Journal of
Cybersecurity and Privacy

MDPI

Review

A Survey of the Recent Trends in Deep Learning Based Malware Detection

Umm-e-Hani Tayyab [1], Faiza Babar Khan [1], Muhammad Hanif Durad [1], Asifullah Khan [2,3,4,*] and Yeon Soo Lee [5,*]

[1] CIPMA Lab, DCIS, Pakistan Institute of Engineering & Applied Sciences, Nilore, Islamabad 45650, Pakistan
[2] Pattern Recognition Lab (PRLab), Department of Computer & Information Sciences, Pakistan Institute of Engineering & Applied Sciences, Nilore, Islamabad 45650, Pakistan
[3] PIEAS Artificial Intelligence Center (PAIC), Pakistan Institute of Engineering & Applied Sciences, Nilore, Islamabad 45650, Pakistan
[4] Deep Learning Lab, Center for Mathematical Sciences (CMS), Pakistan Institute of Engineering & Applied Sciences, Nilore, Islamabad 45650, Pakistan
[5] Department of Biomedical Engineering, College of Medical Science, Catholic University of Daegu Hayangro, 13-13, Hayang-Eup, Gyoungsan-si 38430, Gyoungsangbuk-do, Korea
* Correspondence: asif@pieas.edu.pk (A.K.); yeonsoolee@cu.ac.kr (Y.S.L.)

Abstract: Monitoring Indicators of Compromise (IOC) leads to malware detection for identifying malicious activity. Malicious activities potentially lead to a system breach or data compromise. Various tools and anti-malware products exist for the detection of malware and cyberattacks utilizing IOCs, but all have several shortcomings. For instance, anti-malware systems make use of malware signatures, requiring a database containing such signatures to be constantly updated. Additionally, this technique does not work for zero-day attacks or variants of existing malware. In the quest to fight zero-day attacks, the research paradigm shifted from primitive methods to classical machine learning-based methods. Primitive methods are limited in catering to anti-analysis techniques against zero-day attacks. Hence, the direction of research moved towards methods utilizing classic machine learning, however, machine learning methods also come with certain limitations. They may include but not limited to the latency/lag introduced by feature-engineering phase on the entire training dataset as opposed to the real-time analysis requirement. Likewise, additional layers of data engineering to cater to the increasing volume of data introduces further delays. It led to the use of deep learning-based methods for malware detection. With the speedy occurrence of zero-day malware, researchers chose to experiment with few shot learning so that reliable solutions can be produced for malware detection with even a small amount of data at hand for training. In this paper, we surveyed several possible strategies to support the real-time detection of malware and propose a hierarchical model to discover security events or threats in real-time. A key focus in this survey is on the use of Deep Learning-based methods. Deep Learning based methods dominate this research area by providing automatic feature engineering, the capability of dealing with large datasets, enabling the mining of features from limited data samples, and supporting one-shot learning. We compare Deep Learning-based approaches with conventional machine learning based approaches and primitive (statistical analysis based) methods commonly reported in the literature.

Keywords: malware; machine learning; Deep Learning; few shot learning; Cyber Attacks

Citation: Tayyab, U.-e.-H.; Khan, F.B.; Durad, M.H.; Khan, A.; Lee, Y.S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. *J. Cybersecur. Priv.* **2022**, *2*, 800–829. https://doi.org/10.3390/jcp2040041

Academic Editor: Hossein Saiedian

Received: 11 August 2022
Accepted: 22 September 2022
Published: 28 September 2022

1. Introduction

According to the Panda Security report [1], hackers are involved in creating around 230,000 malware samples daily, a number expected to grow in the coming years. According to an FBI report [2], ransomware is considered to be one of the fastest-growing threats, with over 4000 ransomware attacks occurring every day since 2016. Ransomware is capable of targeting home users, small and large businesses, and has the potential to cause the loss of

sensitive information temporarily or permanently according to [3]. Critical infrastructure is the most luring target for the ones who are well versed with the damages that can be caused by ransomware. Ransomware is the type of malware that uses the encryption module to encrypt the data and makes it unusable for the user [4]. Over the past few decades ransomware has affected not only small businesses but has victimized big companies like FedEx, Nissan, Russian and German railways, and NHS organizations in the UK according to Ref. [5]. According to a report [6] produced by Kaspersky, spam emails are the constant features of phishing, and this trend is unlikely to change soon. Symantec's Internet Security Threat report of 2019 [7] stated that supply chains remained a soft target, with attacks increasing by 78% in 2019 compared to the previous year. The same report mentions blocking 69 million cryptojacking events in 2018, four times increase compared to 2017. Small businesses are severely affected by cyber-attacks and according to statistics in 2019, 40% of small companies were attacked, out of which only 13% could detect and mitigate the attacks [8]. Due to economic losses caused by cyber-attacks, 60% of small companies collapsed. Accenture reports that the US $2.4 M is spent by companies to support malware detection and defense from web-based attacks. Cyber-attacks have heavily created chaos in critical infrastructure as well. State-sponsored attackers had been found involved in launching attacks over industrial control systems lately. One of the biggest examples of such malware is Stuxnet which was designed to choke the working of the Iranian Nuclear Power Plant's centrifuges [9,10]. Cyber physical systems are almost applied in all critically important areas such as traffic lights, health care, power generation, water industry, transportation system, etc. [11]. Communication of these cyber physical systems with network make them vulnerable and many stealthy attacks launching different malicious payloads can be expected easily by looking at the statistics [12]. Malfunctioning of such significantly important systems can cause severe accidents and damages. To protect the cyber physical systems working in all crucial areas, researchers have been trying their level best to device an anti-malware system that can protect them. There are many tools and anti-virus products available in the market for the detection of malware and cyberattacks, however, they have their inherent shortcomings. Anti-virus products work over the signatures of malware, and the signature database needs to be constantly updated. This technique also does not work for zero-day attacks and for the new variants of existing malware (which can have a different signature).

Various strategies have been implemented to speed up the real time detection of different types of malware as explained in Appendix A.1 so that the effect of the malware can be mitigated. A taxonomy of malware analysis is explained in Appendix A.2 and is illustrated in Figure 1: static analysis focuses on detecting a malicious file without executing it, whereas dynamic analysis works by first executing the file. A hybrid strategy involves a combination of both static and dynamic analysis

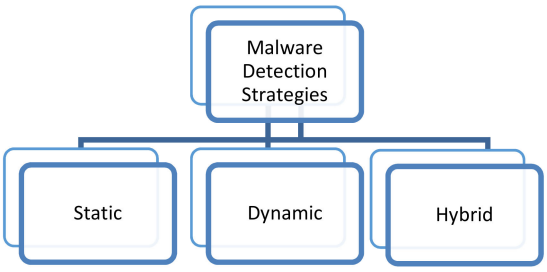

Figure 1. Taxonomy of Malware Analysis.

Various approaches have been reported in the literature to detect malicious behavior and files, involving: (i) statistical data analysis-based research for malware classification; (ii) machine learning methods (including Deep Learning) for malware detection and identification.

The key motivation has been to develop the capability of detecting and identifying malware in a cost-effective manner, and in real-time so that the effects of malware can be mitigated.

Different survey papers have been written in the domain of cyber security surveying the work done in malware detection. Unlike other survey papers, our paper is not focusing on a single strategy to be reported in this literature survey, instead, we have accumulated the research trends in malware detection from various application areas of data science as well as AI. Table 1 shows the comparison between our work and other survey papers.

Table 1. Related survey Papers on Malware Detection Approaches.

Coverage	Other Papers	Our Survey Paper
Survey of statistical based methods for malware detection	[13,14]	✓
Survey of machine learning based algorithms for malware detection	[15]	✓
Survey of deep learning based techniques to detect malware	[13,16,17]	✓
Analysis of problems associated with statistical based approaches of detecting malware	[18]	✓
Analysis of shortcomings of machine learning based solutions for detecting malware	[15]	✓
Analysis of disadvantages of using deep learning based methods to detect malware	[13,16]	✓
Survey of FSL methods in the domain of malware detection		✓

The contributions of this work are as follows:

- Description of malware classification and identification strategies
- Mechanisms for classifying and detecting malware and a comparative analysis between these methods
- Potential issues and challenges in the different categories of proposed solutions
- The future direction of research in this domain

This paper is organized in the following order (Shown in Figure 2): Section 2 describes the methods used in the case of the different trends in malware detection. Section 3 presents the comparative analysis of these trends. It also discusses the issues and challenges faced in each trend. Section 4 highlights future trends in the domain of malware identification and classification.

Figure 2. Organization of Paper.

2. Trends in Malware Detection

Information, in today's era, is one of the most valued but vulnerable assets. There is a constant threat of serious damage to infrastructure caused by evolving sophisticated malware. Various techniques, trends, and strategies are proposed to alleviate the threats triggered by malicious codes. These methods may range from the primitive type of malware detection based on statistical analysis to machine learning-based methodologies and specifically deep neural networks. As this paper is concerned with malware detection methodologies, so it is important to go through the evolution of malware identification and detection. In this section, a hierarchy is built to represent this development of malware detection according to the methodology used.

2.1. Malware Detection with Primitive Methods (Statistical Analysis Based Methods)

Malware detection is being performed with different techniques. Many researchers have explored the different practices for malware discovery and recognition. Ref. [19] focused on detecting a malicious pattern in executables. Majorly [19] has stated that malware detection is a kind of obfuscation-de obfuscation game in today's era, therefore authors in [19] have focused on the techniques of obfuscation to check whether present anti-virus products can overcome the variability introduced by obfuscation or not. They implemented SAFE (Static Analyzer for executables) which is claimed to detect a malicious pattern in executables. Further, they developed an obfuscator for executables that uses four different techniques to obfuscate the executable and then tested antivirus scanners by providing them with obfuscated variants of existing malicious executables. Ref. [19] presented a general architecture for detecting a malicious pattern in executables with two main components i.e., Program annotator and malicious code detector. Obfuscation transformations that are supported by the obfuscator detailed in [19] include register reassignment, dead-code insertion, code transposition, and instruction substitution.

Ref. [20] used a heuristic approach for detecting malware by analyzing windows binary files of obfuscated executables. They have come up with a framework that first generates a risk score by statically analyzing the windows PE (See Appendix B) file for 8 characteristics (abnormal ordinals, Nonstd_name, In_code, TLSection, DLL_no_export, Flagged Section Name, Low function Call, Other_badPEformat). This framework assigns weight and risk score to each characteristic. The risk score is assigned based on experience and comparison between malware and benign files. A total of 2014 windows files were used in experiments.

Ref. [21] primarily focused on malware detection through statistically making use of opcodes. In their methodology, first, the frequency of opcodes appearing in malware and benign files is calculated and then the statistics-based discrimination ratio is calculated through which weights are obtained for opcode sequences. Then the similarity between two executables is computed using weights of opcode sequences. Malware files are collected from the VxHeavens website, which was a total of 13,189 executables. For benign dataset 13,000 files are collected from their computer. The basic assembler is used to disassemble the executables. After obtaining the assembly file, a profile of opcodes' frequency is maintained. This file contains the unnormalized frequency of opcodes appearing in both datasets. Finally, the relevance of all opcodes is calculated giving mutual information between opcode and classification class. Finally, malware opcode sequences are extracted and their frequency of appearance is calculated to detect maliciousness. After calculating weighted term frequency, a vector of weighted opcode sequence frequency is obtained. Experimentally first opcode sequences of lengths 1 and 2 are extracted and the similarity in the sequences appearing in both malware and executables are calculated but, in both datasets, they are appearing almost with the same frequency due to which afterward opcode sequences of length 1 and 2 are combined to check the similarity of their appearance in both datasets. Malware variants have great similarity in terms of frequency of opcode sequences whereas similarity measure is low between malware and benign dataset.

One kind of malware is a botnet that scans the internet to find vulnerable hosts to perform various malicious activities. Normally botnets are coordinated through a Command-and-Control channel C&C and most of the control protocols are IRC based whereas other protocols such as HTTP can also be used. Ref. [22] focused on detecting and confining DDoS and portscan. Authors in [22] brought up a platform that focused on detecting malicious activities by monitoring communication between botnet and C&C and by monitoring traffic for detecting and confining DDoS along with the detection of zombie computers on the network. Resultantly they managed to filter botnet-related traffic, confined infected parts of the network, and found methods for disabling botnets. To collect malware, high and low interaction honeypots were used. Low interaction honeypots used in the experiment were (1) Nepenthes and (2) Honeyd. After the malware was captured, it was analyzed manually. They were identified using various anti-virus tools and were sandboxed to collect useful information. Then a victim PC was connected to the analysis workstation and traffic generated by the victim PC in a clean state was monitored. Wireshark was started on an analysis workstation. Afterward, the victim's PC was rebooted with malware installed on it, and then events related to DNS requests attempted to connect to unknown ports and scanning of unknown ports was recorded. Dnsmsaq, fakemta relay-Http, relay, and Wireshark were used as tools for different purposes. This methodology was cumbersome to perform intended functionalities, therefore, MWNA (Malware Network Analyzer) was developed. It is based on the Linux Packet Filter mechanism. The published method for detecting DDoS analyzes packets during normal traffic: first to establish a baseline and then to derive thresholds. Then finally some attack features are extracted. Finally, above mentioned method is combined with a rate-limiting scheme so that amount of monitored traffic can be reduced.

