process was performed to achieve the high accuracy rate. Each stage is described as noted below: Initially, the pre-training process is carried out in which Resnet and AlexNet architectures are trained using ImageNet dataset [57]. Then, in the second step, the features obtained from Resnet and AlexNet architectures are combined to generate a feature vector. This created vector is consist of 4096 dimension. The features generated by the pre-trained architectures of ResNet-50 and AlexNet are respectively the extraction of a 2048 dimensional feature from final fully connected layer, illustrated in Figure 6 and Figure 7. On each grayscale image frame the pre-training stage of the model is finished after the combined feature vector with 4096 elements is acquired. Thirdly, the combined feature vector is passed into the softmax layer and fully connected layers so as to obtain normalization. At this point, softmax layer has 57 outputs which address

the categories of 57 malware, and the fully connected layers comprise 4096 nodes. This layer aims to raise the learning capability of the proposed network (Figure 5). Lastly, experimental analysis of the proposed model was performed by using the exhaustive datasets as inputs to the trained model.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section explains the details of the implementation, experimental results, and evaluation of the suggested deep neural network model. Our experiments were carried out using an Intel Core i9 processor running at 4.8 GHz with 32 GB of RAM Memory in Linux the environment. In order to implement the suggested architecture, Python programming language was employed. Training, validation, and test data were selected at random for each dataset used and assessment processes are performed one by one. For the training, validation and testing stages, the selection rates of the available data are set at 70%, 10% and 20%, respectively. The training procedure of the network architecture was performed for about 30 hours without GPU support and halted at 150 epochs. In order to show the performance of the proposed methods, several evaluation metrics were used. These metrics are accuracy, sensitivity, specificity, and f-score. This performance

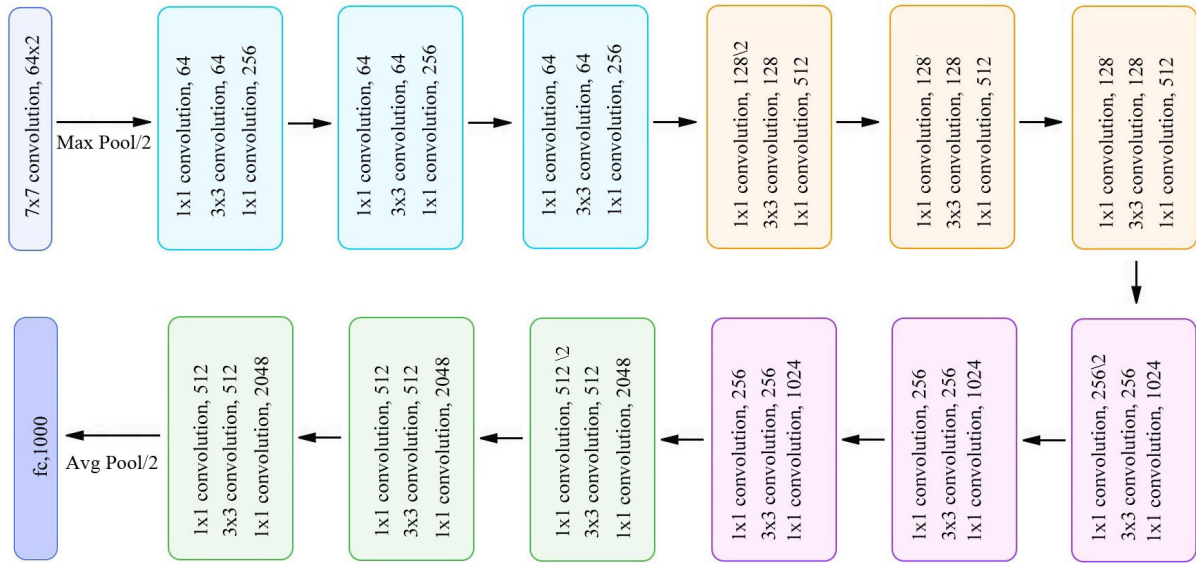


FIGURE 6. An illustration of ResNet-50 network [55].

