Zayed University

ZU Scholars

All Works

10-10-2022

Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review

Rahman Ali
University of Peshawar

Asmat Ali
University of Peshawar

Farkhund Iqbal
Zayed University, farkhund.iqbal@zu.ac.ae

Mohammed Hussain
Zayed University, mohammed.hussain@zu.ac.ae

Farhan Ullah
Northwestern Polytechnical University

Recommended Citation

Ali, Rahman; Ali, Asmat; Iqbal, Farkhund; Hussain, Mohammed; and Ullah, Farhan, "Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review" (2022). *All Works*. 5406.
https://zuscholars.zu.ac.ae/works/5406

Hindawi
Security and Communication Networks
Volume 2022, Article ID 2959222, 31 pages
https://doi.org/10.1155/2022/2959222

Review Article

Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review

Rahman Ali,[1] **Asmat Ali**,[2] **Farkhund Iqbal**,[3] **Mohammed Hussain**,[4] **and Farhan Ullah**[5]

[1] *QACC, University of Peshawar, Peshawar, Pakistan*
[2] *Department of Computer Science, University of Peshawar, Peshawar, Pakistan*
[3] *College of Technological Innovation, Zayed University, Dubai, UAE*
[4] *College of Technological Innovation, Zayed University, Dubai, UAE*
[5] *School of Software, Northwestern Polytechnical University, 127 West Youyi Road, Beilin District, Xi'an 710072, China*

Correspondence should be addressed to Farhan Ullah; farhan@nwpu.edu.cn

Received 4 August 2021; Revised 10 August 2022; Accepted 12 September 2022; Published 10 October 2022

Academic Editor: Andrea Michienzi

Android and Windows are the predominant operating systems used in mobile environment and personal computers and it is expected that their use will rise during the next decade. Malware is one of the main threats faced by these platforms as well as Internet of Things (IoT) environment and the web. With time, these threats are becoming more and more sophisticated and detecting them using traditional machine learning techniques is a hard task. Several research studies have shown that deep learning methods achieve better accuracy comparatively and can learn to efficiently detect and classify new malware samples. In this paper, we present a systematic literature review of the recent studies that focused on intrusion and malware detection and their classification in various environments using deep learning techniques. We searched five well-known digital libraries and collected a total of 107 papers that were published in scholarly journals or preprints. We carefully read the selected literature and critically analyze it to find out which types of threats and what platform the researchers are targeting and how accurately the deep learning-based systems can detect new security threats. This survey will have a positive impact on the learning capabilities of beginners who are interested in starting their research in the area of malware detection using deep learning methods. From the detailed critical analysis, it is identified that CNN, LSTM, DBN, and autoencoders are the most frequently used deep learning methods that have effectively been used in various application scenarios.

1. Introduction

Deep learning (DL) is a representation learning approach having multiple levels of representation, each of which transforms the representation at one level to a representation at a higher level allowing very complex functions to be learned. Deep learning methods may help in solving the problems that have been a challenge for the machine learning community. It has been found very useful in discovering complex structures in high-dimensional data and is applied in various domains of government, business, and science. Representation learning methods allow us to feed a machine with raw data and discover the representation we need for classification or detection [1, 2]. Machine learning is used for enhancing the functionality of computer systems and data processing systems in various fields, like medicine, research, robotics, web search engines, content filtering, and recommendation systems on social networks and e-commerce platforms, and in products, like cameras and smartphones. However, traditional machine learning techniques have certain limitations in processing data in raw form [2].