A hybrid approach is also being used for taking benefit from the amalgam of malware detection methods. Ref. [23] focused on availing the advantages of all techniques for malware detection due to which the implemented framework by [23] is hybrid. They presented a framework that works on the detection methodology involving API calls extracted from the suspected file by running it in a VM environment. Then a graph is built using the information of API calls and operating system resources being utilized. Graph nodes represent API calls and operating system resources, and edges represent the reference between nodes. Then the constructed graph is minimized. Finally, to find a match between two graphs, the Graph Edit Distance algorithm is used, and to make use of this algorithm cost matrix is utilized.

Ref. [24] developed a tool, PyTrigger, which provides the user actions required to trigger, collect, and distill malware behavior profiles. Their paper has made three major contributions including the development of an algorithm that helps in extracting malware behavior, user-triggered malware behavior from among a similar event along with an event recording and playback system, and the full implementation of the PyTrigger system. PyTrigger has two major subsystems: (1) the recording and playback system and (2) the behavior analysis system. The recording and playback subsystem of PyTrigger is supposed to record the values of all objects' data states such as windows' titles, mutable text field values, drop-down menu choices, etc. and are then forcibly entered in GUI while being replayed to create the scenario which triggers the malware behavior. PyTrigger system executes the malware sample several times in VM and uses Events Tracing for Windows to trace the events. PyTrigger system was evaluated on 4100 malware samples from 35 different malware families. Typical user activity that was recorded was related to Gmail, Facebook, and Google HSBC, text editing, file browsing, and execution (Windows Explorer). An added advantage of this system is its ability to extract delegated events. Events that are delegated by the malicious process to other processes which are legitimate and lie outside the malware process chain are called delegated events.

Ref. [25] concentrated on the solution for detecting malicious activity which should be low cost and should not be using any third-party software so that in less time and low budget detection can be done. Secondly, since some malware behavior can overcome the

virtual environment, therefore, running malware in a virtual machine for dynamic analysis can compromise some of the triggering scenarios. The authors manipulated windows audit logs into interpretable features and presented a linear classification model for detecting malicious behavior using the windows audit log as a feature set with high accuracy. This approach explored some new malware behaviors. For performing validation, six different experiment sets were designed. One of the experiments for validation involved a dataset that had malware a year or two older than the malware presented in training. Second experiment for validation was performed based on malware families. Secondly, the same trained classifier was run in a virtual environment as well as in an enterprise environment to cater to the variable of the environment. The experimental dataset consisted of 32,078 samples out of which 17,399 were benign samples and 14,679 malicious samples. 6,898,593 unique features were extracted, and 20,362 audit logs were collected from binaries executed in a cuckoo sandbox.

Figure 3 shows the performance metrics used by the surveyed papers that fall in the category of statistical based methods.

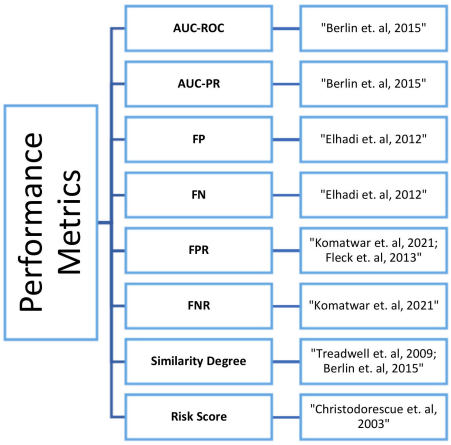

Figure 3. Performance Metrics Used in Literature Proposing Primitive Methods for Malware Detection.

2.2. Malware Detection with Conventional Machine Learning Based Methods

Machine learning plays an important role to capture helpful properties in malware to advance security measures. This whole process of knowledge extraction and learning of patterns helped the researchers to pave their steps into machine learning-based malware analysis and detection. Machine learning has been extensively used not only in malware detection but also for detecting malicious activity through network traffic [26].

Ref. [27] worked on Belief propagation with the file system but could not do well for new samples. Ref. [28] conducted malicious graph matching and extracted APIs/System calls but they used a small dataset. Ref. [29] used a Rule-based classifier and SVM and performed detection based on byte sequences but made use of only specific malware classes for evaluating their model. They built datasets from Windows system files and the Anti-Virus Platform. Ref. [30] also used a Rule-Based Classifier and extracted APIs/System calls but this APIs/System calls categorization was not up to the mark. They conducted their tests on features of the Windows XP system and Program Files folders. Authors of [31,32] used Random Forest and used network and API system calls, Registry, and File system but the dataset was small. Ref. [33] used Decision Trees in their research work and [34] used Naïve Bayes, Random Forest, and SVM and worked on byte sequences, APIs/system calls, file systems, and Windows registry. Ref. [35] used KNN for detecting malicious PEs. Malware code causes damage to the resources, and with a little code change, malware developers can easily beat the protection layer. A lot of research was done for the detection of these variants. Ref. [36] explored the Decision Tree and Random Forest and made use of Opcodes. They used small datasets of Windows XP system and Program Files folders and generated code of malware for making part of the dataset. Ref. [37] performed

Clustering with locality-sensitive hashing Byte sequences but the used dataset was very small. Ref. [38] worked on a Rule-based classifier, they worked on APIs/System calls, and Windows Registry. Ref. [39] used the clustering technique which was being used for variants detection by past researchers also. The authors chose DBSCAN but their approach was not coping with malware evasion techniques. Ref. [40] worked on Logistic Regression and Neural Networks and operated on Byte sequences and APIs/system calls.

Table 2 shows the datasets and performance metrics used by the researchers in the surveyed papers that apply conventional machine learning algorithms.

Table 2. Datasets and Performance Metrics Used in Literature Proposing Machine Learning Methods for Malware Detection.

Title	Author	Data Samples Used			Performance Metrics Used
		Source	Malicious	Benign	
Support Vector Machine for malware analysis and classification	M. Kruczkowski, E. N. Szynkiewicz	N6 Platform	-	-	Classification Accuracy = 0.9498 Sensitivity = 0.9774 Specificity = 0.8971 AUC = 0.9901 F1 = 0.9623 Precision = 0.9475
Improving the detection of malware behavior using simplified data dependent API call graph	E. Elhadi, M. A. Maarof, B. Barry	VxHeavens	75	10	Detection Rate = 98.6% Accuracy = 98.8% False Alarm = 0%
Dynamic VSA: a framework for malware detection based on register contents	M. Ghiasi, A. Sami, Z. Salehi	Windows XP system, Program Files Folder, and Private Repository	850	390	TP = 0.988 FP = 0.125 Recall = 0.988 Precision = 0.888 F-Measure = 0.940 Accuracy = 0.930
Novel feature extraction, selection, and fusion for effective malware family classification	M. Ahmadi, G. Giacinto, D. Ulyanov, S. Semenov, M. Trofimov	Microsoft's Malware classification challenge	21,741	0	Accuracy, Logloss
Probabilistic inference on integrity for access behavior based malware detection	W. Mao, Z. Cai, D. Towsley, X. Guan	Windows XP SP3 VxHeavens	7257	534	TPR, AUC
Robust and effective malware detection through quantitative data flow graph metrics	T. W¨uchner, M. Ochoa, A. Pretschner	Legitimate app downloads Malicia	6994	513	Detection Rate, FPR, Precision, F-Measure
An alternative to NCD for large sequences, Lempel Ziv Jaccard distance	E. Raff, C. Nicholas	Industry Partner	237,349	240,000	Balanced Accuracy
Proposing a HMM-based approach to detect metamorphic malware	M. Gharacheh, V. Derhami, S. Hashemi, S. M. H. Fard	Cygwin VxHeavens	-	-	Detection Rate = 0.9803 FPR = 0.0058 Accuracy = 0.9833
Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms	P. Khodamoradi, M. Fazlali, F. Mardukhi, M. Nosrati	Windows XP system and Program Files folder Self-generated metamorphic malware	280	550	Accuracy
A malware similarity testing framework	J. Upchurch, X. Zhou	Sampled from security incidents	85	0	PR Curve
A behavior based malware variant classification technique	G. Liang, J. Pang, C. Dai	Anubis Website	330,248	0	Similarity measure

Table 2. *Cont.*

Title	Author	Data Samples Used			Performance Metrics Used
		Source	Malicious	Benign	
Scaling Malware Execution with Sequential Multi Hypothesis Testing	P. Vadrevu, R. Perdisci	Security Company and Large Research Institute	1,651,906	0	Jaccard Index
Fast malware classification by automated behavioral graph matching	Y. Park, D. Reeves, V. Mulukutla, B. Sundaravel	Legitimate apps Anubis Sandbox	300	80	Similarity measurement
Automated malware classification based on network behavior	S. Nari, A. A. Ghorban	Communi-cation Research Centre Canada	3768	0	Accuracy = 94.5783%
Malware function classification using APIs in initial behavior	N. Kawaguchi, K. Omote	FFRI Inc.	408	236	Accuracy, FPR, FNR
Feature selection and extraction for malware classification	C.-T. Lin, N.-J. Wang, H. Xiao, C. Eckert	Sandbox	3899	389	Micro Precision, Micro Recall, Micro Specificity, Macro Precision, Macro Recall, Macro F1
High fidelity, behavior based automated malware analysis and classification	A. Mohaisen, O. Alrawi, M. Mohaisen	AMAL system	115,157	0	
Clustering for malware classification	S. Pai, F. Di Troia, C. A. Visaggio, T. H. Austin, M. Stamp	Cygwin utility files and Malicia	8052	213	Silhouette coefficient, purity
Towards Automatic Reverse Engineering of Large Datasets of Binaries	M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw	-	-	-	Jaccard Index
Subroutine based detection of APT malware	J. Sexton, C. Storlie, B. Anderson	-	197	4622	Similarity index
A static signal processing based malware triage	D. Kirat, L. Nataraj, G. Vigna, B. Manjunat	Windows XP, ZDNet, NSRL, Anubis	1,200,000	52,750	Precision and Recall

2.3. Malware Detection with Deep Learning Based Methods

Deep Learning is a specialized form of machine learning in the domain of Artificial Intelligence (AI) that applies deep artificial neural networks also famous as deep neural networks. They are the techniques of machine learning that simulate the process of learning by a human brain. The human brain consists of cells which are referred to as neurons in neural networks. Similarly, in a human brain, all the cells are connected through axons and dendrites with the connection region known as synapses. These connections when found in ANN (Artificial Neural Networks), contain weights to behave as the connections between nerve cells in the human brain. Figure A2 (Appendix B) shows the human brain and simulated version of the human brain through the artificial neural network.

The major difference between conventional neural networks and deep neural networks is the number of layers. Deep neural networks make use of many hidden layers for the high-level abstraction of data. They can learn the features of data. This process of feature engineering is carried out with the help of a big number of examples input to the deep learning-based algorithm which leads to the production of results in the form of classification, identification, or generation of data after learning the most suitable features during feature engineering. The major motivation for using deep learning in various fields was to organize and analyze a large amount of data. Different areas where deep networks are preferred to be used include image processing, speech processing, healthcare, and with the increase in cyber space, now even cybersecurity.

Depending upon its features, this domain can be further categorized into different sub-domains as shown in Figure 4. All features of PE files hold some significance in defining degree of maliciousness in a particular file. Features from the header and Imports, all play

a significant role in defining the nature of PE file as malicious or benign. Ref. [41] made use of LSTM for the selection of optimal features of PEs. These optimal features were selected to train a deep learning based model for detecting malicious PE file.

