Proceedings of the 5th International Conference on Smart Systems and Inventive Technology (ICSSIT 2023)
IEEE Xplore Part Number: CFP23P17-ART; ISBN: 978-1-6654-7467-2

Review on Malware Classification and Malware Detection Using Transfer Learning Approach

Priya V
Department of Computer science and Engineering
PSNA College of Engineering and Technology
Dindigul,India
vpriyamalar@psnacet.edu.in

Dr. Sathya Sofia A
Department of Computer science and Engineering
PSNA College of Engineering and Technology
Dindigul,India
sathyasofia@psnacet.edu.in

2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-7467-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSSIT55814.2023.10061076

***Abstract*— The amount of malware infecting computers and other communication devices and migrating across the internet has greatly increased over the years. Numerous methods and procedures have been put forth up to this point to find and eliminate these hostile agents. However, a lot of malwares is still being developed, which can get past some cutting-edge malware detection algorithms, as new and automated malware production techniques emerge. Consequently, there is a need for the classification and detection of these antagonistic agents that have the potential to compromise the security of individuals, businesses, and a wide range of other digital assets. To develop a new improved approach for efficient zero-day malware detection, there is a compelling need to reduce bias and objectively assess these methods. This study focuses on investigating transfer learning strategies for malware picture classification, including AlexNet, VGG16, VGG19, GoogLeNet, and ResNet. Here, malware binaries are turned into grayscale images before being processed using models for classification based on transfer learning. As transfer learning uses pre-trained models, the primary objective is to save training time. In addition, an effective model for malware classification will be built in order to obtain performance that is unbiased.**

Keywords*— *Malware Detection Algorithms, Zero-Day Malware Detection, Grayscale Images, Transfer Learning, Malware Classification

I. INTRODUCTION

Malicious software is defined as any piece of code that "deliberately performs the damaging intent of an attacker" [1]. (or "malware"). These are malicious programmes meant to steal data, damage a computer, or grant unauthorised access to a network. That's why they pose a threat to everyone's ability to utilise the internet, as well as the security of servers and user privacy. Viruses, worms, trojan horses, rootkits, backdoors, botnets, spyware, and adware are just a few examples of the various types of malware that exist. Given that not all malware falls neatly into one category, it is not uncommon for a single infection to have characteristics with several different types of malware.

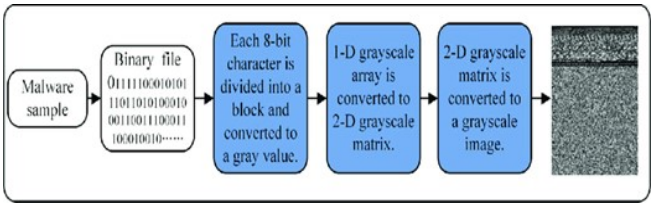

Fig.1 Malware images are converted to gray-scale images

One of the biggest and most urgent problems with Internet security today is malware. FireEye found that 47% of firms have suffered network breaches or suspicious activities involving malware in the preceding year in their survey conducted in June 2013. Malware has grown in volume, variety, and speed due to the development of the threat landscape, the range of destructive tactics, and the fluidity of threats [3]. They're getting better and better at what they do, which is attacking computers and mobile devices in new and different ways. About 100,000 new malware samples are catalogued by McAfee [4] every day. This equates to 69 new hazards per minute or one threat per second. Modern cyber threats and attacks are evolving to be more targeted, long-lasting, and mysterious as a result of the widespread availability of increasingly potent technologies.

As can be seen in Fig.2, there are key distinctions between the ways in which these two types of learning are implemented. It is clear that traditional machine learning methods try to learn each task on its own, while transfer learning methods attempt to apply knowledge learned from one task to another when the latter has less high-quality training data [59].

Fig.3 shows a comparison between two types of malwares: basic and powerful. Targeted, unknown, covert, customized, and zero-day, sophisticated malware is a stark contrast to the standard, one-time, widely distributed variety. The organisms enter the host, neutralize its defenses, and then hide, reproduce, and spread. After initialization, they communicate with their system and network for further instructions, which may include permission to decode messages, infect other systems, or perform investigations. [5]

Malware can be spread in two main ways: by social engineering, in which unsuspecting users are tricked into unwittingly running malicious programmes; and through exploiting security flaws in software like web browsers and operating systems. To get against signature-based traditional defences like firewalls, antivirus software, and gateways, malware authors use obfuscation techniques [6] such as dead code insertion, register reassignment, subroutine reordering, instruction replacement, code transposition, and code integration. Commercial anti-virus providers can't offer instantaneous defence against zero-day malware since they require time to evaluate and develop signatures.

Static and/or dynamic malware analysis methods are being utilized to go past signature-based methods' limits. Methods for analyzing malware help experts determine the

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

potential harm and the motivations behind a malicious code sample. In many cases, the insights gathered in this way are put to use in reaction to emerging malware trends or to prepare for potential future threats. Using the characteristics gleaned from malware analysis, malicious programmers are typically categorized into pre-existing families. In this research, we take a look back at how malware executables have been analyzed and categorized in the past, utilizing a wide range of.