Malware is a piece of code written with the intent to cause harm to, or subvert, the functionality of a computer system. It is a general term that describes viruses, spyware, Trojans, adware, and other types of malicious code [1]. A malware detecting system is a system that tries to determine whether or not a specific program or piece of code is malicious [1, 3]. Malware developers use obfuscation methods to change the form of malware and deceive virus scanners that use pattern matching techniques because they are purely semantic and ignore the instructions' semantics and the pattern matching techniques are not resilient to variations [3]. Malware designed today are polymorphic and metamorphic, having the ability to change their code with propagation. Malware variants share various behavioral patterns that could be exploited to detect unknown malware using machine learning techniques that could not be achieved by using the traditional malware detection techniques [4]. Deep learning techniques have been widely used in various fields, including computer vision, speech recognition, pattern recognition, NLP, and malware detection and classification, and have become an active area of research. It is a challenging task to develop systems that can detect any kind of malicious code accurately in a short time. Due to the increasing number and variants of malware, a malware detection system should automatically perform without or with minimal human intervention. Moreover, signature-based malware detection techniques are not sufficient to fight against malware as they could be easily deceived in an intelligent manner [5]. Using deep learning methods for malware detection and classification enables us to build scalable models that may handle any measure of data and improve their accuracy as they can identify more features than traditional machine learning methods. After the training phase, DL models can acquire a new pattern of malware easily [6].

From the extensive literature survey of the deep learning methods in the area of malware and intrusion detection, one can understand that researchers have worked in this direction; however, the majority of studies focus on deep learning-based methods or they are related to a specific type of malware (e.g., android malware detection or Windows-based malware detection or network anomaly detection, etc.). Very few surveys can be found, such as [7–9], discussing the subject area; however, they reviewed a very limited number of research works and techniques.

A separate study for each type of malware could be conducted, such as Windows-based malware, Android-based malware, intrusion, network anomalies, and other threats to security. Moreover, it is also a good idea to conduct an SLR for different types of threats, such as ransomware; however, we feel the need to present a broad picture and provide a broad-scoped vision to the new researchers in the area, separating the different platforms by discussing them in different sections.

A large amount of data is being produced online and in various organizations, which makes this era the era of big data. Traditional data processing and traditional machine learning methods may not be able to process this data and extract insights from it. Deep learning is thought to be more capable of processing large volumes of data and enable the researchers to reach better solutions. During the recent years, researchers have been focusing on deep learning methods for malware and threats detection and classification. Keeping in view the effectiveness and importance of deep learning techniques and the recent trends in research, we conducted a systematic literature review (SLR) of DL techniques used in malware and intrusion detection systems in the last six years, 2015 to 2022.

The aim of this study is to identify the scope, trends, and methods of deep learning algorithms exploited in the area of malware detection systems for various platforms and develop a better understanding of the new researchers in this area. To achieve the said aim, the following objectives are set as research questions.

(i) RQ1: Which platforms are affected the most by various types of malware attacks?

(ii) RQ2: What are different malware analytics (hot era of the research for malware detection, most vulnerable platform and most popular journals publishing malware detection literature)?

(iii) RQ3: Which methods are used by the research community for malware detection?

(iv) RQ4: What are the major DL algorithms used in the domain of malware detection?

(v) RQ5: What are the main challenges that we face in using deep learning methods for malware and intrusion detection?

(vi) RQ6: Do the proposed android malware detection systems, approaches, and studies support the characteristics of sustainability, evolvability, and automatically picking the most appropriate malware detection DL algorithm?

(vii) RQ7: Do the Android-based surveyed methods have the ability of identifying any newly introduced malware and what type of feature analysis technique (s) the researchers have used in their research?

(viii) RQ8: What are the most commonly used datasets used for malware detection in the Android-based and Windows-based environments?

The rest of the paper is structured as follows. In Section 2, a summary of the surveys done in the subject area is made. In Section 3, the research methodology is elaborated from different perspectives. In this section, we summarize the selected studies and present a complete picture of how DL methods are used for malware detection in Windows and Android platforms. In Section 4, experiments are performed to show how DL methods supersede traditional ML methods. Section 5 critically discusses different aspects of the study. Section 6 recommends key points to the readers and those who are new to the area. Finally, Section 7 provides a conclusion of the study.

2. Existing Surveys on Deep Learning Methods for Malware Detection

There have been a number of relevant studies in the domain of malware and intrusion detection systems using machine learning, deep learning, or other methods. Some of these studies are summarized in Table 1.

Table 1: State-of-the-art research in deep learning for malware detection.