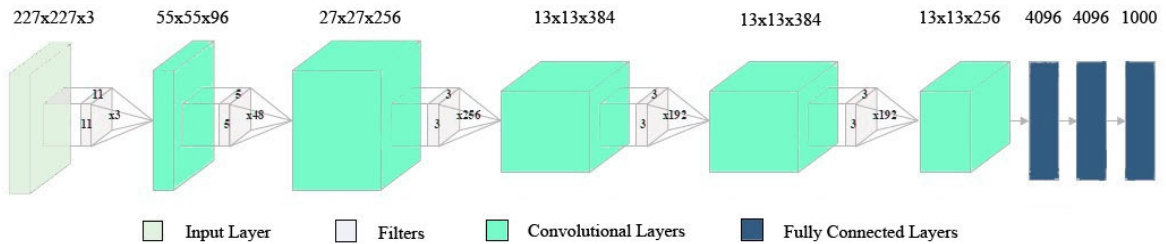


FIGURE 7. An illustration of AlexNet network [56].

metrics are calculated as follows (equation 2, 3, 4 and 5):

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (2)$$

$$Sensitivity = \frac{TP}{TP + FN} \quad (3)$$

$$Specificity = \frac{TN}{TN + FP} \quad (4)$$

$$F - score = \frac{2 * TP}{2 * TP + FP + FN} \quad (5)$$

Here, TP refers to true positive, FP is false positive, whereas TN is true negative and TP is true positive. The performance metrics aforementioned are the first step when interpreting of performance for the suggested model. The comparison processes is carried out for suggested hybrid model with two selected deep neural networks. Figure 8, Figure 9 and Figure 10 show the metric values of the proposed network, AlexNet and Resnet-50 deep neural network models for the individual dataset. Table 1 shows the start values of standard configuration parameters for the suggested hybrid convolutional neural network architecture identified for Mailing, Microsoft BIG 2015, and Malevis datasets.

A. EMPLOYED BENCHMARK DATASETS

Experiments were carried out on three comprehensive datasets. These are Maling, Microsoft BIG 2015, and Malevis datasets. Details of these three datasets are presented below.

The Maling dataset [27] contains 9,339 malware samples. Each malware sample in the dataset belongs to one of the 25 malware classes. Besides, the number of samples belonging to a malware class differ across the dataset. The malware classes contain Adialer.C, Agent.FYI, Allapple.A, Allapple.L, Alueron.gen!J, Autorun.K, Benign, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malx.gen!J, Obfuscator. AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, VB.AT, Wintrim.BX, and Yuner.A.

The Microsoft BIG 2015 dataset [52] contains 21,741 malware samples belonging to 9 different classes, named Ramnit, Lollipop, Kelihos_ver1, Kelihos_ver3, Vundo, Simda, Tracur, Obfuscator.ACY and Gatak. Like the Maling dataset, the number of malware samples over classes is not uniformly distributed. Each malware sample is represented with two files as “.byte”, “.asm”. Here, while “.bytes” file comprises

TABLE 1. Hyper parameter configurations for Malimg, Microsoft BIG 2015, and Malevis datasets.

Parameters	Malimg	Microsoft BIG 2015	Malevis
Batch Size	32	64	64
Dropout	0,5	0,5	0,5
Epoch	40	60	50
Learning rate	0,01	0,01	
0-500 iteration	0,005	0,005	0,01
500-1000 iteration	0,003	0,003	0,005
1000+ iteration			0,003
Momentum	0,9	0,9	0,9
Loss Function	Categorical Cross Entropy	Categorical Cross Entropy	Categorical Cross Entropy
Optimization Algorithm	Stochastic Gradient Descent	Stochastic Gradient Descent	Stochastic Gradient Descent
Weight Initialization	He Initialization	He Initialization	He Initialization
Regularization	0,001	0,001	0,001

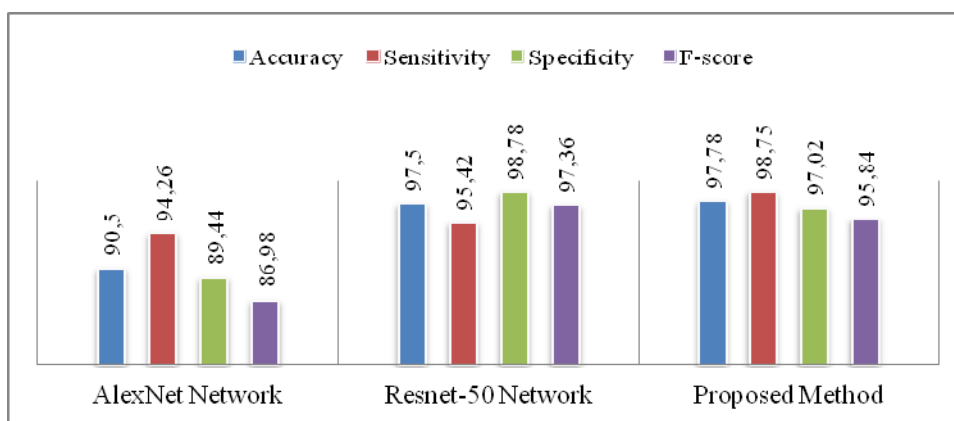


FIGURE 8. Quantitative results on malimg dataset.