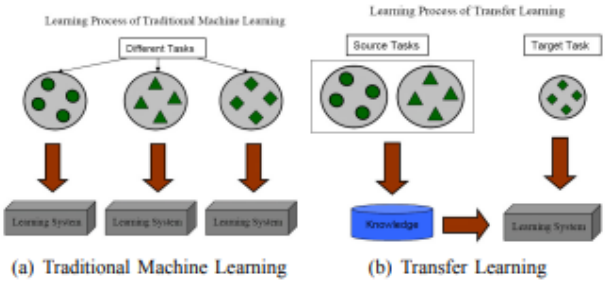

Fig.2 Different Learning Processes between Traditional Machine Learning and Transfer Learning

II Malware Exploration

Newly introduced malware must first be examined in order to comprehend the risks and intentions involved before signatures can be created. You can examine the malicious program's code or run it in a secure environment to see what it is capable of.

Static Analysis

If harmful code could be examined using static analysis rather than requiring execution, this would be a huge gain. Text signatures, byte-sequence n-grams, syntactic library calls, control flow graphs, and opcode (operational code) distribution are just some of the detection patterns used in static analysis. To perform a static analysis, you must first remove the appliance from its packaging and decipher its source code. Anybody can access the inner workings of Windows executables by using a memory dumper and disassembler. Malicious code is represented as Intel 86 assembly instructions in IDA Pro [7] and OllyDbg [8], two disassembler/debugger applications. Without this data, it will be impossible to comprehend how the malware functions and identify any trends that may lead to its makers [9,10] allow you to retrieve and dump protected code from the system's memory into a container. Utilizing this method for evaluating extremely compressed software builds is common practise. [Table 1]

Malware authors frequently utilise binary obfuscation techniques to hide their code, turning it into a compressed, custom binary file that is difficult to reverse engineer. Static analysis is no longer viable because of its high cost and lack of consistency. Malware analysis is already difficult enough without losing information like data structure and variable sizes during static analysis of binary executables (obtained by compiling source code) [11]. Since writers' methods of avoiding detection are always developing, dynamic analysis was created as a countermeasure. In order to understand where static analysis falls short, Moser et al. [12] examined its limits. Their proposed approach, which relies on code obfuscation, made it very evident that static analysis alone is insufficient for identifying and classifying malware. In addition, they debated They also recommended a mixed approach, including static and dynamic analysis, since the latter is harder to encrypt.

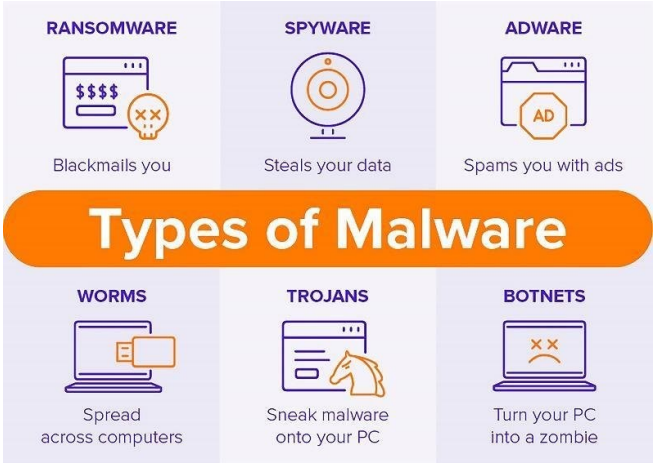

Fig.3 Different types of Malwares

Dynamic Analysis

Binaries used for malicious purposes are often obfuscated using binary obfuscation techniques, which convert them into self-compressing, uniquely constructed binary files that are challenging to reverse engineer. Nowadays, static analysis is impractical since it is too expensive and unreliable. Because information like the sizes of data structures and variables is lost during static analysis when working with binary executables (obtained by compiling source code), analyzing malware code is made even more difficult [11]. Dynamic analysis was necessary to keep up with the constantly shifting evasion strategies employed by different authors. Several static analysis techniques and their limitations were investigated by Moser et al. [12]. They demonstrated an obfuscation-based alternate to static analysis for malware classification as a result of their study. The two sides argued, too. They also proposed a hybrid approach, using both static and dynamic analysis together; in practice, dynamic analysis has become the standard because it is more resistant to decryption-based attacks.

1) Function call monitoring, function parameter analysis, information flow tracing, instruction traces, AutoStart extensibility points, and so on are all examples of dynamic analysis tools [11]. Static analysis is less efficient than dynamic analysis because it requires the programme to be decompiled before it can be analysed. Because the malware's genuine behaviour is now on display, it is more challenging to identify via static analysis. This, however, calls for a significant money and time commitment, which further adds to the obstacles inherent to development. In a variety of online environments, malicious software may behave unexpectedly, producing erroneous results rather than those intended. Malicious behaviour that is enabled only in specified situations might be concealed using the same method, making it undetectable in a simulated environment [19, 20]. It can be used to do malware analysis in real time. Such tools produce analysis reports that not

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

only paint a vivid picture of the virus's activities but also shed valuable light on it. In order for a similarity metric or feature vectors to correctly classify the malware at hand, the analysis system need an accurate representation of the malware at hand [21,22].