Ref.	Year	Description	Limitations
[10]	2019	Survey of approaches that detect permissions demanded by apps that might be used for malicious activities	(i) Limited to android malware detection (ii) Permission-based malware detection only
[11]	2017	Deep learning techniques to detect network anomalies	(i) Only network anomaly and intrusion detection systems
[12]	2019	Survey of approaches to network anomaly detection	(i) Only network anomaly detection (ii) Traditional learning based
[13]	2018	Different malware detection techniques, like signature- and behavior-based detection	(i) Not limited to deep learning
[14]	2018	A survey of intrusion detection techniques in vehicular ad hoc networks	(i) Limited to intrusion detection systems in vehicular networks
[6]	2018	A survey of android malware analysis using deep learning with static analysis, dynamic analysis, and hybrid analysis	(i) Limited to android malware detection (ii) Reviewed very few publications
[15]	2019	A survey of machine learning techniques used in cyber security, like spam detection, phishing detection, and malware detection	(i) Traditional learning-based systems (ii) Limited to cyber security
[7]	2019	Review of deep learning-based android malware detection techniques	(i) Limited to android malware detection (ii) Reviewed very few publications
[16]	2019	A review of different intrusion detection systems in IoT, including anomaly based, specification based, signature based, and hybrid IDS's	(i) IoT-based systems only (ii) Not limited to DL- and ML-based methods
[17]	2018	A survey of IDS's and defense systems in IoT	(i) IoT-based systems only (ii) Not limited to DL and ML
[5]	2018	A survey of applications of deep learning techniques in malware detection	(i) Reviewed only the different DL algorithms used for malware detection
[18]	2020	A survey of deep learning techniques in defense against phishing	(i) Limited to phishing attacks only
[19]	2019	A survey of android malware detection systems	(i) Limited to Android malware (ii) Not DL or ML based
[20]	2019	A survey of techniques for security in IoT	(i) Only IoT-based systems (ii) Not limited to DL
[21]	2019	A survey of intrusion detection systems using deep learning	(i) Only intrusion detection systems
[8]	2020	A review of malware detection systems using deep learning	(i) Reviewed only about 35 publications
[22]	2020	An extensive review of malware detection systems based on deep learning	(i) Limited to Android malware
[9]	2020	A review of DL algorithms used for malware detection and some relevant literature	(i) Reviewed very few publications
Proposed study		Deep learning methods for malware and intrusion detection—a systematic literature review	Key contributions (i) One of the extensive surveys covering a large number of research articles (94) in Windows-, Android-, and IoT-based environments for malware and intrusion detection using deep learning approaches. (ii) Extraction of deeper malware analytics (iii) Extraction of useful information about deep learning methods applied in the domain of malware and intrusion detection (iv) Identification of the most effective deep learning algorithms for malware and intrusion detection (v) Highlighting the key challenges faced during the use of deep learning methods for malware and intrusion detection

It is evident from the last column of Table 1 that these surveys are related to malware or intrusion detection systems; however, most of them are not deep learning-based or related to a specific type of malware (e.g., android malware detection or network anomaly detection). Very few surveys were found that reviewed malware detection systems using deep learning, such as [7–9]; however, they reviewed a very limited number of research works and techniques. In the proposed study, we have tried our level best to cover the use of deep learning algorithms applied to Windows-based, Android-based, and IoT-based environments for the detection and classification of deep learning algorithms. This will provide a base to the beginners in the area to start their research work and easily find the gap for further improvements and start of new research.

Key contributions of the survey are enlisted as follows:

(i) One of the extensive surveys covering a large number of research articles (94) in Windows-, Android-, and IoT-based environments for malware and intrusion detection using deep learning approaches

(ii) Extraction and formulation of malware analytics from the relevant literature on DL methods used for malware and intrusion detection

(iii) An extensive study of the relevant literature to extract useful information about deep learning methods used in the domain of malware and intrusion detection systems

(iv) An analysis of which deep learning algorithms are used in malware and intrusion detection systems

(v) Highlighting the key challenges faced by the research community in using deep learning methods for malware and intrusion detection

3. Research Methodology

To assess the applications and impact of deep learning methods for malware and intrusion detection systems, a systematic literature review (SLR) is conducted. In an SRL, a systematic approach is followed to identify and analyze relevant studies regarding a specific area of interest [23]. There are a number of guidelines defined by experts for conducting a comprehensive and effective SLR. We studied and followed the guidelines provided in [24, 25].