the raw hexadecimal representation of the file’s binary content, “.asm” file includes the disassembled code extracted by the IDA disassembler tool. We only used the bytes files to form the malware images in our experiments.

The Malevis dataset [53] contains 9,100 malware samples for training and 5,126 malware samples for testing belonging to 25 malware classes. Each class includes 350 samples for training and varying samples for testing. Malware classes contain Adposhel, Agent-fyi, Allapple.A, Amonetize, Androm, AutoRun-PU, BrowseFox, Dinwod!rfn, Elex, Expiro-H, Fasong, HackKMS.A, Hlux!IK, Injector, InstallCore.C, MultiPlug, Neoreklami, Neshta, Regrun.A, Sality, Snarasite.D!tr, Stantinko, VBA/Hilium.A, VBKrypt, and Vilsel.

B. RESULTS AND DISCUSSION

Evaluation metrics describe the performance of the classification model. The critical point behind the classification is an evaluation metric used to understand the performance and efficiency of an algorithm [58]. Thus, several evaluation

metrics mentioned in the experimental results and discussion section were utilized so as to show the performance of the proposed methods. These metrics are accuracy, sensitivity, specificity, and f-score. Figure 8, Figure 9 and Figure 10 show the metric values of the AlexNet, Resnet-50 deep neural network models and proposed models for Malimg, Microsoft BIG 2015 and Malevis datasets respectively. In accordance with these graphs it can be stated that suggested method outperforms those deep neural network architectures. Also, the performance of our network shows similar performance results in the three datasets, while the performances of the other two deep neural networks differ significantly from the three datasets. The aforesaid situations indicate that our network is more robust and has superior performance in comparison with the other two deep neural networks.

Secondly, malware variants were investigated along with the confusion matrices. Figure 11, Figure 12 and Figure 13 present the confusion matrices for the Microsoft BIG 2015 dataset for nine malware variants (ramnit, lollipop,

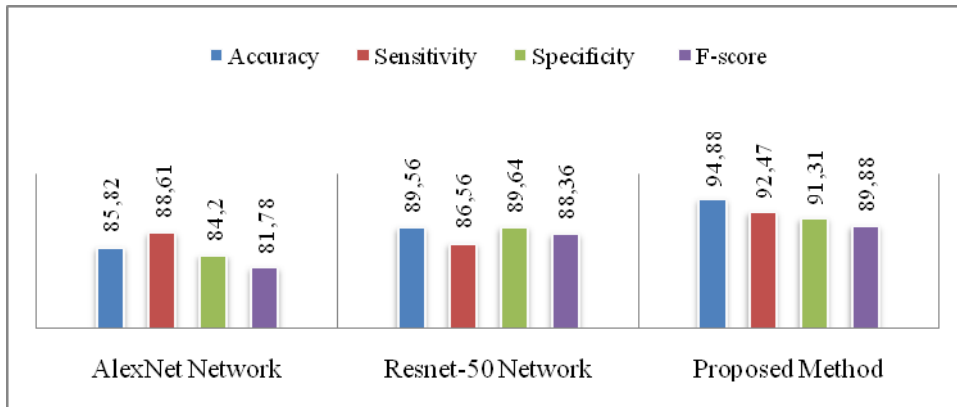


FIGURE 9. Quantitative results on Microsoft BIG 2015 dataset.