Due to the ever-increasing variety of malware samples, anti-virus firms must adopt an automated strategy to reduce the amount of samples requiring in-depth human examination [23]. In order to automate malware analysis and characterisation, the literature will make use of a wide variety of AI technologies, particularly machine-learning based approaches. [Table 1]

The Rise of Polymorphic and Metamorphic Malware

There's no denying that the prevalence of adaptable malware has grown over the past few years. The author, like Tripwire's lead security researcher Travis Smith, believes that the surge in this type of malware is due to the fact that it has been shown to circumvent endpoint security systems that place too much emphasis on signatures. While incidents like these are less common today, there has been a recent uptick in the security community's desire to share threat intelligence. File hashes are the most convenient means of disseminating this data. When defenders share threat data, it is most useful when they have a consistent, unique identifier for the malware in question; if the malware in question changes its file hash with each infection, sharing that information is far less useful [24].

The usual obfuscation approaches, such as polymorphic and metamorphic methods, are used by the new generation of other malware to prevent detection. This kind of malware can rapidly get past firewalls, security software, and other types of protection software that operate in kernel mode. Some malware samples can also constantly exhibit the traits of several other malware classes. Since all malware cannot be detected using a single detection method, it is essentially impossible. The following list explains about the two popular obfuscation methods:

- **Polymorphic**: Malware employs a separate key to encrypt and decrypt data when using the polymorphic approach, just as it does when use the oligomorphic method. The payload component can be encoded in levels and involves multiple versions of the decoder. As a result, polymorphic malware is more challenging to identify than oligomorphic malware.

- **Metamorphic:** The metamorphic approach does not employ encryption. Instead, it employs dynamic code concealment, in which the opcode modifies for each execution of the malicious process. Such virus has an extremely elusive signature, making it exceedingly challenging to identify fresh copies.

III Machine Learning for Identifying and Classifying Malware

The goal of many machine learning algorithms is to recognise and categorise new samples into known malware families or to identify samples that demonstrate problematic behaviour for further examination. These are the published ones that are highlighted here.

Schultz et al. [25] initially advocated using data mining to detect malicious software. Portable Executable (PE), strings, and byte sequences were used to classify malware.

In general, malware images are variable in size, the existing works resize all the images to a uniform size and feed them to DL. In this aspect, we believe that the resizing would affect meaning of the malware codes.

The PE method extracts metadata from PE files via the DLL information provided therein (such as a list of DLLs utilised by the binary, a list of DLL function calls, and the number of different system calls used within each DLL). It is possible to recover text strings from file formats using software. In the byte sequence approach, an n-byte sequence is taken directly from an executable. There were 3,265 malicious files and 1,001 safe ones in the database they used. The rule induction technique Ripper [26] was used to analyse the DLL data. The Multinomial Naive Bayes algorithm fed n-grams of byte sequences to the Naive Bayes algorithm, which is used to train models to find patterns in string data. The most that the Naive Bayes method can do when used to categorise strings is 97.11 percent. For malware detection, data mining techniques were proven to be twice as successful as signature-based methods.

The outcomes were enhanced by the additional work done by Kolter et al. [27]. N-grams and data mining methods were utilised instead of looking for non-overlapping byte sequences in order to find malicious executables. Naive-Bayes, Support Vector Machine, and Decision Tree classifiers, as well as their enhanced variants, were utilised. Results from a variety of data categorization algorithms showed that improved decision trees performed best.

Malware binaries can be seen and categorised using an approach presented by Nataraj et al. [28], which employs image processing techniques to show malware binaries as grayscale images. The K-nearest neighbour method coupled with the Euclidean distance metric is used to categorise malware. While the method is quick in comparison to other malware analysis techniques, it is vulnerable to being misled by an attacker who employs workarounds due to its reliance on global image-based features. For instance, a hacker may rearrange the bits in a binary or add padding to make it longer.

In order to evaluate the merits of dynamic analysis vs analysis based on binary textures, the authors of [29] employed an image processing method. They discovered that this method of categorization outperformed dynamic analysis in terms of speed, scalability, and precision. They also discovered that this method works just as well on frozen as it does on fresh samples, indicating that it can be used to reliably diagnose a wide variety of diseases. One major drawback of this approach is that it can be used by malicious code obfuscation to conceal textures from well-trained attackers.

In order to automatically categorise malware based on structural data, Kong et al. [30] created a method (the function call graph of malicious code). Discriminating distance metric learning may evaluate the degree of similarity between two dangerous programmes for each malware sample by grouping together malware samples that belong to the same family and keeping the other clusters separate by a marginal distance. Next, they employed a library of classifiers that, by comparing distances between

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

pairs of malwares, can determine which families are most closely related to one another.

For their classification of Trojans, Tian et al. [31] looked at the frequency of function lengths. A function's length can be determined by counting the number of bytes it uses. Their findings imply that function length and frequency, when paired with other criteria, can be used to classify malware quickly and at scale. To combat the problem that obfuscated files frequently lack a string of words or phrases, they utilized printable string data from the executables [32]. Malware was sorted using the ML instruments provided by the WEKA [33] library.