3.1. Research Design. Firstly, we do need an assessment to conduct the literature review. We studied a number of papers from various sources, including journals and conferences. We observed that deep learning has been used by a large number of researchers in malware detection, intrusion detection, and other security-related systems. Based on the need assessment, we formulated a number of research questions, mentioned in the introduction section, to show the effectiveness and outcomes of the proposed study.

We then selected a number of search libraries to make sure the selected literature is from an authentic source and is reliable. We selected five different libraries, including Google Scholar, Science Direct, IEEE Explore, ACM, and Springer Link. Next, we formulated a search query to retrieve only the relevant studies focusing on deep learning for malware and intrusion detection. We identified keywords, based on the research questions that represent and reflect the RQs to objectively search the relevant publications. It is not effective to search the libraries using individual words; instead, the search keywords are combined by using "AND" and "OR" operators to generate queries that may return only relevant results. The search query formulated was finally fed into the searching mechanism of each library to retrieve the relevant studies from the last six years, 2015 to 2022. We searched these libraries for the previous years as well; however, we found very few, or no relevant studies during these years; thus, we limited our search to the recent years only. Moreover, we intend to focus on the latest state-of-the-art research in our study. We selected the primary studies by analyzing the title and abstract of the publications. We created an EndNote library and downloaded all the selected papers which were then read in full to exclude the papers which were not relevant to the theme of the current study and generated a final set of the relevant studies. This set of publications was then studied to answer the research questions (RQs) and to achieve the objectives of the study.

A tracer list for the findings and insights of each RQ is given below to easily guide the readers towards the respective section of the paper for better comprehension. Section 3.2.1 describes different platforms and extracts the insights regarding the severely affected platform so that to find answer to RQ1. Section 3.2.1 extracts different malware analytics and tries to find answer to RQ2. Column 3 of Tables 2–6 of Sections 4–8, respectively, extracts insights regarding the DL-based malware detection methods and tries to find answer to RQ3, insights for RQ4, which are the major DL algorithms used so far and have been discussed in Section 9. The key research challenges faced during implementation of DL-based malware detection systems are focused in RQ5 and described in Section 11.3. RQ6 and RQ7 describe the issues of sustainability and evolvability which are also discussed in Section 11.3 and RQ8 extracts insights of the most commonly used datasets for malware detection in Section 11.4.

3.1.1. Criteria for the Survey. As deep learning-based security systems and other intelligent software tools are getting more and more popular in the field of computer science, it is important to conduct research about how deep learning technology is performing better than the traditional approach for threat hunting and traditional machine learning-based systems. In this survey, we are interested in analyzing which deep learning algorithms are used in malware detection systems and how they can perform better when compared with traditional machine learning-based systems. We carefully read the selected literature to find out which types of threats and what platforms the researchers are targeting and how accurately the deep learning-based systems can detect new security threats when trained with a training dataset. Moreover, in the case of Android-based malware detection research, the focus is also given to know whether the proposed methods support the characteristics of sustainability, evolvability, automatically picking the most appropriate malware detection DL algorithm, ability of identifying new malware (s), and diverse feature analysis methods, such as static, dynamic, or hybrid. Apart from the above, the survey also discusses the issue of data quality for deep learning-based methods in the form of publicly available malware datasets for research purpose.