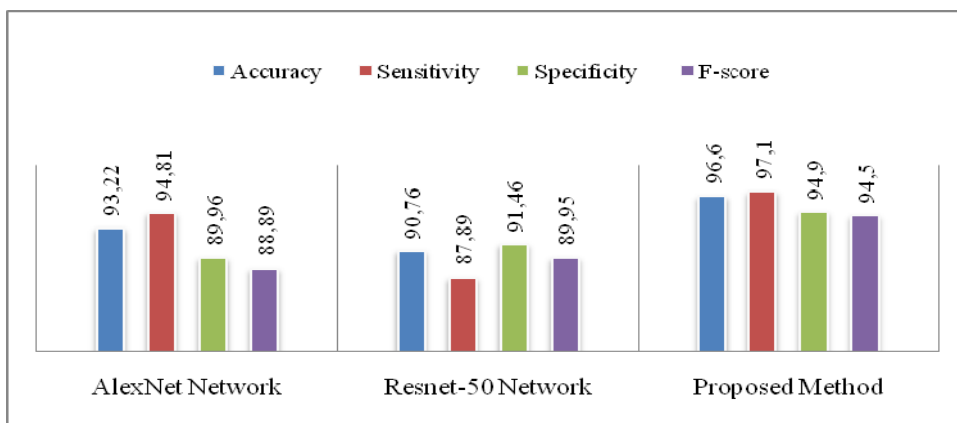


FIGURE 10. Quantitative results on malevis dataset.

TABLE 2. Comparison with existing state-of-the-art algorithms performed on the Maling dataset.

Method	Accuracy(%)
Luo <i>et al.</i> [59]	93.72
Cui <i>et al.</i> [60]	94.5
Gilbert [61]	95.33
Singh <i>et al.</i> [62]	96.08
Vinayakumar <i>et al.</i> [18]	96,3
Proposed Network	97.78

kelihos_ver1, kelihos_ver3, vundo, simda, tracur, obfuscator.acy and gatak) of the AlexNet, ResNet-50 and proposed network models, respectively.

Herein, accuracy rates for each malware variant is demonstrated together with using the confusion matrices. It can be observed that the proposed method, illustrates the confusion matrix in Figure 13, grants better results for whole malware classifications excluding vundo. Besides, The ResNet-50 model which is illustrated in Figure 12, provides a better detection of the vundo malware variant than other network models. In this case, all network models can easily be recognized simda and tracur malware variants.

TABLE 3. Comparison with existing state-of-the-art algorithms performed on the Microsoft BIG 2015 dataset.

Method	Accuracy(%)
Vinayakumar <i>et al.</i> [18]	91.27
Cui <i>et al.</i> [60]	93.4
Luo <i>et al.</i> [59]	93.57
Singh <i>et al.</i> [62]	94.24
Gilbert [61]	94.64
Proposed Network	94.88

Finally, comparison was realized against state-of-the-art results. Table 2, Table 3 and Table 4 show the accuracy values obtained for the Maling, Microsoft Big 2015 and Malevis datasets for the proposed network model and other state-of-the-art studies, respectively. It should be noted that the performance of the proposed architecture outperformed the state-of-the-art algorithms since it generated a higher accuracy value.

V. LIMITATIONS AND FUTURE WORKS

Even though our proposed hybrid deep learning architecture is effective to detect and classify the various malware variants and families, there are still some limitations which need to be addressed. The proposed deep learning architecture

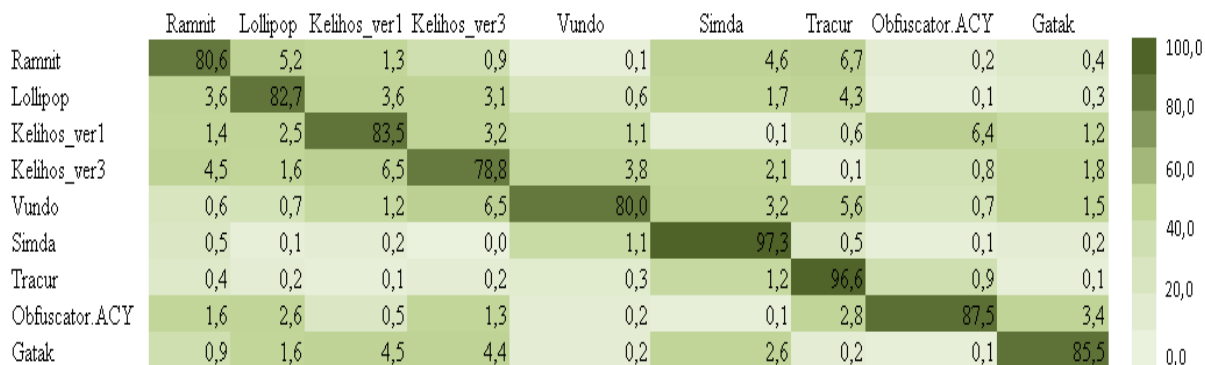


FIGURE 11. The produced confusion matrix on Microsoft BIG 2015 dataset for nine malware variants of AlexNet network.