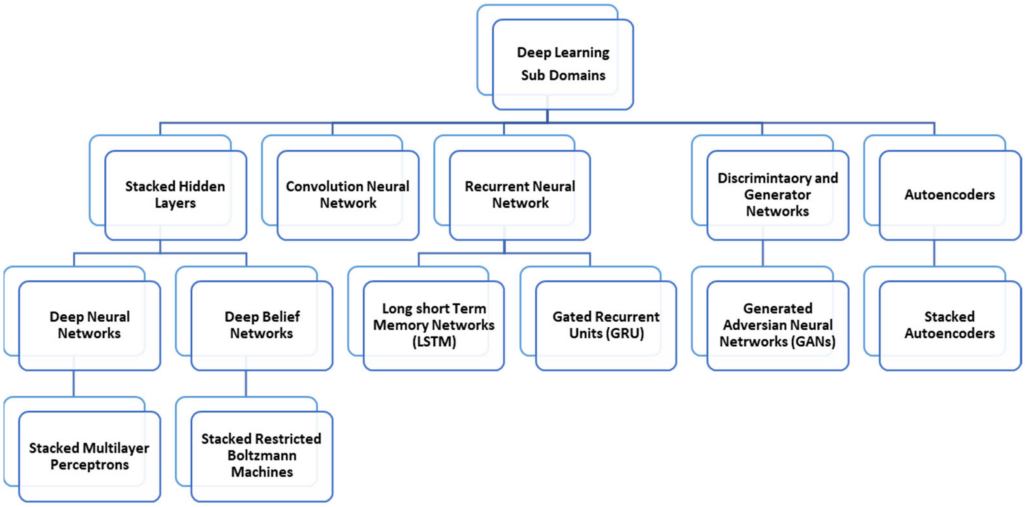

Figure 4. Types of Deep Learning.

Refs. [42,43] made use of sequential dynamic data and claimed that an ensemble of recurrent neural networks can be capable to detect the maliciousness of an executable within the first 4 s of execution with almost 93% accuracy. GRU (Gated Recurrent Units) were used with RNN to reduce training time. User CPU usage, and system CPU usage, sent packets to count, received bytes count, total bytes sent, count of the processes being executed, the maximum number of processes being carried out, the number of milliseconds elapsed since the file started to run and maximum process ID assigned were used as features.

Ref. [44] combined two types of neural network layers i.e., convolutional, and recurrent layers for modeling system call sequences for classifying malware. These two types of layers use dissimilar types of approaches for modeling sequential data. Convolutional networks use sequences in the form of a set of n-grams, and recurrent networks tend to train a stateful model by using full sequential information. The input of the system was 60 distinct system calls.

Ref. [45] performed malware detection using stacked AutoEncoders (SAE) with the input of Windows API calls mined from the PE files. The SAEs model worked on a greedy layer-wise training operation for performing unsupervised feature learning. Then this process was followed by supervised parameter fine-tuning. Results showed that the model with 3 hidden layers and 100 neurons at each layer gave the best training and testing accuracy as compared with ANN, SVM, Naïve Bayes, and Decision Tree.

Ref. [46] implemented a method that manipulates raw inputs to detect maliciousness. The implemented model called eXpose picks generic short strings from security inputs. These strings include malicious URLs, mutexes, registry keys, etc. Then it learns to identify their maliciousness. eXpose makes use of a neural network convolutional kernel for feature extraction. The architecture is composed of notional components along with character embedding, feature detection components, and classifier. Results showed that eXpose outperformed manual feature extraction approaches, attaining a 5–10% detection rate gain at a 0.1% false-positive rate compared to these baselines.

The proposed model by Ref. [47] is comprised of phases of OpCode-Sequence Graph Generation, Deep Eigensapce Learning, and Feature Selection for the detection of Internet of Battlefield Things (IoBT) malware. Ref. [47] used a Convolutional Network for the deep learning module, because it can give more accurate results of classification when the data patterns are complex and nonlinear. This approach achieved 99 % accuracy and 98% Recall.

Ref. [48] focused on addressing the detection task of malware variants with the help of deep learning methods. The authors got a method published in which they transformed the nasty code into a grayscale image. Then the images were recognized and classified by employing a Convolutional Neural Network (CNN) which could extract the features of the malware images automatically. The implemented CNN was composed of an input layer, convolutional, and subsampling layers. This model also classified malware into related malware families.

Ref. [49] used the approach of converting the disassembled malware code into a greyscale image using SimHash and then used a Convolutional Neural Network to identify the malware family. The presented methodology is comprised of three phases: Feature extraction, Malware image generation, and CNN training. Results showed that the authors were successful to obtain an accuracy of approximately 99% with 10,805 samples.

Ref. [50] have focused on the description of state-targeted APT using a Deep Neural Network (DNN). Researchers utilized the ability of Deep Neural Networks (DNN) to make use of raw features as input, whereas the learning of higher-level features was done during the training process. In this progression, every hidden layer extracted higher-level features from the preceding layer, building a hierarchy of higher-level features.

Ref. [51] devised an approach of using a neural network comprised of convolutional and feed-forward neural constructs for malware classification. In this approach PE file metadata, import features and Assembly opcode features categories were used.

Ref. [52] made use of a dynamic analysis approach based on Windows API call graphs and SAE models. A Behavior-based Deep Learning Framework (BDLF) was developed in this paper which makes use of SAE for feature reduction from behavior graphs and then performs classification through Decision Tree, KNN, Naïve Bayes, and SVM.

Ref. [53] focused on malware detection based on process behavior in possible infected terminals. The published solution applies DNN in 2 stages, the first stage is for extracting process activities by RNN and converting them into feature vectors. Feature vectors were then treated as images that were classified by CNN.

Ref. [54] have worked on a new image processing technique with optimized parameters for Machine Learning algorithms and Deep Learning architectures to produce an efficient zero-day detection system of malware. First malware detection was performed using deep learning based on static analysis on ember dataset and privately collected samples and it was deduced that the performance of malware detection can marginally be enhanced by using a hybrid system pipeline proposed as Windows-Static-Brain-Droid (WSBD), which was composed of both classical machine learning algorithms and deep learning models. In the next stage of research, malware detection was performed using deep learning based on dynamic analysis. It conducted a comparison between classical machine learning algorithms and deep learning architectures based on dynamic analysis, and deep learning architectures outperformed all experiments. Finally, experiments were conducted for categorizing the malware into malware families using deep learning based on image processing. A novel technique DeepImageMAlDetect (DIMD) was proposed which is based on the image processing technique and uses CNN and LSTM. The proposed method can work on malware from different operating systems. Finally, architecture by the name of ScaleMalNet was developed. It collects data from different data sources and uses self-learning techniques such as classical machine learning algorithms, deep learning architectures, and image processing techniques for detecting, classifying, and categorizing malware to their corresponding malware family efficiently.

Authors in [55] proposed a new technique to generate a signature for malware that does not depend on any specific behavior of malware so that it can be used for variants of malware as well. To achieve the goal, researchers first recorded the behavior of malware through Sandbox and then converted the output text file into a binary vector sized. After creating a binary vector Deep Belief Network was trained by a Deep Stack of Denoising Autoencoders.

Ref. [56] focused on a technique that made use of a Deep Neural Network for malware detection using features extracted statically with more accuracy and minimum FPR. There

are three main components of the framework defined in this paper: (1) the First component focuses on the extraction of four features from benign and malicious binaries (2) 2nd component is a Deep Neural Network consisting of an input layer, two hidden layers, and one output layer (3) 3rd component is the score calibrator.

Research of [57] focused on one-shot learning which is referred to when there are very few samples to learn from. It implements a model LRUA-MANN which modifies the memory access capability of a Neural Turing Machine to adapt a one-shot learning task. LRUA-MNN is used with LSTM as a controller and makes use of LSTM state and memory bank as memory.

Ref. [58] has focused on carrying out the process of malware detection without having in-depth knowledge of malware and its analysis. Two Neural Networks were used; one was fully connected, and the other was a Recurrent Neural Network. The model had 3 LSTM layers with attention mechanisms before classification. Sax et al. used Neural nets and extracted Strings and PE file characteristics but did not cope with obfuscation and did not produce good accuracy in such situations.

Ref. [59] implemented the idea of a multitasking learning model which was trained for seven classification tasks for malware image classification. The implemented model by [59] consisted of 5 CNN layers with PRelu activation function.

Ref. [60] have explored the advantages of using transfer learning in the domain of malware identification. Their research focused on utilizing transfer learning for extracting the features of malware dataset. They made use of an already trained deep learning model (trained over ImageNet) and finally classified the malware into their respective families.

Figure 5 summarizes the types of deep learning algorithms used by researchers over the years and Table 3 summarizes the performance metrics used by researchers while using deep learning based methods for malware detection.

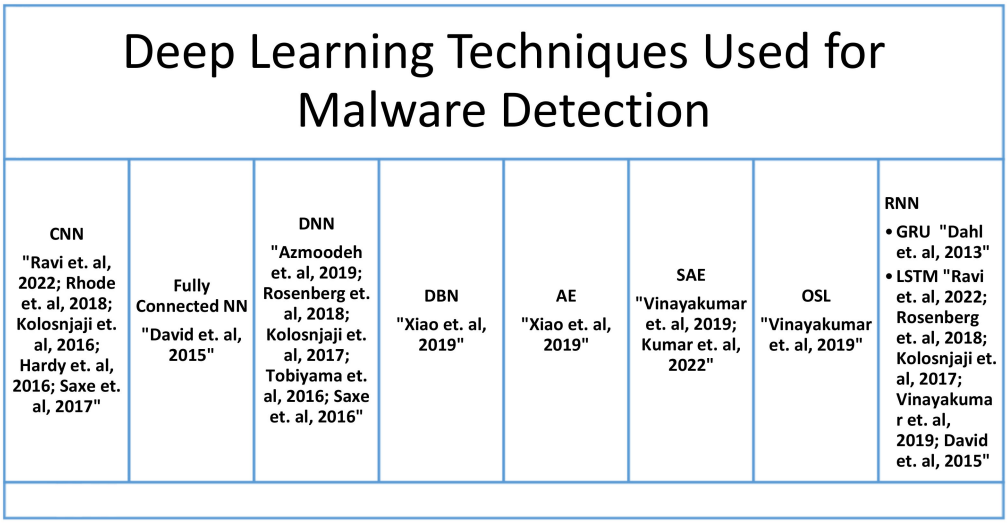

Figure 5. Deep Learning Techniques Used for Malware Detection.

Critical analysis of all the surveyed papers that implemented deep learning algorithms, emphasizes the grave need of using a large dataset to produce reliable results. Deep learning architectures heavily make use of supervised learning that requires a large no. of labeled examples for training the model as mentioned by [61]. Using the small dataset does not help the model to learn the features properly during the training phase which leads to non-reliable results. Another aspect that got unveiled during this survey referred to the fact that this large dataset is supposed to contain a large no. of examples for each class that must be identified by the trained model. And processing the bulk of data in deep learning needs powerful hardware, high computational processing power, and high training time which diminishes the chance of applying the trained models to real-time data. Because of these unavoidable features of deep learning models, the market could not get successful in

replacing the signature-based anti-malware systems with artificially intelligent systems. Therefore, researchers shifted their direction of research from developing deep models for feature learning to finding out the possibilities of developing models that can work over small datasets. In the quest of achieving the previously mentioned objective, researchers explored the concept of Few Shot Learning (FSL) which is based on meta learning with a focus on learning the strategy of how to learn the meaningful properties of data. Meta learning utilizes the concept of transfer learning (multi-task learning) and semi-supervised or unsupervised learning approaches which need a few examples for the training. And thus, according to [62], the meta learning model can be trained with the help of prior knowledge. Meta learning based algorithms that are being used in malware analysis include Few Shot Learning (FSL), One shot Learning (OSL), and Zero Shot Learning (ZSL). Figure 6 shows the relationship between machine learning and meta learning models. Major advantages of meta learning based algorithms are listed in Figure 7.

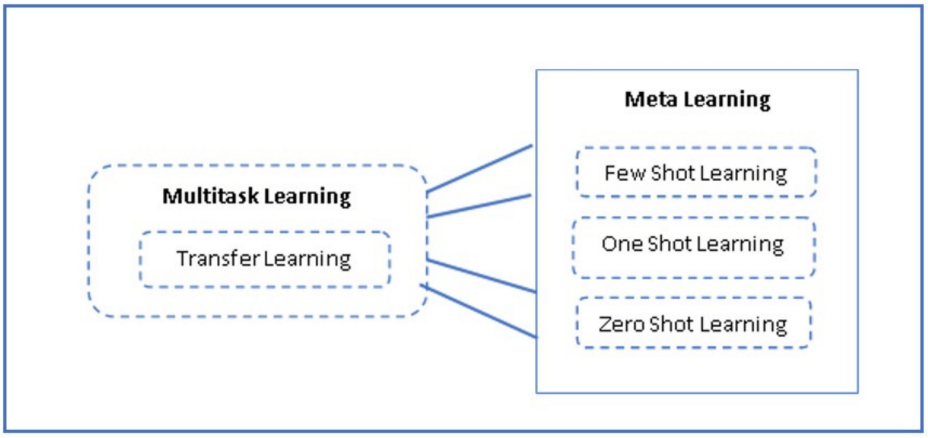

Figure 6. Relationship Between Machine Learning and Meta Learning.