3.1.2. Query Formulation. As we discussed earlier, it is not the correct way to search each library using a single keyword each time as it is a very tedious and lengthy process. We need to formulate a search query containing all the important keywords including names, synonyms, and abbreviations,

Table 2: Summary of the metadata extracted from the literature on Windows-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Platform	Dataset used	Accuracy/F_1 score
[26]	Malware classification by extracting static features and converting to gray images	CNN	Not stated	Windows	Kaggle by Microsoft	98.86%
[27]	Malware classification by converting malware binary file to gray image through code mapping, texture partitioning, and texture extraction	CNN	Not stated	Windows	BIG 2015	99%
[28]	Malware classification by extracting series of system calls having malicious behavior	Not stated	Not stated	Windows	Self-generated	95.6%
[29]	Malware detection and classification by using the op-code and API calls data of malware and benign-ware	CNN, BPNN	Not stated	Windows	Self-generated	95%
[30]	Multilevel deep learning system for malware detection using different static and dynamic features	Proposed MLDLS	Not stated	Windows	Self-generated	Not stated
[31]	Ransomware detection system based on n-gram op-code with deep learning	CNN	Keras, TensorFlow	Windows	Self-generated	89.5%
[32]	Malware detection by transforming PE file to op-code sequences and representing the op-code as n-gram vector	DBN	Not stated	Windows	Self-generated	About 98%
[33]	Malware detection by visualizing the malware binary file as gray image	CNN	MatConvNet in MATLAB	Windows	Self-generated	—
[34]	Malware detection using API calls of Windows' executable files	DAE, RBM	Not stated	Windows	Comodo Cloud Security Center's dataset	Around 98%
[35]	Malware detection based on API calls sequence and statistical features	LSTM, RNN	TensorFlow	Windows	Self-generated	95.7%
[36]	Identifying executable files as malware or benign using static and dynamic analysis and categorizing the malware to the corresponding family	CNN, LSTM	TensorFlow, Keras	Windows	Malimg, EMBER, self-generated	98.8%
[37]	Hybrid image-based technique for malware detection by converting malware binaries to gray images	CNN, LSTM	TensorFlow, Keras	Windows	BIG 2015	96–97%
[38]	Malware detection by extracting API call sequences of malware using dynamic analysis and generating feature images	CNN	Not stated	Windows	VirusShare dataset	About 99%
[39]	Predicting malicious behavior of executable program based on small amount of behavioral data within the first few seconds of execution	RNN	Keras, scikit-learn	Windows	Self-generated	96%
[40]	DLMD: malware detection technique based on static features using byte and ASM files	CNN	PyTorch	Windows	BIG 2015	97.5%
[41]	Malware detection extracting control flow graph of the sample by lazy binding and transforming it into an image	CNN	Not stated	Windows	MALICIA, VirusShare, VXHeaven	92%–97.7%
[42]	Deep learning system with two hidden layers for malware detection using dependency of malware sequence and avoiding back-propagation	TELM	Not stated	Windows	Kaggle, VXHeaven	Above 99%
[43]	Malware classification by transforming malware binary file to grayscale images	CNN, LSTM	TensorFlow, Keras	Windows	BIG 2015	98.2%
[44]	Zero-day malware detection by generating fake malware and learning to distinguish it from the real malware	DAE, DCGAN	Keras	Windows	Kaggle	About 99%
[45]	Malware detection by visualizing the malware as grayscale image	CNN	TensorFlow	Windows	Malimg, Microsoft dataset	99.97%