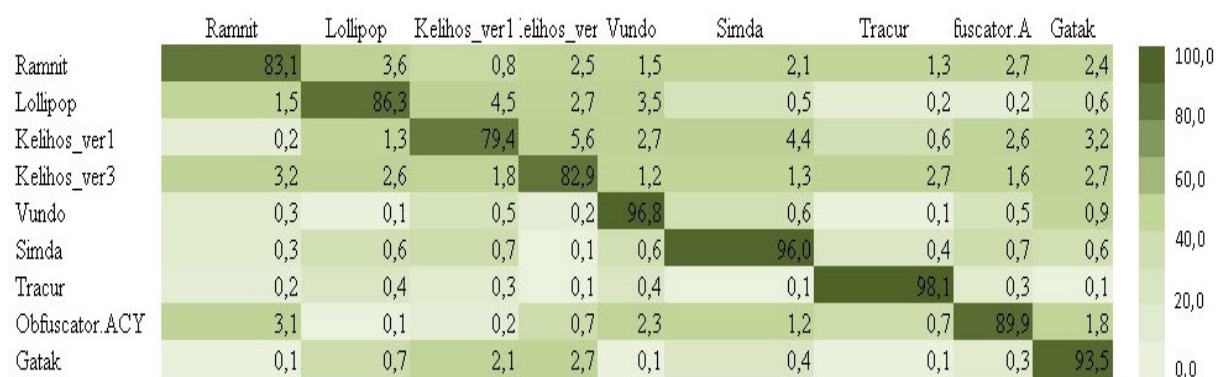


FIGURE 12. The produced confusion matrix on Microsoft BIG 2015 dataset for nine malware variants of ResNet-50 network.

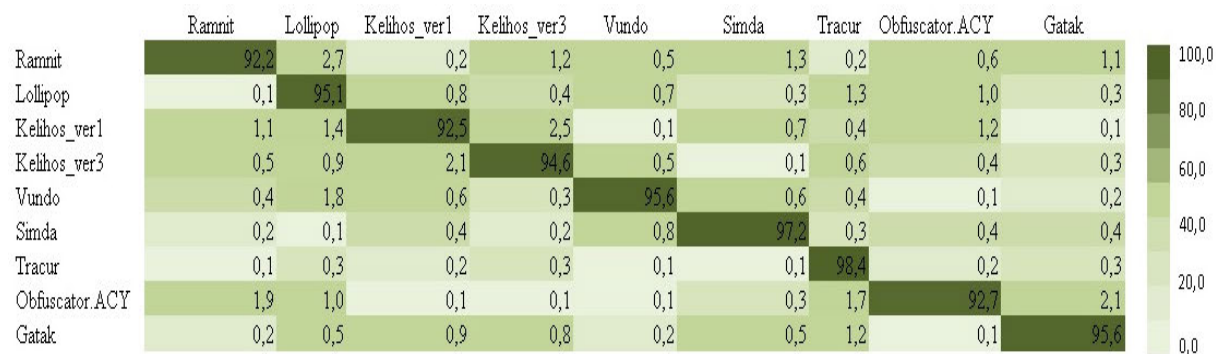


FIGURE 13. The produced confusion matrix on Microsoft BIG 2015 dataset for nine malware variants of proposed network.

is somehow resistant to obfuscation because we obtained satisfactory results on Microsoft BIG 2015 Dataset, which contains some obfuscated malware samples. The proposed method did not test the adversary’s attacks with crafted inputs. We aim to measure our method resiliency to evasion attacks in the next study. Some malware features are similar for different malware families, which cause misclassification. We will improve our deep learning model to decrease such features. The proposed model is tested only on three

datasets including Maling, Microsoft BIG 2015, and Malevis. We would like to test our model on more datasets in the future. Proposed architecture is performed with limited computer power and resources. In the future, we plan to build our model in a cloud computing environment which will provide more computational power and resources. We examined that using more hidden layers in deep learning increases the performance up to a certain level. For future study, less hidden layers will be used to reduce the model complexity.

TABLE 4. Comparison with existing state-of-the-art algorithms performed on the Malevis dataset.