Santos et al. [34], the objective is to use a large dataset with both labelled and unlabeled instances to train a machine learning classifier. Here, we employ a semi-supervised learning approach termed Learning with Local and Global Consistency (LLGC), which makes use of both labelled and unlabeled data and yields a solution that accounts for the underlying structure in both types of scenarios. An n-gram is used to illustrate the functioning of the programmes. In addition, we analyse how changing the sample size of the labelled data affects the model's accuracy and determine an optimal value. This work's key contribution is a lower total number of labelled examples. When compared to existing supervised learning algorithms like those reported in [27] and [35], the proposed semi-supervised technique produces inferior results (over 90% accuracy). In [36], a similar strategy for detecting unknown illnesses called collaboration learning is given. This subfield of semi-supervised learning in machine learning is a method for improving the classification of data that has been inadequately labelled. Collective classification techniques enable the training of multiple machine learning classifiers on a dataset consisting of both labelled and unlabeled samples. There is research that suggest that unsupervised learning is more efficient than supervised learning when it comes to labelling data.

To identify worms in the wild, Siddiqui et al. [37] used machine learning in conjunction with a variable length instruction sequence. They can identify the compilers or packers used before they even begin deconstructing the files. Two types of machine learning models, decision trees and random forests, were utilized to accomplish sequence reduction and classification. There were a total of 2774 files included for this investigation; 1444 were worms and 1330 were safe.

Researchers are increasingly leaning toward using dynamic approaches to malware classification due to their increased precision and efficiency.

A methodology for analyzing the behavior of malware was presented by Zolkipli et al. [38]. They used security tools like Amun and Honey Clients to collect malware. Researchers can analyze the behavior of malware samples by running them in virtual machines using tools like CW Sandbox [20] and Anubis [21]. Human behavior is analyzed by both of these analyzers so that each user can get tailored results. The malicious software is then classified as either Trojans or worms. Human analysis modification is not practical for the serious and sophisticated real-time traffic of today, which is a major downside of this approach.

In [39], Rieck et al. advocated utilizing machine learning to automatically analyze malware behavior. This framework collected numerous malware samples and examined them in a virtual lab. They use a vectorization of the observed behavior to implement the learning procedures. Malware families are studied together because they share similar characteristics. We can then potentially use these unanticipated subtypes to classify previously unknown forms of malware.

Anderson et al. [40] proposed Malware Detection System using Graph Analysis for Dynamically Analyzing Instruction Traces for Identifying Malware. To achieve this, we employ a customised version of the Ether malware analysis framework [23] to filter and organise the raw data. The transition probability along the Markov chain is dependent on the 2-grams (treated as a graph). The generation of a similarity matrix for training set instances is a common application of graph kernels. When multiplying the spectral kernel and the Gaussian kernel (which respectively define the local and global similarity of the graph's edges), we get a kernel matrix. In order to classify test data, a support vector machine must be trained, and the kernel matrix is an essential input in this process. However, despite its obvious benefits, this strategy is rarely employed due to the high computational cost associated with it.

Bayer et al. [41] analysed the behaviour of a large group of potentially malicious binaries and found that they could be quickly sorted into different groups. Each sample is run through Anubis [21] to generate execution traces, and this is the recommended method. While engaged in this activity, Anubis gained the ability to diffuse taint, allowing him access to previously inaccessible data [42]. Tian et al. [43] automatically collected API call sequences from VM-running executables. The WEKA library's classifiers [33] were used to separate malware into families, label benign files, and determine which ones were malicious. By applying their algorithm to a dataset containing 1368 malwares and 456 cleawares, the researchers demonstrated its efficacy, attaining an accuracy of above 97%.

Antivirus (AV) products, as stated by Biley et al. [44], classify malware in various ways depending on the product, are incomplete when comparing different types of malwares, and lack semantic transparency. They devised a method for dividing malicious software into several categories based on the ways in which each one modifies the operating system.

Binaries run in a virtual computer and install Windows XP. The virtual machine has a firewall installed to prevent unauthorized access to the application. Malware leaves a digital trail of activity by creating new files, processes, and network connections. Over the course of six months, 3,700 malware samples were gathered, and this method was utilized to automatically categorized and analyze them. In addition, they look at how consistent, detailed, and succinct the clusters are while discussing AV products. Since the virtualized system's capabilities and environment are maintained throughout the research, it is possible that the findings are confined.

Malware classification using the discovery of maximal component subgraphs was proposed by Park et al. [45]. When malware samples are executed in a sandbox, information about the system calls and the values of their arguments is captured. For a more thorough analysis of the programmes under consideration, the greatest common subgraph can be determined. However, this approach has its limitations, as certain known malware samples can still

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

dodge detection by bypassing analysis and gaining access to kernel-mode capabilities in other ways.

An effective method for detecting malware was recently proven by Firdausi et al. [46]. First, malware samples are run with Anubis [21] to see how they behave in a virtual environment. The obtained reports are converted into sparse vector models before being used in the machine learning categorization process..

It has been suggested that malware can be automatically organised into clans using the network behaviour classification scheme provided by Nari et al. [47]. This architecture can analyse pcap files containing network traces to determine traffic patterns. After that, a behaviour graph is constructed to show the malware's network activity and the interdependencies between network flows. These behaviour graphs are mined for information such as graph size, root out-degree, average out-degree, maximum out-degree, and the number of distinct nodes. To classify malware based on these traits, we use the J48 decision tree and other classification methods from the WEKA toolkit [33].