Table 2: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Platform	Dataset used	Accuracy/F_1 score
[46]	Detecting threats in the cloud-assisted Internet of things by extracting API calls data from malware	DBN	Not stated	Windows	VXHeaven	Up to 99.78%
[47]	Malware classification by visualizing the malware as grayscale image	CNN	Not stated	Windows	Malimg, BIG 2015	97.5%
[48]	Malware variants detection by visualizing malware samples as grayscale images	CNN	Caffe NN framework	Windows	Dataset by Vision Research Lab	94.5%
[49]	Malware detection by converting malware executable to grayscale image and using NSGA-II algorithm to deal with data imbalance	CNN	TensorFlow	Windows	Dataset by Vision Research Lab	97.6%
[50]	Malware detection by visualizing the malware sample as a grayscale image	Deep transfer learning	Not stated	Windows	Not stated	99.25%
[51]	Malware detection by using static analysis to extract features of the malware samples	LSTM	Keras, TensorFlow	Windows	Self-generated dataset named MC-dataset-multiclass	90.63%
[52]	Malware detection by visualizing the malware sample as a grayscale image	CNN	TensorFlow	Windows	Malimg	80.5%
[53]	Malware detection by extracting features, like file activity, registry activity, service activity, processes, runtime DLLs and network activities, etc., and applying big data analytics techniques	Not stated	Keras	Windows	Self-generated	97%
[54]	Malware classification by converting malware binaries to Markov images	CNN	Keras, TensorFlow	Windows	Microsoft dataset, Drebin dataset	97.3% for Drebin, 99.3% for Microsoft
[55]	A comparative study of CNN and ELM-based detection systems using malware represented as grayscale images	CNN	Keras	Windows	Malimg	96.3% for CNN, 97.7% for ELM
[56]	Metamorphic malware detection using API calls made on the operating system	LSTM	Keras	Windows	Self-generated API sequence dataset	Up to 98.5%
[57]	Malware detection by extracting features of PE files, including import functions feature, general information feature, and bytes entropy feature	Not stated	Not stated	Windows	Self-generated	AUC up to 0.989
[58]	Cryptomining malware detection by static and dynamic analysis of the op-code sequences of PE files	CNN, LSTM, ATT-LSTM	Not stated	Windows	Self-generated	97% on average
[59]	Malware classification using malware samples represented as grayscale images	CNN	Keras, TensorFlow	Windows	Malimg	99.72%
[60]	Malware classification by extracting features including API calls, sequence of assembly language instructions, and malware's binary contents	CNN	TensorFlow	Windows	Kaggle	99.7%
[61]	Image-based malware classification system using an ensemble of CNN	CNN	TensorFlow, Keras, scikit-learn	Windows	Malimg	99.5%
[62]	Malware detection by black-and-white embedding of malware images rather than grayscale to avoid bit loss in byte	CNN	Keras, TensorFlow	Windows	KISA dataset	92.8%
[63]	Malware classification by generating a low-dimensional vector and using op-codes and API function calls to train model	Bi-LSTM	Not stated	Windows	Microsoft dataset	96.8%
[64]	Malware detection by extracting the API calls sequence and generating the API pixel vector and finally visualizing the malware	CNN	Not stated	Windows	Self-generated	94.7%