Method	Accuracy(%)
Vinayakumar <i>et al.</i> [18]	86.29
Gilbert [61]	90.59
Ma <i>et al.</i> [63]	91.31
Cui <i>et al.</i> [60]	92.13
Luo <i>et al.</i> [59]	92.24
Singh <i>et al.</i> [62]	93
Proposed Network	96.5

VI. CONCLUSION

Even though a lot of research has been conducted on malware detection and classification, effectively detecting malware variants still remains a serious threat in the cyber security domain. Code obfuscation and packing techniques make the malware detection process a very challenging task. This paper proposed a novel deep learning architecture to effectively detect malware variants. The proposed architecture uses a hybrid approach. This approach includes several exhaustive pre-trained networks which rely upon the transfer learning method. Initially, the collection of the malware data was accomplished by using several comprehensive datasets. Then, the features are extracted by using pre-trained networks. Finally, the training phase of deep neural network architecture is performed regarding a supervised learning method.

The main contribution of the proposed method is to present a hybrid model by optimally combining two well-known pre-trained network models. The proposed deep learning method is evaluated on Maling, Microsoft BIG 2015, and Malevis datasets. Here, the suggested hybrid model is first compared with each individual model separately. The test results confirmed that the proposed method can effectively classify malware with high precision, recall, accuracy and f-score. In addition to this, it is observed that the proposed method is efficient and reduces feature space on a large scale domain. Secondly, the proposed model was evaluated by state-of-the-art methods. The results obtained here also disclose and approve the advantage and supremacy of the proposed method over leading methods in the literature. On the other hand, a minority of malware samples could not be classified correctly. This is because those malware variants are using advanced code obfuscation techniques and show similar features with other malware types. For the next study, we aim to propose a detection system which specifically detects and classifies malware which uses obfuscation techniques.

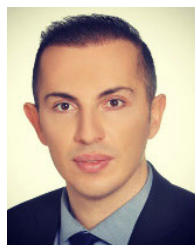
REFERENCES

- [1] R. Lyer, *The Political Economy of Cyberspace Crime and Security*. New York, NY, USA: Academic, 2019.
- [2] Ö. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, Jan. 2020.
- [3] R. Gupta and S. P. Agarwal, "A comparative study of cyber threats in emerging economies," *Globus, Int. J. Manage. IT*, vol. 8, no. 2, pp. 24–28, 2017.
- [4] R. Komatwar and M. Kokare, "A survey on malware detection and classification," *J. Appl. Secur. Res.*, pp. 1–31, Aug. 2020.
- [5] A. A. Yilmaz, M. S. Guzel, E. Bostanci, and I. Askerzade, "A novel action recognition framework based on deep-learning and genetic algorithms," *IEEE Access*, vol. 8, pp. 100631–100644, 2020.
- [6] A. A. Yilmaz, M. S. Guzel, I. Askerbeyli, and E. Bostanci, "A vehicle detection approach using deep learning methodologies," in *Proc. Int. Conf. Theor. Appl. Comput. Sci. Eng.*, Nov. 2018, pp. 64–71.
- [7] A. A. Yilmaz, M. S. Guzel, I. Askerbeyli, and E. Bostanci, "A hybrid facial emotion recognition framework using deep learning methodologies," in *Human-Computer Interaction*, T. Özseven, Eds. Hauppauge, NY, USA: Nova Science Publishers, 2020.
- [8] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," *IEEE Access*, vol. 8, pp. 206303–206324, 2020.
- [9] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Appl. Sci.*, vol. 10, no. 14, p. 4966, 2020.
- [10] J. Love. (2018). *A Brief History of Malware—Its Evolution and Impact*. Accessed: Mar. 20, 2021. [Online]. Available: <https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/>
- [11] D. Palmer. (2017). *Ransomware: Security Researchers Spot Emerging New Strain of Malware*. Accessed: Mar. 20, 2021. [Online]. Available: <https://www.zdnet.com/article/ransomware-security-researchers-spot-emerging-new-strain-of-malware/>
- [12] (2020). *McAfee Mobile Threat Report Q1*. Accessed: Mar. 21, 2021. [Online]. Available: <https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf>
- [13] Ö. Aslan, M. Ozkan-Okay, and D. Gupta, "A review of cloud-based malware detection system: Opportunities, advances and challenges," *Eur. J. Eng. Technol. Res.*, vol. 6, no. 3, pp. 1–8, Mar. 2021.
- [14] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. San Francisco, CA, USA: No starch press, 2012.
- [15] Ö. Aslan, "Performance comparison of static malware analysis tools versus antivirus scanners to detect malware," in *Proc. Int. Multidisciplinary Stud. Congr. (IMSC)*, 2017, pp. 1–6.
- [16] S. K. Pandey and B. M. Mehtre, "Performance of malware detection tools: A comparison," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, May 2014, pp. 1811–1817.
- [17] Ö. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1277–1284.
- [18] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.
- [19] M. Chandramohan, H. B. K. Tan, L. C. Briand, L. K. Shar, and B. M. Padmanabhuni, "A scalable approach for malware detection through bounded feature space behavior modeling," in *Proc. 28th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2013, pp. 312–322.
- [20] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 289–302, Feb. 2016.
- [21] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "AccessMiner: Using system-centric models for malware protection," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 399–412.
- [22] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis," in *Proc. 5th Int. Conf. Malicious Unwanted Softw.*, Oct. 2010, pp. 23–30.
- [23] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.*, vol. 7, no. 4, pp. 247–258, Nov. 2011.
- [24] Z. Shan and X. Wang, "Growing grapes in your computer to defend against malware," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 2, pp. 196–207, Feb. 2014.
- [25] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *J. Comput. Virol. Hacking Techn.*, vol. 13, no. 1, pp. 29–46, Feb. 2017.
- [26] X. Hu, T.-C. Chiueh, and K. G. Shin, "Large-scale malware indexing using function-call graphs," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, 2009, pp. 611–620.