Table 3. Datasets and Performance Metrics Used in Literature Proposing Deep Learning Methods for Malware Detection.

Title	Author	Year	Dataset Samples				Performance Metrics
			Source	Malicious		Benign	
Early Stage Malware Prediction Using Recurrent Neural Networks	Rhode, Matilda, et al.	2018	Machine Activity collected in VM using Cuckoo Sandbox	594		594	Accuracy = 93% (After 4 min of malware execution)
DL4MD: A Deep Learning Framework for Intelligent Malware Detection	Hardy, William, et al.	2016	Comodo Cloud Security Centre	22,500		22,500	TP = 22,035 FP = 953 TN = 21,547 FN = 465 Accuracy = 96.85%
eXpose: A Character Level Convolutional Neural Network with Embeddings for Detecting Malicious URLs, File Paths and Registry Key	Saxe, Joshua, and Konstant-in Berlin.	2017	VirusTotal	URLs	7,211,705	1,496,198	TPR = 0.77×10^{-4} FPR = 0.84×10^{-3} AUC = 0.993
				File Paths	869,836	3,677,404	TPR = 0.16×10^{-4} FPR = 0.43×10^{-3} AUC = 0.978
				Regist-ry Keys	250,819	1,282,292	TPR = 0.51×10^{-4} FPR = 0.62×10^{-3} AUC = 0.992

Table 3. *Cont.*

Title	Author	Year	Dataset Samples				Performance Metrics
			Source		Malicious	Benign	
Robust Malware Detection for the Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning	Azmood-eh, Amin, Ali Dehghanta-nha, and Kim Kwang Raymond Choo.	2018	VirusTotal		1078	128	Accuracy = 99% Recall = 98%
Detection of Malicious Code Variants Based on Deep Learning	Cui, Zhihua, et al.	2018	Vision Research Lab		9342 (25 Malware Families)	-	Accuracy = 94.5 Precision = 94.6 Recall = 94.5 Runtime = 20 ms
Malware Identification Using visualization images and deep learning	Ni, Sang, Quan Qian, and Rui Zhang	2018	Kaggle 2015		10,085 (9 Malware Families)	-	Accuracy = 99%
End-to-End Deep Neural Networks and Transfer Learning for Automatic Analysis of Nation State Malware	Rosenberg, Ishai, Guillaume Sicard, and Eli David.	2018	Cuckoo Sandbox		3200 (2 APT classes)	-	Accuracy = 98.6%
Empowering Convolutional Networks for Malware Classification and Analysis	Kolosnjaji, Bojan, et al.	2017	Virusshar, Maltrieve, Private Collection		-	-	Precision = 0.93 Recall = 0.93 F-1 Score = 0.92
Malware Detection Based on Deep Learning of Behavior Graphs	Fei Xiao et al.	2019	Vx Heaven		880	880	Precision = 0.986 Recall = 0.992 F-1 Score = 0.989
Deep Learning for Classification of Malware System Call Sequences	Bojan et al.	2016	Virusshar, Maltrieve, Private Collection		4753	-	Precision = 85.6% Recall = 89.4%
Malware Detection with Deep Neural Network Using Process Behavior	Shun Tobiyama et al.	2016	NTT Secure Platform Laboratory		81	69	AUC = 0.96
Robust Intelligent Malware Detection Using Deep Learning	R. Vinaya Kumar et al.	2018	WSBD	Ember	70,140	69,860	Accuracy = 98.9% Precision = 99.7% Recall = 98.1% F-1 score = 98.9%
			WDBD	Cukoo Sandbox	173,946	169,509	Accuracy = 93.6% Precision = 94.8% Recall = 92.0% F-1 Score = 93.4%
			DIMD	Malimg, Virus-sign, Virus-share	24,851	-	Accuracy = 96.3%
Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features	Joshua et al.	2015			81,910	350,016	TPR = 95.2% AUC = 0.999

Table 3. *Cont.*

Title	Author	Year	Dataset Samples				Performance Metrics
			Source		Malicious	Benign	
Learning the PE Header, Malware Detection With Minimal Domain Knowledge	Edward Raff, Jared Sylvester, Charles Nicholas	2017	Group A	Virus- share	301,575	291,285	Accuracy = 90.8% AUC = 97.7%
			Group B	Industry Partner	240,000	237,349	Accuracy = 83.7% AUC = 91.4%
One Shot Learning Approach for Unknown Malware Classification	True Kien, Hiroshi Sato, Masao Kubo	2018	Malicia Project, Virustotal			23,080	Accuracy (with training) = 0.74 Accuracy (without training) = 0.85
DTMIC: Deep transfer learning for malware image classification	Sanjeev Kumar, B. Janet	2022	MalImg and MS BIG dataset		9339 + 10,868		Accuracy on MalImg = 98.92% Accuracy on BIG dataset = 93.19
Deep multitask learning for malware image classification	Ahmed Bensaoud, Jugal Kalita	2022	Virusshare, Virus total, contagio				Accuracy = 99.97% TPR = 99.98 FPR = 0.73
DTMIC: Deep transfer learning for malware image classification	Sanjeev Kumar, B. Janet	2022	MalImg and Microsoft		9339 + 21,741		Accuracy = 98.92 Precision = 99 Recall = 99

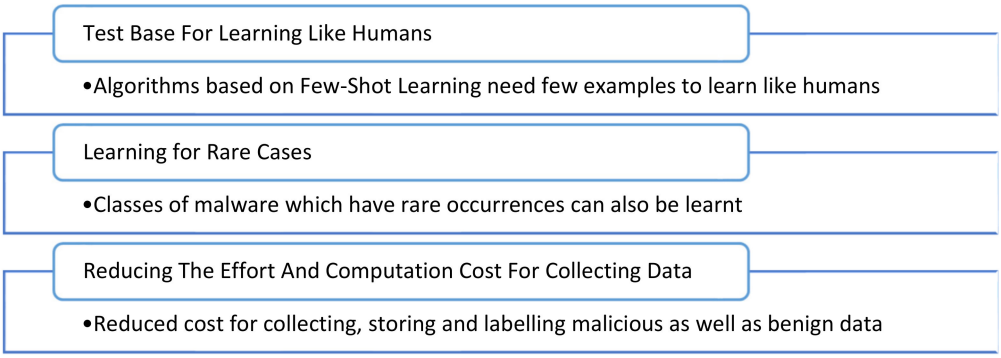

Figure 7. Advantages of Meta Learning.

Ref. [63] have explored the Siamese network for malware image classification. Siamese network architecture is the application of one shot learning field. The basic approach used by [63] was to transform the features into malware images that were input to Siamese Convolutional Neural Networks shown in Figure 8. Siamese CNNs used by the [63] produce 2 feature vectors. Finally, the Manhattan distance between those feature vectors was calculated and given to the sigmoid function to generate the similarity score.

Another surveyed paper [57] mentioned the use of one shot learning approach with a memory augmented neural network using the API calls sequence. Ref. [57] adapted an approach that has two domains of learning. The first domain in this approach is used to train the model with known malware and 2nd domain is used to train or test with a dataset of an unknown type of malware. Domain 2 makes use of domain 1′s trained model. The working of the implemented approach [57] is shown in Figure 9.

Ref. [64] have explored one shot learning approach with matching and prototypical networks. The developed model by [64] is shown in Figure 10. Ref. [64] take advantage of visual dissimilarity in the images of different malware families (shown in Figure 11) and have converted the malware binaries into 8-bit greyscale images to be given as input to the few shot learning models.

Ref. [65] presents a few shot learning based neural network ConvProtoNet. ConvProtoNet in [65] used stacked convolutional layers rather than only computing means, to

generate features of malware classes. ConvProtoNet is capable of being trained on one dataset and tested on another.

Ref. [66] composed the dataset of splash screen images showing the message of the system being attacked by the ransomware. They trained their one shot learning model on a dataset of 50 ransomware families splash screen images. Different augmentation techniques are used by [66] to tune the images for adapting one shot learning.

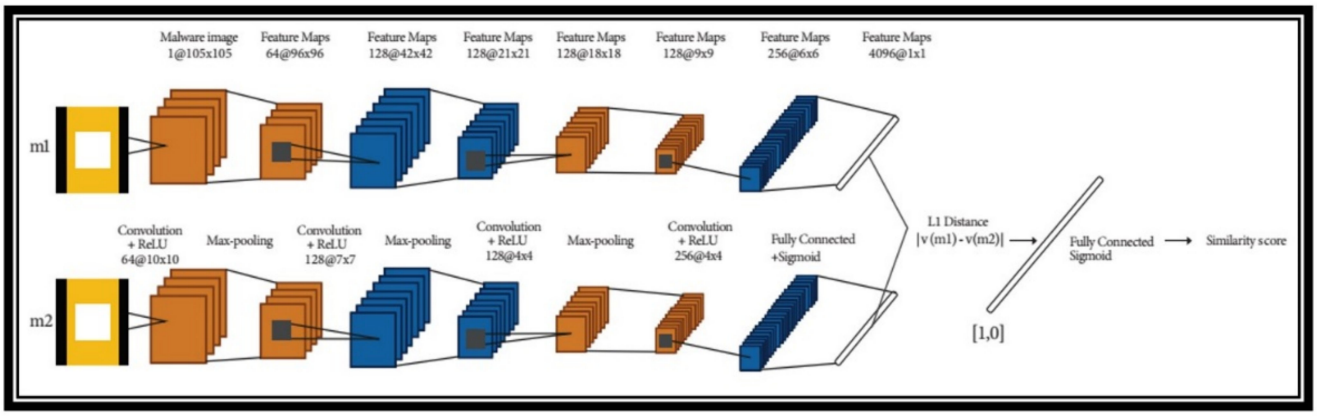

Figure 8. Siamese CNN used by [50].

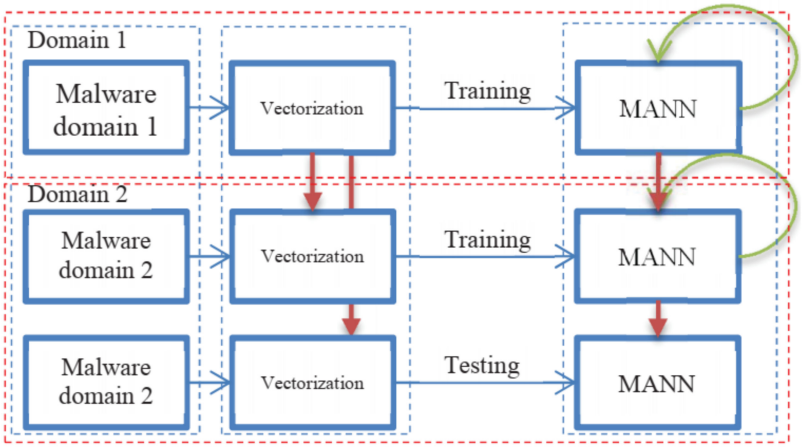

Figure 9. Proposed Approach of [46]. 2 Phases of Training and Testing, Domain 1 and Domain 2.

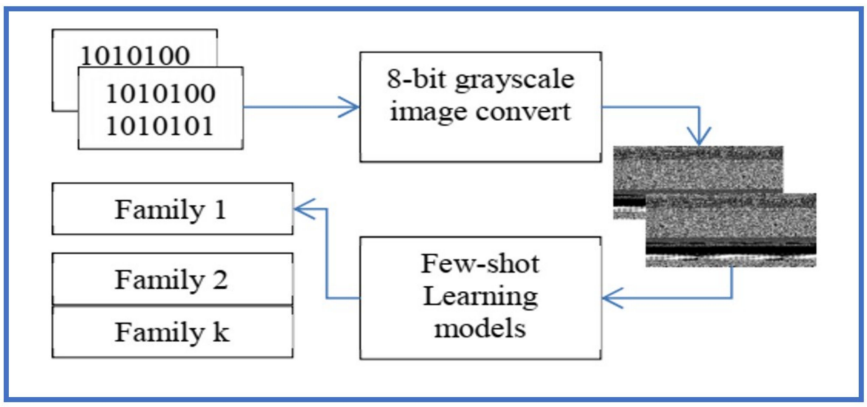

Figure 10. Proposed Approach in [51].