The Lee et al. [48] technique use a machine learning mechanism to categorizes malware. System calls and associated parameters can be monitored when data samples are executed in a simulated environment. Information on how the sample uses the system, such as registry keys, new files created, and network activity, is used to create a behavioral profile. K-medoids are used to organize samples, but only after the similarity between two profiles has been determined and used to form clusters. Following the completion of the training phase, unidentified samples from new collections are assigned to the cluster holding the medoid that is physically nearest to them.

Similarly, Islam et al. [50] classified executables as malicious or safe based on static and dynamic properties. The study makes advantage of static facts, such as the distribution of function lengths and information about printed strings, to draw conclusions. The names of API functions and the values of API parameters are two instances of this type of variable. Of the 2939 executables employed in the investigation, only 541 were deemed secure. When applied to a collection of interrelated traits, all meta-classifiers, including the unflinchingly superior meta-RF, are capable of reaching their full accuracy. They found that their unified method was more precise than the others they had tried.

The static binary, the disassembled binary file, the control flow graph, a dynamic instruction trace, a system call trace, and a feature vector based on file metadata are all used in the method developed by Anderson et al. [51]. To analyse the binary file, the disassembled file, and the two dynamic traces, we employ Markov chain graph-based kernels. We employ a graphlet kernel for the CFG and a regular Gaussian kernel for the fv that keeps track of file data. To categorise data as safe or risky, support vector machine classifiers use a weighted combination of the data sources determined through multiple kernels learning. The test found an accuracy of 98.07%, with the dataset included 776 benign instances and 780 malwares.

To categorise malware grey-scale images, Sumit S. Lad et al. [52] propose employing a Convolutional Neural Network model that can take advantage of pre-processing and augmentation approaches. To evaluate the efficacy of various malware classification methods, researchers have used the Malimg dataset, which contains 9339 samples from 25 malware families. The author developed the CNN model by integrating data pre-processing and augmentation techniques on this dataset in order to circumvent the computing resource and processing time limitations of typical CNN models. The results (98.03%) are remarkable when compared to those of other existing CNN designs like VGG16 (96.96%), InceptionV3 (97.22%), Xception (97.56%), and ResNet50 (97.56%). (3.97 out of 4) The proposed technique can be brought to market more quickly than alternative CNN designs (1 hour). Several models are available, such as ResNet50 (1.10hr), VGG16 (1.10hr), InceptionV3 (9.7hr), and Xception (7.5hr) (19hrs). To accomplish this sorting, we employ a hybridization strategy. A multi-class support vector machine (SVM) built using the binary SVM classifier and the linear SVC kernel achieves a 99.59 percent accuracy rate when applied to malware samples.

The malware classification solution provided by Prima Bouchaib et al. [53] makes use of transfer learning and pre-trained Deep Learning models on large image datasets. As both the volume and variety of dangerous programs have surged in recent years, there has been a growing need for better automatic malware identification and categorization. The neural network methodology has been improving rapidly recently, and it may soon be able to beat older machine learning approaches such as Hidden Markov Models and Support Vector Machines (SVM). On tasks like image classification, convolutional neural networks (CNNs) have surpassed more conventional learning approaches. Given this success, the author proposes a Convolutional Neural Network (CNN)-based architecture for malware classification. The deep network was trained using the VGG16 layers from the ImageNet dataset, and the final fully connected layer was tweaked to improve malware classification based on family. Malicious binaries are depicted as monochrome images. The author's evaluations show that this method consistently achieves 98% accuracy on the MALIMG dataset.

In order to better highlight the section distribution data in the converted malware image, Mao Xiao et al. [54] present a novel visualisation technique based on coloured label boxes (CoLab) for locating the PE file's components. MalCVS, an abbreviation for "Malware categorization using Co-Lab image, VGG16, and Support Vector Machine," is another option. Malware from VX Heaven, Virus share, and the Microsoft Malware Classification Challenge dataset were used to demonstrate MalCVS's ability to reliably and successfully group malware into families in a series of experiments. MalCVS has an average accuracy of 96.59 and 98.94% when used on these two datasets.

Danish et al. [55], it makes use of CNN-based deep learning architecture to expand our understanding of malware families, discover new variants, and improve malware detection. Thanks to its innovative CNN architecture, the system makes it easy to recognise and categorise various forms of malware. The framework is able to transform malicious code into high-resolution colour images.

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

This study also demonstrates that a dataset with colour photographs of viruses is more reliable than one with only black-and-white images.

MDMC is a byte-level malware classification strategy introduced by Baoguo Yuan et al., which makes use of deep learning and Markov pictures. [56] A vital part of the MDMC, bytes transfer probability matrices are employed to transform harmful binaries into Markov pictures. The images from the Markov chain are then fed into a deep convolutional neural network and sorted into categories. Evaluations make use of the Microsoft and Drebin malware datasets. Using the same model, the average MDMC accuracy for the two datasets is 99.264%, whereas it is only 97.3646% when using a different model. Furthermore, testing with different dataset splits demonstrate that MDMC performs better than GDMC.

A new technique for detecting unknown malware was developed by Hashem Hashemi et al. [57], which involves analyzing the micro-patterns embedded within executable files. The author of this piece offered a workaround that employs a well-known technique from the field of machine vision in order to successfully extract the necessary micro-patterns. The new procedure entails the following measures: To identify malicious code, researchers are first converting executables into digital photographs, then extracting visual aspects of the executables from the photos, and finally using machine learning algorithms. This strategy is predicated on the assumption that malicious and safe files exhibit distinct behavioral and operational features that can be recognized from one another by means of individual micro-patterns. Therefore, the method used in this research involves extracting micro-patterns from executable files and classifying them based on their textures.