- [27] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Visualizat. Cyber Secur. (VizSec)*, 2011, pp. 1–7.
- [28] P. Trinius, T. Holz, J. Göbel, and F. C. Freiling, "Visual analysis of malware behavior using treemaps and thread graphs," in *Proc. 6th Int. Workshop Vis. Cyber Secur.*, Oct. 2009, pp. 33–38.
- [29] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020.
- [30] M. F. Zolkipli and A. Jantan, "A framework for malware detection using combination technique and signature generation," in *Proc. 2nd Int. Conf. Comput. Res. Develop.*, May 2010, pp. 196–199.
- [31] J. Scott, "Signature based malware detection is dead," Inst. Crit. Infrastruct. Technol., Washington, DC, USA, Tech. Rep., 2017.
- [32] K. Griffin, S. Schneider, X. Hu, and T. C. Chiueh, "Automatic generation of string signatures for malware detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer, 2009, pp. 101–120.
- [33] Y. Tang, B. Xiao, and X. Lu, "Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms," *Comput. Secur.*, vol. 28, no. 8, pp. 827–842, Nov. 2009.
- [34] B. Liu and R. Sandhu, "Fingerprint-based detection and diagnosis of malicious programs in hardware," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1068–1077, Sep. 2015.
- [35] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Y. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proc. USENIX Secur. Symp.*, vol. 4, 2009, pp. 351–366.
- [36] A. Singh, R. Arora, and H. Pareek, "Malware analysis using multiple API sequence mining control flow graph," 2017, *arXiv:1707.02691*. [Online]. Available: <http://arxiv.org/abs/1707.02691>
- [37] Ö. Aslan, R. Samet, and Ö. Ö. Tanrıöver, "Using a subtractive center behavioral model to detect malware," *Secur. Commun. Netw.*, vol. 2020, pp. 1–17, Feb. 2020.
- [38] N. Usman, S. Usman, F. Khan, M. A. Jan, A. Sajid, M. Alazab, and P. Watters, "Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics," *Future Gener. Comput. Syst.*, vol. 118, pp. 124–141, May 2021.
- [39] W. Han, J. Xue, Y. Wang, F. Zhang, and X. Gao, "APTMalInsight: Identify and cognize APT malware based on system call information and ontology knowledge framework," *Inf. Sci.*, vol. 546, pp. 633–664, Feb. 2021.
- [40] K. M. A. Alzarooni, "Malware variant detection," Ph.D. dissertation, University College London, London, U.K., 2012.
- [41] Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: Intelligent malware detection system," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2007, pp. 1043–1047.
- [42] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in *Proc. Int. Symp. Softw. Test. Anal. (ISSTA)*, 2012, pp. 122–132.
- [43] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 646–656, Mar. 2013.
- [44] E. M. Alkhateeb and M. Stamp, "A dynamic heuristic method for detecting packed malware using naive bayes," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2019, pp. 1–6.
- [45] A. Holzer, K. Johannes, and V. Helmut, "Using verification technology to specify and detect malware," in *Proc. Int. Conf. Comput. Aided Syst. Theory*. Berlin, Germany: Springer, 2007, pp. 497–504.
- [46] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Proactive detection of computer worms using model checking," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 4, pp. 424–438, Oct. 2010.
- [47] F. Song and T. Touili, "Pushdown model checking for malware detection," *Int. J. Softw. Tools Technol. Transf.*, vol. 16, no. 2, pp. 147–173, Apr. 2014.
- [48] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.
- [49] W. Huang and J. W. Stokes, "MtNet: A multi-task neural network for dynamic malware classification," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Cham, Switzerland: Springer, 2016, pp. 399–418.