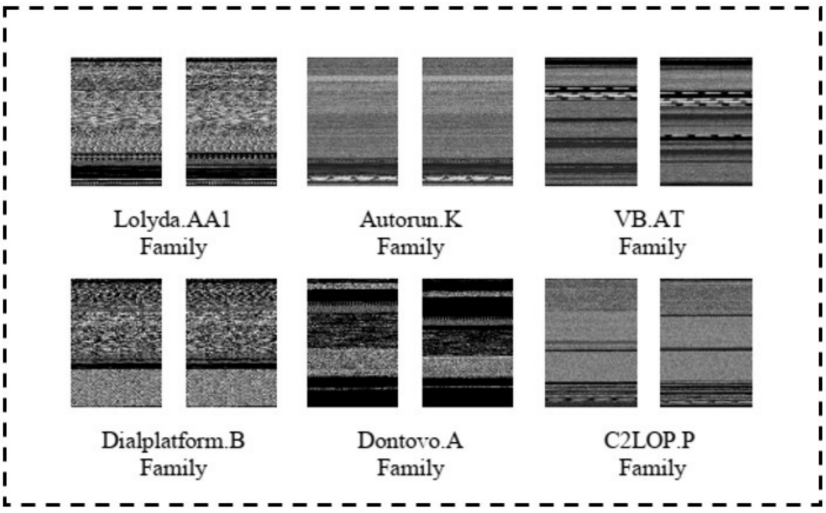

Figure 11. Visual Samples showing Dissimilarity Between the Images of Different Families [51].

3. Issues and Challenges

Every trend in malware detection and analysis has come forward with some of its shortcomings due to which trend of research got shifted to other technologies for detecting malware in real-time with minimum false positive rate and maximum accuracy. This section will highlight all the challenges faced by each trend and the disadvantages of different techniques adapted for malware detection and analysis. Tables 4–6 summarize all issues of surveyed papers based on different analysis methods.

Table 4. Limitations of Surveyed Papers Proposing Primitive Methods for Malware Detection.

Title	Author	Year	Weakness/Limitation
The architecture of a Platform for Malware Analysis and Confinement	Gilles Berger et al.	2010	Usage of low interactive honeypots. Malware that gets active under certain conditions might not be detected in such a scenario.
A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection	Anusha Damodaran et al.	2015	In this research work, the comparison is performed only on opcode and system calls whereas there are many more static and dynamic useful features.
PyTrigger: A System to Trigger & Extract User-Activated Malware Behavior	Dan Fleck et al.	2013	Selected features were used in this research work which could have compromised the information which could be gained by other features.
Malicious Behavior Detection Using Windows Audit Logs	Konstantin et al.	2015	Researchers in this search have not mentioned if they have dealt with obfuscated logs. Moreover, each sample is run for four minutes, and this window audit log is not sufficient for slow-moving malware.
Static Analysis of executables to Detect Malicious Patterns	Mihai et al.	2006	The detection algorithm is context insensitive and cannot track the calling context of the executable. It can be made context-sensitive.
Idea: Opcode-Sequence-Based Malware Detection	Igor Santos, Felix Brezo et al.	2010	One of the limitations of this paper was that the authors did not deal with packed executables which are a major part of real time data. Secondly, they used quite a small dataset.
Malware Detection Based on Hybrid Signature Behavior Application Programming Interface Call Graph	Ammar Ahmed et al.	2012	Evasion techniques were not catered to.
A Heuristic Approach for the Detection of Obfuscated Malware	Scott Treadwell, Mian Zhou	2009	Some legitimate applications are reported as malware so the False Positive Rate is high.
Detecting Metamorphic Malware Using Code Graphs	Jusuk Lee, Kyoochang Jeong, Heejo Lee	2010	Only 3 obfuscation techniques were mitigated whereas there are 6 to 8 more obfuscation techniques that are normally applied by the malware writers.

Table 5. Limitations of Surveyed Papers Proposing Machine Learning Based Solutions.

Title	Author	Year	Weakness/Limitation
Improving the detection of malware behavior using simplified data dependent API call graph	E. Elhadi, M. A. Maarof, B. Barry	2015	One of the major limitations of this research work was the small dataset.
Dynamic VSA: a framework for malware detection based on register contents	M. Ghiasi, A. Sami, Z. Salehi	2015	The authors of this paper used a small dataset which means there is a great chance that models were not trained well. Secondly, a subset of features was used which means there might be many more useful features that were ignored while training.
Novel feature extraction, selection, and fusion for effective malware family classification	M. Ahmadi, G. Giacinto, D. Ulyanov, S. Semenov, M. Trofimov	2015	Feature optimization ignored real distribution
Probabilistic inference on integrity for access behavior based malware detection	W. Mao, Z. Cai, D. Towsley, X. Guan	2015	This research work was carried out on a small dataset, and evasion techniques were not even taken care of, due to which reliability is compromised. Some of the malware need human interaction to get activated as they get triggered over certain input. This important fact was even ignored while the feature extraction process.
Robust and effective malware detection through quantitative data flow graph metrics	T. W¨uchner, M. Ochoa, A. Pretschner	2015	One of the major limitations of this research work was the usage of a small dataset, secondly, since obfuscation was not dealt with while training so in case encountering the obfuscated malware model would not perform well.
An alternative to NCD for large sequences, Lempel-Ziv Jaccard distance	E. Raff, C. Nicholas	2017	Researchers did not consider the obfuscation while training the models, therefore the performance of models would not be good on real time data.
Proposing a hmm-based approach to detect metamorphic malware	M. Gharacheh, V. Derhami, S. Hashemi, S. M. H. Fard	2015	The authors of this paper used a small dataset, secondly, a subset of features was used which means there might be many more useful features that were ignored while training.
Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms	P. Khodamoradi, M. Fazlali, F. Mardukhi, M. Nosrati	2015	Once again, this research work has used small, and again a subset of features is used in this work which means the research has ignored many useful features also.
A malware similarity testing framework	J. Upchurch, X. Zhou	2015	An extremely small dataset which raises a serious question on the reliability of the model's training.
A behavior-based malware variant classification technique	G. Liang, J. Pang, C. Dai	2016	Small dataset, non-optimized feature set, ignored real distribution
Scaling Malware Execution with Sequential Multi Hypothesis Testing	P. Vadrevu, R. Perdisci	2016	Since this research work ignored evasion techniques and user interaction for triggering malware behavior during feature extraction, therefore model might not perform well on real time data.
Automated malware classification based on network behavior	S. Nari, A. A. Ghorban	2013	Using small dataset in training machine learning algorithms, compromises the reliability of results.
Malware function classification using APIs in initial behavior	N. Kawaguchi, K. Omote	2015	The small dataset which has been used in this research work is the main limitation of this paper.
Feature selection and extraction for malware classification	C.-T. Lin, N.-J. Wang, H. Xiao, C. Eckert	2015	Since this research work ignored evasion techniques and user interaction for triggering malware behavior during feature extraction, therefore model might not perform well on real time data. Moreover, a small dataset was used for training which establishes the fact that the models were not well trained.
High-fidelity, behavior based automated malware analysis and classification	A. Mohaisen, O. Alrawi, M. Mohaisen	2015	This research work ignored evasion techniques.
Clustering for malware classification	S. Pai, F. Di Troia, C. A. Visaggio, T. H. Austin, M. Stamp	2015	Researchers did not consider the obfuscation while training the models, therefore the performance of models would not be good on real time data. Secondly, a small dataset was used for training the models which is not recommended.

Table 5. *Cont.*

Title	Author	Year	Weakness/Limitation
Towards Automatic Reverse Engineering of Large Datasets of Binaries	M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw	2015	Evasion techniques, packed malware, and user interaction for triggering malware behavior were ignored while extracting features. Although all these phenomena are found in real time data. Therefore, the model's performance on real time data will not be accurate.
Subroutine based detection of APT malware	J. Sexton, C. Storlie, B. Anderson	2015	Obfuscated samples are quite commonly found in real time data and this research work ignored obfuscated samples while training the model.
On the comparison of malware detection methods using data mining with two feature sets	S. Srakaew, W. Piyanuntcharatsr, S. Adulkasem	2015	Ignored real distribution
A static signal processing based malware triage	D. Kirat, L. Nataraj, G. Vigna, B. Manjunat	2013	Ignored real distribution
Towards an Automated Pipeline for Detecting and Classifying Malware through Machine Learning	Nicola Loi et al.	2021	Used static features only

Table 6. Limitations of Surveyed Papers Proposing Deep Learning Based solutions for Malware Detection.

Title	Author	Year	Weakness/Limitation
Early Stage Malware Prediction Using Recurrent Neural Networks	Rhode, Matilda, et al.	2018	The system should be tested for large data. This approach can be a failure if the attacker comes to know that file is being monitored in the first 5 s so this can be evaded.
DL4MD: A Deep Learning Framework for Intelligent Malware Detection	Hardy, William, et al.	2016	Sparsity constraints are not imposed on SAE which can improve malware detection.
eXpose: A Character-Level Convolutional Neural Network with Embeddings for Detecting Malicious URLs, File Paths and Registry Key	Saxe, Joshua, and Konstantin Berlin.	2017	The computational cost of training on long strings is very high. This approach labeled any sample that had 0 occurrences in malware data as benign, and the rest was labeled malware. So, strings, file paths, and registry keys due to less training data can decrease this model's generalizability.
Robust Malware Detection for the Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning	Azmoodeh, Amin, Ali Dehghantanha, and Kim-Kwang Raymond Choo.	2018	Dataset was small for training the neural network which implies that the network could not learn the features at its best.
Detection of Malicious Code Variants Based on Deep Learning	Cui, Zhihua, et al.	2018	The model required all the input images to be of fixed size due to which images could have lost meaningful information while image processing.
Malware Identification Using visualization images and deep learning	Ni, Sang, Quan Qian, and Rui Zhang	2018	Detection of packed, encrypted malware or malware using anti-debugging and anti-dissembling approaches is not performed. Real time data consist of all these kind of malware, therefore network might not work well with real time data.
End-to-End Deep Neural Networks and Transfer Learning for Automatic Analysis of Nation State Malware	Rosenberg, Ishai, Guillaume Sicard, and Eli David.	2018	Due to the unavailability of nation-state APT, the proposed classifier is not evaluated for it. Moreover, static features provided by Cuckoo sandbox are not verified
Empowering Convolutional Networks for Malware Classification and Analysis	Kolosnjaji, Bojan, et al.	2017	Results of only static malware analysis are used. Code obfuscation can affect the proposed approach. A system unknown family of malware can affect the system's performance.
Malware Detection Based on Deep Learning of Behavior Graphs	Fei Xiao et al.	2019	The time of execution for extracting API calls is not mentioned. If the time of execution would have been small, then the results would not be reliable

Table 6. *Cont.*

Title	Author	Year	Weakness/Limitation
Deep Learning for Classification of Malware System Call Sequences	Bojan et al.	2016	The approach did not consider the evasion of malware detectors by inserting noise in system calls. Moreover, work depends on system calls' paths which depend on input data.
Malware Detection with Deep Neural Network Using Process Behavior	Shun Tobiyama et al.	2016	Due to the small amount of data, large Deep Neural Networks are not used.
Robust Intelligent Malware Detection Using Deep Learning	R. Vinaya Kumar et al.	2018	Malware are transformed into fixed-size images but can be converted to variable size to get good model learning.
Deep Neural Network-Based Malware Detection Using Two Dimensional Binary Program Features	Joshua et al.	2015	A small amount of data is used for getting low false positives, moreover only syntactic features are used, and semantic features are not focused.
Learning the PE Header, Malware Detection With Minimal Domain Knowledge	Edward Raff, Jared Sylvester, Charles Nicholas	2017	In one of the baseline approaches, features were not normalized
One-Shot Learning Approach for Unknown Malware Classification	True Kien, Hiroshi Sato, Masao Kubo	2018	For training, only malware samples were given. Malware families given for training were insufficient
Windows PE Malware Detection Using Ensemble Learning	Nureni Ayofe Azeez et al.	2021	Base ensemble classifiers only used static features.
Ensemble-Based Classification Using Neural Networks and Machine Learning Models for Windows PE Malware Detection	Robertas Damaševičius et al.	2021	Base ensemble classifiers only used static features.

3.1. Shortcomings of Primitive Methods (Statistical Analysis Based Methods) for Detecting Malware

Primitive methods of malware analysis depend upon statistical analysis of changes in the system or probabilistic explanation of an executable being malware based on the appearance of literals. But this probabilistic or statistical approach gives approximation over only a few features of malware and even gets stuck with obfuscated malware.

Packed executables were ignored by [21] and even the dataset was small which led to the uncertainty of results if the implemented solution is deployed in real-time. Ref. [19] made use of a detection algorithm that is context insensitive and is unable to track the calling context of the executable.