Jin-Young Kim et al. [58] offer a unique technique they call transferred deep-convolutional generative adversarial network (tDCGAN), which generates false malware and learns to identify it from actual malware. This can be used to accurately detect malware, even zero-day attacks. These data were generated using a random process and share many features with real-world samples while differing in a few key respects. To learn a wide range of malware characteristics from both actual and manipulated data, the tDCGAN is founded on a deep autoencoder (DAE), which identifies the relevant features and stabilizes the GAN training. The DAE first learns about the virus, then generates generic data, and then uses its own expertise to train a GAN. The malware detector receives malware recognition abilities from the trained discriminator via transfer learning. We show that the tDCGAN model improves on average classification accuracy and has more robust learning stability than its rivals (95.74 percent). Additionally, it provides the best protection against mock zero-day assaults of any existing solution.

TABLE I. OUTLINE OF THE SURVEYED WORKS

Work	Year	Analysis	Method(s)	ML Technique(s)
[25]	2001	Static	Data Mining Methods	DLL
[26]	1995	Static	data mining method	Naive Bayes
[27]	1995	dynamic	Naive Bayes + strings	boosted decision tree
[28]	2011	dynamic	Texture analysis	KNN
[29]	2011	Image processing technique	Euclidean distance method	KNN
[30]	2013	Automated Malware classification	Function call graph	KNN and SVM
[31]	2008	automated classification system	IDA	Database functions
[32]	2009	Static	Random forest classification	SVM
[34]	2009	Hybrid	LLGC	Semi Supervised ML
[35]	2008	dynamic	Text categorization	ANN SVM NB
[36]	2011	Hybrid	Collective learning approach	Semi Supervised ML
[37]	2009	Static	Data Mining Methods	decision tree and random forest
[38]	2011	Dynamic	Signature based method	AI
[39]	2011	Dynamic	Clustering	SVM
[40]	2011	Dynamic	multiple kernel learning method	SVM
[41]	2009	Dynamic	Clustering	Unsupervised ML
[42]	1998	Dynamic	LSH	KNN
[43]	2010	Dynamic	API	WEKA SVM
[44]	2007	Dynamic	Behaviour based clustering	Naive
[45]	2010	Dynamic	Graph Theory	AutoML
[46]	2010	Dynamic	Data Mining Methods	SVM KNN
[47]	2013	automated classification system	Reference clustering	WEKA
[48]	2006	Hybrid	System call	KNN
[49]	2013	Hybrid	Behaviour based Analysis	Decision Tree KNN SVM NB
[50]	2013	Hybrid	Coarse-grained modelling	SVM, NN, DT, RF
[52]	2012	Hybrid	CNN	SVM
[53]	2020	Hybrid	CNN	Deep Learning
[55]	2020	Hybrid	data augmentation	Deep Learning
[57]	2017	Static	K-fold Cross Validation	Markov Chains
[58]	2018	Dynamic	SSIM method	Deep Convolutional approach

IV Expressions for the evaluation metrics based on the transfer learning algorithm.

Pan and Yang provide a conceptual framework for learning transfer in their study "A Survey on Transfer Learning" [59], which they do through the use of domain, task, and marginal probability. The structure is described as follows:

Domain (D) is a two-element tuple consisting of feature space x and marginal probability, P(X), where X is a sample data point. Thus, we can represent the domain mathematically as D = {□, P(X)}

A Domain consists of two components: D= {□, P(X)}

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

Feature space: □
Marginal distribution: P(X), X = { x_1, … , x_n }, x_i ∈ □

Here x_i represents a specific vector as represented in the above depiction. A task, T, on the other hand, can be defined as a two-element tuple of the label space, γ, and objective function, η. The objective function can also be denoted as P(γ| X) from a probabilistic view point.

Conclusion:

Modern malware is a serious danger to the online community and to computer infrastructure. Viruses that have not been found and described cannot be detected by an antivirus program that uses a signature database. In response to the constant efforts of the malicious software industry to evade static analysis, dynamic analysis was developed to execute the malware sample in a controlled environment and watch its behavior. Reports generated by dynamic analyses are often consolidated into behavior profiles that may be grouped to classify samples into logical groups based on their shared characteristics. One significant disadvantage of dynamic analysis is the time and effort it requires. A solution combining static and dynamic data for malware identification has been developed by researchers to solve the trade-off between processing speed and identifying disguised malware. Strategies for analyzing, detecting, and classifying malware are explored at length. The papers in Table 1 are organized by different methods for analyzing malware. The dynamic and highly asymmetrical nature of network data presents difficulties for machine learning techniques used to detect and classify malware. These should be updated so that their potential can be put toward addressing cybersecurity issues.

References

[1] Bayer, U., Moser, A., Kruegel, C. and Kirda, E. (2006) Dynamic Analysis of Malicious Code. Journal in Computer Virology, 2, 67-77.