- [50] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: A heterogeneous deep learning framework for intelligent malware detection," *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 265–285, Feb. 2018.
- [51] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *J. Inf. Secur. Appl.*, vol. 47, pp. 377–389, Aug. 2019.
- [52] *Microsoft Malware Classification Challenge (Big 2015)*. Accessed: Apr. 20, 2021. [Online]. Available: <https://www.kaggle.com/c/malware-classification>
- [53] A. S. Bozkir, A. O. Cankaya, and M. Aydos, "Utilization and comparison of convolutional neural networks in malware recognition," in *Proc. 27th Signal Process. Commun. Appl. Conf. (SIU)*, Apr. 2019, pp. 1–4.
- [54] S. Sriram, R. Vinayakumar, V. Sowmya, M. Alazab, and K. P. Soman, "Multi-scale learning based malware variant detection using spatial pyramid pooling network," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 740–745.
- [55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [56] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, S. Srinivas, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
- [57] *The ImageNet Dataset*. Accessed: Apr. 20, 2021. [Online]. Available: <http://www.image-net.org/>
- [58] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Procedia Comput. Sci.*, vol. 171, pp. 1251–1260, Jan. 2020.
- [59] J.-S. Luo and D. C.-T. Lo, "Binary malware image classification using machine learning with local binary pattern," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4664–4667.
- [60] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018, doi: [10.1109/tii.2018.2822680](https://doi.org/10.1109/tii.2018.2822680).
- [61] D. Gibert, "Convolutional neural networks for malware classification," M.S. thesis, Univ. Rovira Virgili, Tarragona, Spain, Oct. 2016.
- [62] A. Singh, A. Handa, N. Kumar, and S. K. Shukla, "Malware classification using image representation," in *Proc. Int. Symp. Cyber Secur. Cryptogr. Mach. Learn. Cham, Switzerland: Springer, Jun. 2019, pp. 75–92*.
- [63] X. Ma, S. Guo, H. Li, Z. Pan, J. Qiu, Y. Ding, and F. Chen, "How to make attention mechanisms more practical in malware classification," *IEEE Access*, vol. 7, pp. 155270–155280, 2019, doi: [10.1109/access.2019.2948358](https://doi.org/10.1109/access.2019.2948358).



ÖMER ASLAN received the B.Sc. degree from the Computer Engineering Department, University of Trakya, Turkey, in 2009, the M.Sc. degree in information security from The University of Texas at San Antonio, USA, in 2014, and the Ph.D. degree in cyber security field from the University of Ankara, Turkey, in 2020. He is currently a Ph.D. Researcher with the Computer Engineering Department, University of Siirt, Turkey. He is working on computer systems, information security, cyber security, malware analysis, cloud computing, and the IoT device security. He has published several articles in international journals and conferences. He has been also serving as a reviewer in some prestigious journals.



ABDULLAH ASIM YILMAZ received the B.S. degree from the Computer Engineering Department, Başkent University, Turkey, in 2010, the master's degree in computer engineering from the Graduate School of Natural and Applied Sciences, TOBB University of Economics and Technology, Turkey, in 2012, and the Ph.D. degree in computer engineering from the Graduate School of Natural and Applied Sciences, Ankara University, in 2020. He has been working as a Computer Engineer with the Republic of Turkey Ministry of Agriculture and Forestry, Ankara, Turkey, since 2011. His research interests include the artificial intelligence, image processing, computer vision, pattern recognition, software development, object tracking and recognition, and cyber security.

• • •