Another framework that was mentioned by [25] made use of windows audit logs but since windows audit logs can be obfuscated then in such case the presented solution is of no use. Secondly, researchers in [25] run the experiments for only 4 min which could have easily ignored the slow executing malware.

The solution given by [24] did not consider all those features which play an important role in the detection of malware.

The solution modeled by [22] used low interactive honeypots which allow only limited interaction of malware; thus, some malware can get undetected and get active only on the occurrence of certain conditions.

In the work done by [20], FPR is too high to implement the system in a real environment. The solution given by [67] tried to cater to metamorphism but dealt with only 3 techniques of obfuscation whereas there are many more techniques to obfuscate due to which claimed results cannot be reproduced in a real environment.

Hence, papers surveyed proposing the solutions for malware detection based on heuristic and statistical approaches, show that there is a need of adopting other techniques. Those techniques should be capable of improving FPR to generate a robust and reliable solution that can be implemented in real-time.

3.2. Shortcomings of Conventional Machine Learning Based Methods for Detecting Malware

In the case of static analysis being used by researchers, the foremost problem which hinders the analysis process is obfuscation, encryption, and packing. Refs. [34,35,68–71] have executed the solution without catering to the issue of obfuscation, packing, and encryption. One of the major problems seen in many papers during the survey is the problem of anti-analysis techniques which can be called evasion techniques also. Professional malware developers or in other words sophisticated malware developers take care of the fact that the target machine can be an analysis machine or can have a virtual environment setup, so they purposefully make use of evading techniques through which, normally, first they check for the presence of virtual environment and in case of its presence malware hibernates itself. This is called environmental awareness and is very clearly stated in [58]. Malware can easily comprehend and identify if it is being run in a virtual or debugging environment. Another evasion approach is timing-based which means malware gets only active at any date or time or gets activated at user interaction only. In solutions applied by [32,39,72–74] detection accuracy gets noticeably reduced on facing the evasion techniques, encrypted malware, and if malware needs user interaction for getting activated. Another problem that was identified during the survey was the small or insufficient datasets being used for analysis due to which results produced might not be reliable.

Researchers in [28,30,32,37,38,68–72,74–79] used small dataset. Since conventional machine learning algorithms are supposed to carry out the process of feature engineering, therefore, a very prominent problem that could be seen during the paper survey related to machine learning-based solutions for malware detection was the use of few features out of all those features which can very distinctively play a vital role in the detection of malware. Solutions carried through in [30,36,38,80,81] considered only a subset of useful features. Another shortcoming was the lack of capability of detecting the variants of malware.

3.3. Shortcomings of Deep Learning Based Methods for Malware Detection

The approach of deep learning has taken over the field of malware analysis because of its capability of automatic feature engineering but since still it is in the phase of evolution, therefore, certain issues still need to be catered to. One of the issues faced by deep learning-based methods is small data. The solution published by [43] indicates that the system was tested against small data and malware was executed for a very small time which can be easily catered by malware writers through evading techniques. Similarly, the research work of [46] used small data for training to avoid computational constraints but it affected the generalization. Again, the same problem was seen in the work of [47] due to which results of the given solution cannot be relied upon when implementing the presented framework in a real-time environment. Solutions given by [25,53,57] also suffer from the same problem.

Another problem that can be seen is the size of the input. Since CNN works over images and it is observed that most of the produced solutions work over the fixed size of images only. Solutions presented by [48,54] could perform better by handling variable size input data. Ref. [52] have not mentioned the execution time of samples for extracting API calls. In case samples would not have run for enough time, then claimed results would be non-reliable. Some of the proclaimed solutions have not catered to obfuscated samples due to which if they are implemented in real-time, their results will be affected on encountering packed or obfuscated samples.

Solutions communicated by [44,49,51] have not catered to the circumstances where evasion techniques could have been applied. In the research work of [45], sparsity constraint was not considered. Most of the solutions adapting dynamic analysis did not pay heed to multipath execution problems. Comparison between approaches carried out by [35] is not reliable because, in one of the approaches, features were not normalized whereas the value of features had a big range. Research work of [57] made use of only malware samples for malware classification although the real time system receives benign as well as malicious files so the system should have been trained on both types of files. Secondly, even malware families that were considered for training were too few.

The solution proposed by [82] is a stacked approach consisting of two stages. In the first stage, multiple base line machine learning based classifiers were used using the static features only. In the 2nd stage, the final classifier was used which worked over the dataset created by the predictions of the base classifiers used in the 1st stage. Similarly proposed methodology in [83] is also following the ensemble method. The first stage of ensemble classification in [83] is using multiple machine learning algorithms which are trained using static features only.

4. Direction for Future Work

There are different trivial problems that we have outlined in this paper and need to be addressed to produce a viable product capable of detecting malware in real-time. This section will highlight all such issues which need to be paid heed to, in future work.

4.1. Moderate Sized and Updated Dataset

As highlighted in the previous section, most of the survey papers have taken a small dataset which is not enough for research to produce reliable results. This problem is mainly due to the constraints of handling big data or due to the unavailability of the labelled datasets. Small datasets that have been used in research produce biased results that can't be reproduced in a real environment. Problems of unavailability of labeled data, imbalanced data, or unavailability of enough samples for a particular class of malware can be coped with through Few Shot Learning (FSL) and its variants. So that improved or state-of-the-art results can be achieved without jumping into the problems of handling and processing large datasets. Secondly, some of the datasets used for research purposes were quite old. Since malware is being produced daily with the latest and new characteristics, therefore, research carried out on outdated data might not be helpful in real-time. It is recommended that up-to-date data should be collected which should consist of all the latest variants of malware. Another issue that needs to be taken care of is the reflection of real data distribution in the datasets for training and validation of proposed frameworks.

4.2. Using Significant Features

The selection of appropriate features plays a vital role in training a model for producing effective results. Features extracted statically and dynamically both hold their contributions to the detection of malicious behavior. Most of the surveyed papers have used a subset of features or have used either statically extracted or in some cases dynamically extracted features only which paves way for the concern that some features which might be quite decisive in detecting malicious nature, may have been ignored. Many papers indicated that non-optimal features were focused on and should be taken care of in future work. Using the combination of static and dynamic features can train the model with better learned capabilities. To deploy the anti-malware system in real time environment, extracting dynamic features can pose a problem as per the limitations available in the market. In such a case, the application of neural networks can be helpful. Neural networks can deal with the images of samples of both malicious and benign classes. This way rather than focusing on any feature, all the static semantic features of the samples can be focused on. The usage of neural networks automates the process of feature engineering. Rather than selecting the features on the hit and trial method, embedding layers of the neural network can be used to automatically select the most contributing features.

4.3. Handling Evasion Techniques

As described earlier evasion techniques can be categorized as environment awareness-based or timing-based. A framework that can be claimed to be deployed in a real-time environment should be accurate and effective so that it does not get affected by evasion techniques. It should be taken care of in future work because new malware can detect the virtual environment. Another sophisticated capability of malware is to get activated at a particular date and time and till that activating time, it does not exhibit malicious behavior.

Some malware gets triggered over getting certain input otherwise they behave as a benign file. This kind of behavior can be traced by multipath execution which should be the focus of future work.

4.4. Combating Anti sAnalysis Techniques

Malware developers perform various anti-analysis techniques to suppress the detection and analysis of their released malware. Obfuscating a sample, compressing/packing the binary/exe, and encrypting the file, all are tools to make it difficult to detect and analyze the malware. Future work can be to mitigate all these anti-analysis tools to eradicate the possibility of the destructive threat posed by malware.

In short, the aggressively dynamic nature of the cyber world demands researchers to take care of the following points while conducting their research in this domain of malware detection.

- Since malware easily changes its shape due to sophisticated techniques used by malware writers so research in the future should be conducted with the motive of dealing with metamorphic, polymorphic, and obfuscated malware.
- The day-by-day increase in malware is the prime reason for the increasing no. of malware families and with the passage of a certain period various new forms of malware keep on showing up on the surface of the cyber world. Future research should focus on developing a generic model that should be capable of detecting zero day malware.
- To implement the real time solution, a model should be reliable enough to handle any kind of unseen malware as well.

Deep learning based research has proved to be fruitful by producing quotable results in the detection of malware. To further improve the solution, meta learning based algorithms can be exploited in conjunction with deep learning. Meta learning based algorithms help in producing generic models. These generic models are trained for self-learning. Through self-learning, the strategies of learning the properties of even unseen types of malware can be learned easily. More specifically few shot learning has proved itself worthy of being explored in the future due to its effectiveness, efficiency, and robustness.

5. Conclusions

In this survey paper, we investigated the research lack in building a real-time anti-malware system. This literature survey is about different techniques adapted to detect malware and analyze them. Work in this paper is organized in such a way that three different trends in techniques of detecting and analyzing malware are highlighted. Different malware detection trends have been categorized into primitive methods, which include statistical measures only, machine learning-based methods, and methods that involve new emerging technology of deep learning. The presented work's contributions include the distribution of techniques into three different trends, issues, and challenges faced by all different methods and directions of future work by mitigating all the issues faced by existing methods. Different statistical strategies are categorically highlighted that are used in the literature for detecting malware. Additionally, we shed light on machine learning algorithms and features that are used to detect malware. And finally, we discuss different deep learning models that are used in detecting and analyzing malware. This work indicates different issues related to datasets, the use of features' subsets, effects of evasion techniques, and hindrance caused by anti-analysis techniques.

Finally, future direction leading towards meta learning based algorithms have been suggested for producing a viable product capable of detecting and analyzing malware in real-time with improved accuracy.

Author Contributions: Conceptualization, U.-e.-H.T. and F.B.K.; Methodology, U.-e.-H.T.; Validation, A.K., M.H.D. and F.B.K.; Formal Analysis, U.-e.-H.T. and F.B.K.; Investigation, U.-e.-H.T. and F.B.K.; Resources, A.K. and M.H.D.; Data Curation, U.-e.-H.T. and F.B.K.; Writing—Original Draft Prepara-

tion, U.-e.-H.T. and F.B.K.; Writing—Review & Editing, A.K., M.H.D. and Y.S.L.; Supervision, A.K. and M.H.D.; Project Administration, A.K.; funding acquisition, A.K. and Y.S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: This research work was supported by the Higher Education Commission (HEC) Pakistan and the Ministry of Planning, Development and Special Initiatives under National Centre for Cyber Security. Moreover, we thank the CIPMA and PR Lab, Department of computer Information Sciences, PIEAS, Islamabad and department of Cybersecurity, AIR university, Islamabad for providing the necessary computational resources and a healthy research environment.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Taxonomy of Malware Analysis

Appendix A.1. Malware Types

Malware is a piece of code, which on executing performs, illegal actions such as stealing users' personal information, faltering the working of a system, creating any backdoor without user's information, and encrypting the data to make it useless for the user. Malware can be dedicated with the sole purpose to hinder the working of any system like Stuxnet as described by [84] or of a kind, which can victimize several systems or applications. Malware mostly falls into the following malware families:

- Virus: It can replicate itself by getting attached to any file/document. It has the potential to corrupt the system, destroy the data, and can pose a great threat to assets.
- Worm: It behaves just like a virus but can replicate itself over the network.
- Trojan horse: It masquerades itself as a useful program but contains malicious code.
- Backdoor: It gets itself installed on the system and gives access to the attacker without or with very little authentication.
- Botnet: It behaves just like a backdoor. The difference lies when it comes to the command and control server. All systems compromised by the botnet receive the same command from the same command and control server.
- Spyware: It behaves as a useful application but leaks users' data.
- Downloader: It is normally installed by the attacker on victims' machines. Its sole purpose is to download malicious code on the system.
- Rootkit: It gets paired with other malware and hides the existence of that malware. Another devastating effect of the rootkit is the root level access that it gives to the malware.
- Scareware: It frightens the users to buy their products to keep their data and system safe.
- Many malware fall into more than one category as they exhibit features of more than one malware family.

Malware analysis is the process that has become extremely important, not only to mitigate network attacks but massive destruction can also be prevented. These attacks can pave the way through the execution of malware on a standalone, dedicated system or by controlling no. of systems on a network.

Appendix A.2. Malware Analysis

Major objectives of malware analysis include:

- To gain the capability of responding to network intrusion
- To determine how can systems and files be infected
- To analyze the potential of suspected binaries/PE
- To devise the mechanism for identifying malware
- To find host-based signatures or indicators
- To find network-based signatures or indicators

- The scale of devastation that malware can pose

Normally in the case of malware what we get hold of for the sake of analysis are binary files or executables which are not easily understandable by humans. Therefore, different analysis techniques have been proposed to get full insights into malware. Broad categories of these techniques are shown in Figure A1.