[2](2013) The Need for Speed: 2013 Incident Response Survey, FireEye. http://www.inforisktoday.in/surveys/2013-incident-response-survey s-18

[3](2012) Addressing Big Data Security Challenges: The Right Tools for SmartProtection.http://www.trendmicro.com/cloudcontent/us/pdfs/business/white-papers/wp_addressing-big-data-scurity-challenges.

[4](2013)Infographic:TheStateofMalware.http://www.mcafee.com/in/security-awareness/articles/state-of-malware-2013.aspx

[5](2013) Next Generation Threats. http://www.fireeye.com/threat-protection/

[6] You, I. and Yim, K. (2010) Malware Obfuscation Techniques: A Brief Survey. Proceedings of International conference on Broadband, Wireless Computing, Communication and Applications, Fukuoka, 4-6 November 2010, 297-300. http://dx.doi.org/10.1109/BWCCA.2010.85

[7]IDAPro.https://www.hexrays.com/products/ida/support/download_freeware.shtml

[8] OllyDbg. http://www.ollydbg.de/

[9]LordPE.http://www.woodmann.com/collaborative/tools/index.php/LordPE

[10]OllyDumphttp://www.woodmann.com/collaborative/tools/index.php/OllyDump

[11] Egele, M., Scholte, T., Kirda, E. and Kruegel, C. (2012) A Survey on Automated Dynamic Malware- Analysis Techniques and Tools. Journal in ACM Computing Surveys, 44, Article No. 6.

[12] Moser, A., Kruegel, C.and Kirda, E. (2007) Limits of Static Analysis for Malware Detection. 23rd Annual Computer Security Applications Conference, Miami Beach, 421-430.

[13](2014)ProcessMonitor.http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

[14] Capture BAT. https://www.honeynet.org/node/315

[15](2014)ProcessExplorer.http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx

[16] Process Hacker replace. http://processhacker.sourceforge.net/

[17] Wireshark. http://www.wireshark.org/

[18] Regshot. http://sourceforge.net/projects/regshot/

[19] Norman Sandbox. http://sandbox.norman.no

[20] Willems, C., Holz, T. and Freiling, F. (2007) Toward Automated Dynamic Malware Analysis Using Cwsandbox. IEEE Security & Privacy, 5, 32-39. http://dx.doi.org/10.1109/MSP.2007.45

[21] Anubis. http://anubis.iseclab.org/

[22] Bayer, U., Kruegel, C. and Kirda, E. (2006) TT Analyze: A Tool for Analyzing Malware. Proceedings of the 15th European Institute for Computer Antivirus Research Annual Conference.

[23] Dinaburg, A., Royal, P., Sharif, M. and Lee, W. (2008) Ether: Malware Analysis via Hardware Virtualization Extensions. Proceedings of the 15th ACM Conference on Computer and Communication Security, CCS'08, Alexandria, 27-31 October 2008, 51-62.

[24] ThreatExpert. http://www.threatexpert.com/submit.aspx

[25] V. R, R. S, V. K and S. P, "Integrated Communal Attentive & Warning System via Cellular Systems," 2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC), 2022, pp. 370-375, doi: 10.1109/ICESIC53714.2022.9783481

[26] Cohen, W. (1995) Fast Effective Rule Induction. Proceedings of 12th International Conference on Machine Learning,San Francisco, 115-123.

[27] Kolter, J. and Maloof, M. (2004) Learning to Detect Malicious Executables in the Wild. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 470-478.

[28] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B. (2011) Malware Images: Visualization and Automatic Classification. Proceedings of the 8th International Symposium on Visualization for Cyber Security, Article No. 4.

[29] Nataraj, L., Yegneswaran, V., Porras, P. and Zhang, J.(2011) A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis. Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, 21-30.

[30] V. K, K. Jayashree, V. R and B. Rajendiran, "Forecasting Methods and Computational Complexity for the Sport Result Prediction," 2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC), 2022, pp. 364-369, doi: 10.1109/ICESIC53714.2022.9783514.

[31] Tian, R., Batten, L. and Versteeg, S. (2008) Function Length as a Tool for Malware Classification. Proceedings of the 3rd International Conference on Malicious and Unwanted Software, Fairfax, 7-8 October 2008, 57-64.

[32] Tian, R., Batten, L., Islam, R. and Versteeg, S. (2009) An Automated Classification System Based on the Strings of Trojan and Virus Families. Proceedings of the 4th International Conference on Malicious and Unwanted Software, Montréal, 13-14 October 2009, 23-30.

[33] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. (2009) The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter, 10-18.

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.

[34] Santos, I., Nieves, J. and Bringas, P.G. (2011) Semi-Supervised Learning for Unknown Malware Detection. International Symposium on Distributed Computing and Artificial Intelligence Advances in Intelligent and Soft Computing, 91,415-422.

[35] Moskovitch, R., Stopel, D., Feher, C., Nissim, N. and Elovici, Y. (2008) Unknown Malcode Detection via Text Categorization and the Imbalance Problem. Proceedings of the 6th IEEE International Conference on Intelligence and Security Informatics, Taipei, 17-20 June 2008, 156-161.

[36] Santos, I., Nieves, J. and Bringas, P.G. (2011) Collective Classification for Unknown Malware Detection. Proceedings of the International Conference on Security and Cryptography, Seville, 18-21 July 2011, 251-256.

[37] Siddiqui, M., Wang, M.C. and Lee, J. (2009) Detecting Internet Worms Using Data Mining Techniques. Journal of Systemics, Cybernetics and Informatics, 6, 48-53.

[38] Zolkipli, M.F. and Jantan, A. (2011) An Approach for Malware Behavior Identification and Classification. Proceeding of 3rd International Conference on Computer Research and Development, Shanghai, 11-13 March 2011, 191-194.

[39] Rieck, K., Trinius, P., Willems, C. and Holz, T. (2011) Automatic Analysis of Malware Behavior Using Machine Learning. Journal of Computer Security, 19, 639-668.

[40] Anderson, B., Quist, D., Neil, J., Storlie, C. and Lane, T. (2011) Graph Based Malware Detection Using Dynamic Analysis. Journal in Computer Virology, 7, 247-258. http://dx.doi.org/10.1007/s11416-011-0152-x

[41] Bayer, U., Comparetti, P.M., Hlauschek, C. and Kruegel, C. (2009) Scalable, Behavior-Based Malware Clustering. Proceedings of the 16th Annual Network and Distributed System Security Symposium.

[42] Indyk, P. and Motwani, R. (1998) Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. Proceedings of 30th Annual ACM Symposium on Theory of Computing, Dallas, 24-26 May 1998, 604-613.

[43] Tian, R., Islam, M.R., Batten, L. and Versteeg, S.(2010) Differentiating Malware from Cleanwares Using Behavioral Analysis. Proceedings of 5th International Conference on Malicious and Unwanted Software (Malware), Nancy, 19-20 October 2010, 23-30.

[44] Biley, M., Oberheid, J., Andersen, J., Morley Mao, Z., Jahanian, F. and Nazario, J. (2007) Automated Classification and Analysis of Internet Malware. Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection, 4637,178-197. http://dx.doi.org/10.1007/978-3-540-74320-0_10

[45] Park, Y., Reeves, D., Mulukutla, V. and Sundaravel, B. (2010) Fast Malware Classification by Automated Behavioral Graph Matching. Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research,Article No. 45.

[46] Firdausi, I., Lim, C. and Erwin, A. (2010) Analysis of Machine Learning Techniques Used in Behaviour Based Malware Detection. Proceedings of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT), Jakarta, 2-3 December 2010, 201-203.

[47] Nari, S. and Ghorbani, A. (2013) Automated Malware Classification Based on Network Behavior. Proceedings of International Conference on Computing, Networking and Communications (ICNC), San Diego, 28-31 January 2013,642-647.

[48] Lee, T. and Mody, J.J. (2006) Behavioral Classification. Proceedings of the European Institute for Computer Antivirus Research Conference (EICAR'06).

[49] Kirubasri, G. (2021). A contemporary survey on clustering techniques for wireless sensor networks. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(11), 5917-5927.

[50] Islam, R., Tian, R., Battenb, L. and Versteeg, S. (2013) Classification of Malware Based on Integrated Static and Dynamic Features. Journal of Network and Computer Application, 36, 646-556. http://dx.doi.org/10.1016/j.jnca.2012.10.004

[51] Anderson, B., Storlie, C. and Lane, T. (2012) Improving Malware Classification: Bridging the Static/Dynamic Gap.Proceedings of 5th ACM Workshop on Security and Artificial Intelligence (AISec), 3-14

[52] Sumit S. Lad, Amol C Adamuthe (2020) Malware Classification with Improved Convolutional Neural Network Model I. J. Computer Network and Information Security, 2020, 6, 30-43 Published Online December 2020 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijcnis.2020.06.03

[53] PRIMA Bouchaib , BOUHORMA Mohamed ,(2020) Using Transfer Learning for Malware Classification. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLIV-4/W3-2020, 2020 5th International Conference on Smart City Applications, 7–8 October 2020,

[54] M Xiao, C Guo, G Shen, Y Cui, C Jiang, (2021) Image-based malware classification using section distribution information.Computers and Security Volume 110 Issue C Nov2021 https://doi.org/10.1016/j.cose.2021.102420

[55] Danish Vasan , Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei , Qin Zheng , IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture, journal homepage: www.elsevier.com/locate/comnet

[56] Baoguo Yuan, Junfeng Wang, Dong Liu, Wen Guo, Peng Wu , Xuhua Bao Byte-level malware classification based on markov images and deep learning, journal homepage: www.elsevier.com/locate/cose

[57] Kong, D. and Yan, G. (2013) Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification. Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, 347-348. https://doi.org/10.1007/s11416-018-0314-1

[58] Jin-Young Kim, Seok-Jun Bu, Sung-Bae Cho, Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders, Information Sciences 460–461 (2018) 83–102.

[59] A Survey on Transfer Learning Sinno Jialin Pan and Qiang Yang Fellow, IEEE. IEEE transactions on knowledge and data engineering. 1041-4347, 2009.

[60] Schultz, M., Eskin, E., Zadok, F. and Stolfo, S. (2001) Data Mining Methods for Detection of New Malicious Executables. Proceedings of 2001 IEEE Symposium on Security and Privacy, Oakland, 14-16 May 2001, 38-49.

Authorized licensed use limited to: University of West London. Downloaded on April 29,2023 at 17:23:46 UTC from IEEE Xplore. Restrictions apply.