Static Analysis: It refers to the phenomena of analyzing a file without executing it to keep the process of analysis safe. This approach includes the extraction of low-level information such as CFGs (Control Flow Graphs), DFGs (Data Flow Graphs), and system call analysis. Different tools can aid in static analysis such as IDAPro for disassembling the file. The static analysis gets failed when malware is obfuscated as it cannot penetrate through the packed samples as explained by [18].

Basic Static Analysis: It can confirm the maliciousness of the file. It can provide information about the functionality of malware, but it can't work with diligently programmed malware because of the lack of understanding of sophisticated malware's behavior.

Advanced Static Analysis: It refers to reverse engineering, which can be performed through a disassembler to understand the instruction code of the malware.

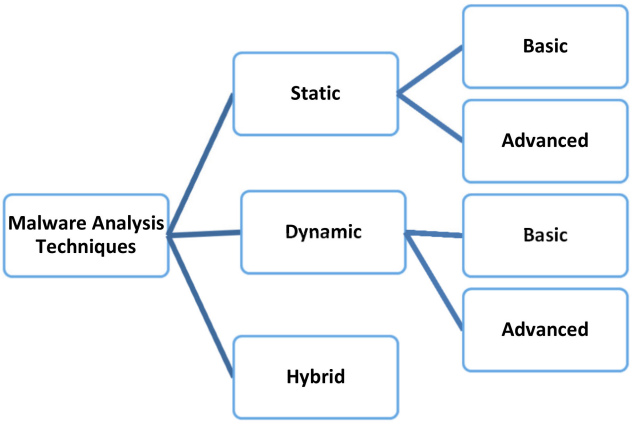

Figure A1. Malware Analysis Techniques.

Dynamic Analysis: When the file is executed in the safe/virtual environment for the sake of analysis then, it is called dynamic analysis It should be conducted by hiding the virtual environment from malware otherwise, malware can hibernate itself. This approach gets failed, when a particular triggering condition doesn't occur on which malware executes in its malicious state.

Basic Dynamic Analysis: It executes the malware in a safe environment to observe behavior to find any signature. It provides low-level information so cannot work with sophisticatedly programmed malware.

Advanced Dynamic Analysis: It uses a debugger to investigate the internal state of running malicious executables. It extracts detailed information, which helps in understanding the code as shown by [85].

Hybrid Analysis: This approach is a combination of both static and dynamic approaches. Researchers are trying to make use of the beneficial features of both approaches.

Table A1 refers to the summary of the advantages and disadvantages of static and dynamic approaches in malware analysis.

Table A1. Comparative Analysis of static and Dynamic Approaches to Malware Analysis.

	Advantages	Disadvantages
Static Analysis	• Fast and safe • Low FPR • Can analyze multipath	• Cannot deal with obfuscation • Cannot detect unknown malware
Dynamic Analysis	• Can deal with obfuscation • Can detect new malware • Can observe behavioral changes made by malware	• It is slow • Malware can hibernate on detecting a safe environment (high FPR) • Cannot trace multipath

Appendix B. Glossary of All Terms

This section is organized to help the reader get aware of some technical terms that he/she would come across quite frequently while reading this paper.

Obfuscation: Ref. [86] explains it as the process of hiding a code using different techniques so that malware can bypass security devices/software.

Polymorphism: Ref. [87] states it as the strategy through which malware keeps on changing its appearance to overcome detection. It is achieved through encryption using a different set of keys every time the malware executes.

Metamorphism: Using metamorphism malware changes its code and signature pattern but it is achieved without using encryption.

PE (Portable Executable): It is a file format for executables used in versions of windows.

Opcode: In machine language, the opcode is the part of instruction that refers to the operation.

DDOS: It is an acronym for Distributed Denial of Service, and it is categorized as a network attack.

Honeypot: It is a system attached to the network to attract cyber attackers as mentioned by [88] in their work. It works by luring the attackers away from the systems having critical info. Furthermore, it helps in observing the attacker's behavior and collecting information about the attacker's activity. Honeypots are the systems that imitate to contain the data values for the attacker, but these systems do not get accessed by legitimate users.

Ref. [89] further categorized into low interaction honeypots and high interaction honeypots. Low interaction honeypots contain software that emulates the real service whereas high interaction honeypots contain a complete operating system, services, and applications to give a complete real feeling of a valuable system to the attacker.

Machine Learning: It is a specialized field that comes under the hood of Artificial Intelligence. It makes use of AI to take decisions by mining the information from data as described by [90].

Supervised Learning: It is a learning technique used by AI-based algorithms for finding out the mapping function between input (x) and output (y) provided input and corresponding output.

Unsupervised Learning: It is a learning technique utilized by AI-based algorithms to find the underlying structure in data when only input is given.

Classification: It is a supervised learning technique that is applied when the output variable is a category and there is no relationship among the values of the output.

Regression: It is a supervised learning technique and is shed when the output variable is a real value and values of the output variable have a relationship (greater than or less than).

Clustering: It is an unsupervised learning technique in which data is divided into groups based on some similarity measure.

SVM: Support Vector Machine-It is a machine learning algorithm based on supervised learning and can be used for both classification and regression.

KNN: K Nearest Neighbour-It is a machine learning algorithm that works by measuring similarity.

Random Forest: It is a machine learning algorithm that can be used for both classification and regression.

Naïve Bayes: It is a supervised learning-based machine learning algorithm that works over applied Bayes.

LSH: It is a clustering-based machine learning algorithm.

Neural Networks: Neural networks also known as artificial neural networks are techniques of machine learning that simulate the process of learning by a human brain. The human brain consists of cells which are referred to as neurons in neural networks. Similarly, in a human brain, all the cells are connected through axons and dendrites with the connection region known as synapses. These connections when found in ANN (Artificial Neural Networks), contain weights to behave as the connections between nerve cells in the human brain. Figure A2 shows the human brain and simulated version of the human brain through the artificial neural network.

Deep Learning: Ref. [13] explained it as a specialized form of machine learning in the domain of Artificial Intelligence (AI) which applies deep artificial neural networks also known as deep neural networks. The major difference between conventional neural networks and deep neural networks is the number of layers. Deep neural networks make use of many hidden layers. Deep learning networks can be further categorized into different types of models such as deep neural networks (DNN), recurrent neural networks (RNN), and long short-term memory (LSTM). Unlike machine learning, it is capable to deal with unstructured data as well.

RNN: Recurrent Neural Network is a generalized form of feed-forward network that can handle sequential data by processing the current input as well as the previously received input stored in its internal memory (hidden units). The internal memory of RNN refers to the hidden units in intermediate or hidden layers which have got the capability of retaining and processing the previous inputs concerning time, having interdependency on each other. The Standard and unfolded architecture of RNN is shown in Figure A3. It is used where sequence and time series are important.

Autoencoder: According to [91] it is a type of feed forward neural network which makes use of an encoder and decoder to first compress the input and then decompress it. This process of compression and decompression is to learn the features of input first so that the same input can be reconstructed at the output. This is a type of NN that makes use of learned most important features of data to reconstruct it.

Stacked AutoEncoder: It is a neural network that consists of many AutoEncoder layers with the output of each layer connected to the input of the successive layer as explained by [89].

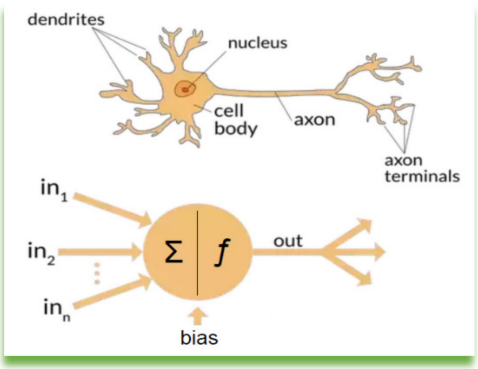

Figure A2. Human Brain and its Simulation Through ANN Ref. [82].

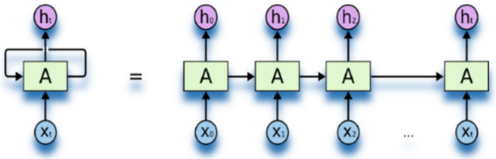

Figure A3. RNN Ref. [83].

References

1. *PandaLabs Annual Report 2018*; Panda Security: Chertsey, UK, 2018.
2. FBI. Addressing Threats to the Nations Cybersecurity 1. FBI Report, Retrieved 3 August 2022. Available online: https://www.fbi.gov/file-repository/addressing-threats-to-the-nations-cybersecurity-1.pdf/view (accessed on 10 August 2022).
3. Manavi, F.; Hamzeh, A. A novel approach for ransomware detection based on PE header using graph embedding. *J. Comput. Virol. Hacking Tech.* **2022**, *14*, 1–12. [CrossRef]
4. Zahoora, U.; Rajarajan, M.; Pan, Z.; Khan, A. Zero-day Ransomware Attack Detection using Deep Contractive Autoencoder and Voting based Ensemble Classifier. *Appl. Intell.* **2022**, 1–20. [CrossRef]
5. Mohurle, S.; Patil, M. A brief study of Wannacry Threat: Ransomware Attack 2017. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 1938–1940.
6. Maria Vergelis, T.S. *Spam and Phishing in Q2 2019*; SecureList by Kaspersky: Moscow, Russia, 2019.
7. *ISTR Internet Security Threat Report*; Symantec: Tempe, AZ, USA, 2019; Volume 24.
8. Cyberattacks. Available online: https://www.cnbc.com/2019/10/13/cyberattacks-cost-small-companies-200k-putting-many-out-of-business.html (accessed on 9 March 2022).
9. Baezner, M.; Robin, P.; Wenger, A. Stuxnet. 2017. Available online: https://css.ethz.ch/ (accessed on 5 July 2020).
10. Mo, Y.; Chabukswar, R.; Sinopoli, B. Detecting integrity attacks on SCADA systems. *IEEE Trans. Control Syst. Technol.* **2014**, *22*, 1396–1407. [CrossRef]
11. Marelli, D.; Sui, T.; Fu, M.; Lu, R. Statistical Approach to Detection of Attacks for Stochastic Cyber-Physical Systems. *IEEE Trans Autom. Contr* **2021**, *66*, 849–856. [CrossRef]
12. Sui, T.; Mo, Y.; Marelli, D.; Sun, X.; Fu, M. The Vulnerability of Cyber-Physical System under Stealthy Attacks. *IEEE Trans Autom. Contr* **2021**, *66*, 637–650. [CrossRef]
13. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
14. Souri, A.; Hosseini, R. A state-of-the-art survey of malware detection approaches using data mining techniques. *Hum. Cent. Comput. Inf. Sci.* **2018**, *8*, 3. [CrossRef]
15. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2019**, *81*, 123–147. [CrossRef]
16. Mahdavifar, S.; Ghorbani, A.A. Application of deep learning to cybersecurity: A survey. *Neurocomputing* **2019**, *347*, 149–176. [CrossRef]
17. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2019**, *10*, 122. [CrossRef]
18. Komatwar, R.; Kokare, M. A Survey on Malware Detection and Classification. *J. Appl. Secur. Res.* **2021**, *16*, 390–420. [CrossRef]
19. Christodorescu, M.; Jha, S. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th USENIX Security Symposium (USENIX Security 03), Washington, DC, USA, 4–8 August 2003. [CrossRef]
20. Santos, I. Idea: Opcode-sequence-based malware detection. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 5965. [CrossRef]
21. Sabbatel, G.B.; Korczynski, M.; Duda, A. Architecture of a Platform for Malware Analysis and Confinement. In Proceedings of the Proceeding MCSS 2010: Multimedia Communications, Services and Security, Cracow, Poland, 2–3 June 2011.
22. Elhadi, A.A.E.; Maarof, M.A.; Osman, A.H. Malware detection based on hybrid signature behavior application programming interface call graph. *Am. J. Appl. Sci.* **2012**, *9*, 283–288. [CrossRef]
23. Fleck, D.; Tokhtabayev, A.; Alarif, A.; Stavrou, A.; Nykodym, T. PyTrigger: A system to trigger & extract user-activated malware behavior. In Proceedings of the 2013 International Conference on Availability, Reliability and Security, Regensburg, Germany, 2–6 September 2013. [CrossRef]
24. Berlin, K.; Slater, D.; Saxe, J. Malicious behavior detection using windows audit logs. In Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, Denver, CO, USA, 16 October 2015. [CrossRef]
25. Kumar, G.; Thakur, K.; Ayyagari, M.R. MLEsIDSs: Machine learning-based ensembles for intrusion detection systems—A review. *J. Supercomput.* **2020**, *76*, 8938–8971. [CrossRef]
26. Chen, L.; Li, T.; Abdulhayoglu, M.; Ye, Y. Intelligent malware detection based on file relation graphs. In Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), Anaheim, CA, USA, 7–9 February 2015. [CrossRef]
27. Elhadi, A.A.E.; Maarof, M.A.; Barry, B.I.A. Improving the detection of malware behaviour using simplified data dependent API call graph. *Int. J. Secur. Its Appl.* **2013**, *7*, 29–42. [CrossRef]
28. Feng, Z.; Xiong, S.; Cao, D.; Deng, X.; Wang, X.; Yang, Y.; Zhou, X.; Huang, Y.; Wu, G. HRS: A Hybrid Framework for Malware Detection. In Proceedings of the 2015 ACM International Workshop on International Workshop on Security and Privacy Analytics, San Antonio, TX, USA, 4 March 2015. [CrossRef]

29. Ghiasi, M.; Sami, A.; Salehi, Z. Dynamic VSA: A framework for malware detection based on register contents. *Eng. Appl. Artif. Intell.* **2015**, *44*, 111–122. [CrossRef]
30. Kwon, B.J.; Dumitras, T. The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics Categories and Subject Descriptors. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (Ccs'15), Denver, CO, USA, 12–16 October 2015.
31. Mao, W.; Cai, Z.; Towsley, D.; Guan, X. Probabilistic inference on integrity for access behavior based malware detection. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2015; Volume 9404. [CrossRef]
32. Piyanuntcharatsr, S.S.W.; Adulkasem, S.; Chantrapornchai, C. On the comparison of malware detection methods using data mining with two feature sets. *Int. J. Secur. Its Appl.* **2015**, *9*, 293–318. [CrossRef]
33. Wüchner, T.; Ochoa, M.; Pretschner, A. Robust and effective malware detection through quantitative data flow graph metrics. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2015; Volume 9148. [CrossRef]
34. Raff, E.; Nicholas, C. An alternative to NCD for large sequences, lempel-ZiV jaccard distance. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; Volume 129685. [CrossRef]
35. Khodamoradi, P.; Fazlali, M.; Mardukhi, F.; Nosrati, M. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. In Proceedings of the 18th CSI International Symposium on Computer Architecture and Digital Systems, (CADS 2015), Tehran, Iran, 7–8 October 2015. [CrossRef]
36. Upchurch, J.; Zhou, X. Variant: A malware similarity testing framework. In Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 20–22 October 2015. [CrossRef]
37. Liang, G.; Pang, J.; Dai, C. A Behavior-Based Malware Variant Classification Technique. *Int. J. Inf. Educ. Technol.* **2016**, *6*, 291. [CrossRef]
38. Vadrevu, P.; Perdisci, R. MAXS: Scaling malware execution with sequential multi-hypothesis testing. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, 30 May–3 June 2016. [CrossRef]
39. Dahl, G.E.; Stokes, J.W.; Deng, L.; Yu, D. Large-scale malware classification using random projections and neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013. [CrossRef]
40. Ravi, V.; Alazab, M.; Selvaganapathy, S.; Chaganti, R. A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems. *Comput. Commun.* **2022**, *195*, 73–81. [CrossRef]
41. Rama, K.; Kumar, P.; Bhasker, B. Deep Learning to Address Candidate Generation and Cold Start Challenges in Recommender Systems: A Research Survey. *arXiv* **2019**, arXiv:1907.08674.
42. Rhode, M.; Burnap, P.; Jones, K. Early-stage malware prediction using recurrent neural networks. *Comput Secur* **2018**, *77*, 578–594. [CrossRef]
43. Kolosnjaji, B.; Zarras, A.; Webster, G.; Eckert, C. Deep learning for classification of malware system call sequences. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2016; Volume 9992. [CrossRef]
44. Hardy, W.; Chen, L.; Hou, S.; Ye, Y.; Li, X. *DL 4 MD: A Deep Learning Framework for Intelligent Malware Detection*; CSREA Press: Las Vegas, NV, USA, 2016; pp. 61–67.
45. Saxe, J.; Berlin, K. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. *arXiv* **2017**, arXiv:1702.08568.
46. Azmoodeh, A.; Dehghantanha, A.; Choo, K.K.R. Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning. *IEEE Trans. Sustain. Comput.* **2019**, *4*, 88–95. [CrossRef]
47. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans Ind. Inf.* **2018**, *14*, 3187–3196. [CrossRef]
48. Ni, S.; Qian, Q.; Zhang, R. Malware identification using visualization images and deep learning. *Comput Secur* **2018**, *77*, 871–885. [CrossRef]
49. Rosenberg, I.; Sicard, G.; David, E. End-to-end deep neural networks and transfer learning for automatic analysis of nation-state malware. *Entropy* **2018**, *20*, 390. [CrossRef]
50. Kolosnjaji, B.; Eraisha, G.; Webster, G.; Zarras, A.; Eckert, C. Empowering convolutional networks for malware classification and analysis. In Proceedings of the International Joint Conference on Neural Networks, Anchorage, AK, USA, 14–19 May 2017. [CrossRef]
51. Xiao, F.; Lin, Z.; Sun, Y.; Ma, Y. Malware Detection Based on Deep Learning of Behavior Graphs. *Math. Probl. Eng.* **2019**, *2019*, 8195395. [CrossRef]
52. Tobiyama, S.; Yamaguchi, Y.; Shimada, H.; Ikuse, T.; Yagi, T. Malware Detection with Deep Neural Network Using Process Behavior. In Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 10–14 June 2016; Volume 2. [CrossRef]
53. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Venkatraman, S. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access* **2019**, *7*, 46717–46738. [CrossRef]

54. David, O.E.; Netanyahu, N.S. DeepSign: Deep learning for automatic malware signature generation and classification. In Proceedings of the International Joint Conference on Neural Networks, Killarney, Ireland, 12–17 July 2015. [CrossRef]
55. Saxe, J.; Berlin, K. Deep neural network based malware detection using two dimensional binary program features. In Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 20–22 October 2015. [CrossRef]
56. Tran, T.K.; Sato, H.; Kubo, M. One-shot learning approach for unknown malware classification. In Proceedings of the 2018 5th Asian Conference on Defense Technology (ACDT), Hanoi, Vietnam, 25–26 October 2018. [CrossRef]
57. Raff, E.; Sylvester, J.; Nicholas, C. Learning the PE header, malware detection with minimal domain knowledge. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017. [CrossRef]
58. Bensaoud, A.; Kalita, J. Deep multi-task learning for malware image classification. *J. Inf. Secur. Appl.* **2022**, *64*, 103057. [CrossRef]
59. Kumar, S.; Janet, B. DTMIC: Deep transfer learning for malware image classification. *J. Inf. Secur. Appl.* **2022**, *64*, 103063. [CrossRef]
60. Mohammadi, F.G.; Amini, M.H.; Arabnia, H.R. An introduction to advanced machine learning: Meta-learning algorithms, applications, and promises. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2020; Volume 1123. [CrossRef]
61. Kadam, S.; Vaidya, V. Review and analysis of zero, one and few shot learning approaches. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2020; Volume 940. [CrossRef]
62. Hsiao, S.C.; Kao, D.Y.; Liu, Z.Y.; Tso, R. Malware image classification using one-shot learning with siamese networks. *Procedia Comput. Sci.* **2019**, *159*, 1863–1871. [CrossRef]
63. Tran, T.K.; Sato, H.; Kubo, M. Image-based unknown malware classification with few-shot learning models. In Proceedings of the 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), Nagasaki, Japan, 26–29 November 2019. [CrossRef]
64. Tang, Z.; Wang, P.; Wang, J. ConvProtoNet: Deep prototype induction towards better class representation for few-shot malware classification. *Appl. Sci.* **2020**, *10*, 2847. [CrossRef]
65. Atapour-Abarghouei, A.; Bonner, S.; McGough, A.S. A King's Ransom for Encryption: Ransomware Classification using Augmented One-Shot Learning and Bayesian Approximation. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019. [CrossRef]
66. Lee, J.; Jeong, K.; Lee, H. Detecting metamorphic malwares using code graphs. In Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre, Switzerland, 22–26 March 2010. [CrossRef]
67. Santos, I.; Devesa, J.; Brezo, F.; Nieves, J.; Bringas, P.G. OPEM: A static-dynamic approach for machine-learning-based malware detection. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2013; Volume 189. [CrossRef]
68. Pai, S.; di Troia, F.; Visaggio, C.A.; Austin, T.H.; Stamp, M. Clustering for malware classification. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 95–107. [CrossRef]
69. Polino, M.; Scorti, A.; Maggi, F.; Zanero, S. Jackdaw: Towards automatic reverse engineering of large datasets of binaries. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2015; Volume 9148. [CrossRef]
70. Sexton, J.; Storlie, C.; Anderson, B. Subroutine based detection of APT malware. *J. Comput. Virol. Hacking Tech.* **2016**, *12*, 225–233. [CrossRef]
71. Lin, C.T.; Wang, N.J.; Xiao, H.; Eckert, C. Feature selection and extraction for malware classification. *J. Inf. Sci. Eng.* **2015**, *31*, 965–992.
72. Mohaisen, A.; Alrawi, O.; Mohaisen, M. AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Comput Secur* **2015**, *52*, 251–266. [CrossRef]
73. Lindorfer, M.; Kolbitsch, C.; Milani Comparetti, P. Detecting environment-sensitive malware. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2011; Volume 6961. [CrossRef]
74. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
75. Park, Y.; Reeves, D.; Mulukutla, V.; Sundaravel, B. Fast malware classification by automated behavioral graph matching. In Proceedings of the 6th Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW '10), Oak Ridge, TN, USA, 21–23 April 2010. [CrossRef]
76. Islam, R.; Tian, R.; Batten, L.M.; Versteeg, S. Classification of malware based on integrated static and dynamic features. *J. Netw. Comput. Appl.* **2013**, *36*, 646–656. [CrossRef]
77. Nari, S.; Ghorbani, A.A. Automated malware classification based on network behavior. In Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA, 28–31 January 2013. [CrossRef]
78. Kawaguchi, N.; Omote, K. Malware function classification using apis in initial behavior. In Proceedings of the 2015 10th Asia Joint Conference on Information Security, Kaohsiung, Taiwan, 24–26 May 2015. [CrossRef]
79. Gharacheh, M.; Derhami, V.; Hashemi, S.; Fard, S.M.H. Proposing an HMM-based approach to detect metamorphic malware. In Proceedings of the 2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Zahedan, Iran, 9–11 September 2015. [CrossRef]

80. Loi, N.; Borile, C.; Ucci, D. Towards an Automated Pipeline for Detecting and Classifying Malware through Machine Learning. *arXiv* **2021**, arXiv:2106.05625.
81. Azeez, N.A.; Odufuwa, O.E.; Misra, S.; Oluranti, J.; Damaševičius, R. Windows PE malware detection using ensemble learning. *Informatics* **2021**, *8*, 10. [CrossRef]
82. Damaševičius, R.; Venčkauskas, A.; Toldinas, J.; Grigaliūnas, Š. Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. *Electronics* **2021**, *10*, 485. [CrossRef]
83. Langner, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
84. Roseline, S.A.; Geetha, S.; Kadry, S.; Nam, Y. Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm. *IEEE Access* **2020**, *8*, 206303–206324. [CrossRef]
85. Barriga, J.J.A.; Yoo, S.G. Malware detection and evasion with machine learning techniques: A survey. *Int. J. Appl. Eng. Res.* **2017**, *12*, 7207–7214.
86. Kim, K.; Moon, B.R. Malware detection based on dependency graph using hybrid genetic algorithm. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, Portland, OR, USA, 7–11 July 2010. [CrossRef]
87. Sanders, C.; Smith, J. *Applied Network Security Monitoring*; Elsevier: Amsterdam, The Netherlands, 2014. [CrossRef]
88. William Stallings, L.B. *Computer Security: Principles and Practice*, 4th ed.; Pearson: Upper Saddle River, NJ, USA, 2021.
89. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
90. Vinh, P.C. Context-Aware Systems and Applications (ICCASA 2018) and Nature of Computation and Communication (ICTCC 2018). *Mob. Netw. Appl.* **2019**, *24*, 80–81. [CrossRef]
91. Chouhan, N.; Khan, A.; Rasheed, R.; Khan, H. Network anomaly detection using channel boosted and residual learning based deep convolutional neural network. *Appl. Soft Comput. J.* **2019**, *83*, 105612. [CrossRef]