Table 3: Summary of the metadata extracted from the literature on android-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[65]	Malware detection using neural networks and k-means clustering	Not stated	No	No	Static analysis	Yes	No	Not stated	Android	Self-generated	88.0%
[66]	Malware detection based on API method calls sequence mining	CNN	No	No	Static analysis	Yes	No	TensorFlow	Android	Malgenome, Drebin, MalDozer	About 99%
[67]	Malware detection by analyzing the permission wanted by app	Deep eigenspace learning	No	No	Static analysis	Yes	No	Not stated	Android	Self-generated	Not stated
[68]	Malware detection by extracting and analyzing several features	Multimodal neural networks	No	No	Static analysis	Yes	No	Keras, TensorFlow, scikit-learn	Android	VirusShare, malgenome	94 – 98%
[69]	Dynamic malware detection system based on CPU, memory, and battery usage	LSTM RNN, encoder-decoder	No	No	Static analysis	Yes	No	Not stated	Android	M0Droid	About 80%
[70]	Malware detection by associating the features from static analysis with the features from dynamic analysis	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Self-generated, malgenome	96.76%
[71]	Malware detection using several static and dynamic features	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Self-generated	96.5%
[72]	Malware detection by using the importance of words from the apk file of applications	CNN	No	No	Static + renaming variables and prioritizing	Yes	No	Not stated	Android	Self-generated	92.67%
[73]	Malware detection by extracting several features for model training	CNN	No	No	Static analysis for static features	Yes	No	Keras	Android	Self-generated	99.25%
[74]	Malware detection using seven different features of android applications	DAE, CNN	No	No	Static analysis for 7 categories of static features	Yes	No	Keras, TensorFlow, scikit-learn	Android	Self-generated	99.82%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[75]	ITMF, (image texture median filter) for analyzing and detecting malware on Derbin dataset	DBN	No	Potential of dynamic activity of malware	Static analysis	Yes	No	Keras, TensorFlow, scikit-learn	Android	Drebin	95.43%
[76]	Malware detection using static analysis, dynamic analysis, and system calls	DBN	No	No	Hybrid analysis	Yes	No	Not stated	Android	Not stated	99.1%
[77]	Malware detection by extracting the API calls graph of applications and generating graph embedding	CNN, RNN	No	No	Pseudodynamic analysis	Yes	No	Keras, TensorFlow	Android	AMD dataset, AndroZoo, Drebin, ISCX	98.86%
[78]	Malware detection by examining all execution paths and detecting malicious and benign paths	LSTM RNN	No	No	Pseudodynamic analysis	Yes	No	TensorFlow, Keras, scikit-learn	Android	AndroZoo,	91.42%
[79]	Malware detection using features extracted from manifest file and through static analysis and various deep learning methods	CNN, DBN, LSTM, DAE	No	No	Static analysis	Yes	No	TensorFlow, Keras, theano	Android	Drebin, VirusShare	Up to 93.6%
[80]	Malware detection by extracting features through dynamic analysis and generating Markov chains	RNN, CNN, LSTM	No	No	Dynamic analysis	Yes	No	TensorFlow, Keras, scikit-learn	Android	Drebin	Around 81%
[81]	Malware detection by extracting texture fingerprint features and mapping malicious code to grayscale image	DBN	No	No	Static analysis	Yes	No	Theano, GDBN, TensorFlow, Keras, scikit-learn	Android	Drebin	95.9%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/ dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/ resilience against evolution? (yes/ no)	Library/ framework used	Platform	Dataset used	Accuracy/ F_1 score
[82]	Malware detection by using various features, including hardware components, permissions, filtered intents, API calls, and network addresses	LSTM, CNN	No	No	Static analysis	Yes	No	Not stated	Android	Drebin	Up to 98.53%
[83]	Malware detection using LASSO feature shrinkage and selection technique and deep belief networks	DBN	No	No	Static analysis	Yes	No	Not stated	Android	DroidWare	85.22%
[84]	Malware detection by generating API images from the sequence of API calls of applications	DAE	No	No	Dynamic analysis	Yes	No	Not stated	Android	Malgenome, contagio minidump	98%
[85]	Malware detection by using features like permissions and API calls and generating a feature vector	DBN	No	No	Static analysis	Yes	No	TensorFlow	Android	Drebin, VirusTotal, contagio, self-generated	99.04%
[86]	Malware detection by extracting byte code from the.dex file of the android package	DAE, DBN, RNN, LSTM, BiLSTM,	No	No	Static analysis	Yes	No	Keras, theano	Android	Drebin, AMD, VirusShare	Up to 99.9%
[87]	Malware detection by extracting features like API calls, events and permissions by dynamic analysis	Not stated	No	No	Dynamic analysis	Yes	No	Not stated	Android	Self-generated	98.5%
[88]	Malware detection by converting static features like permissions API calls and components into a set of semantic features	GCN	No	No	Static analysis	Yes	No	Not stated	Android	Drebin, AMD, lab-built, AndroZoo, PRAGuard	Up to 99.7%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[89]	Hybrid deep learning for android malware detection using various static and dynamic feature of the application	DBN	No	No	Hybrid analysis	Yes	No	TensorFlow, Keras	Android	Self-generated	96.8%
[90]	Malware detection by using dataset comprising of intent features and permission features extracted from benign and malicious applications	Not stated	No	No	Hybrid analysis	Yes	No	Not stated	Android	Omnidroid	91%
[91]	Malware detection by converting the application binary to gray-scale image	Not stated	No	No	Static analysis	Yes	No	Not stated	Android, iOS	AMD, self-generated	96.6% for Android, 95.8% for iOS
[92]	Malware detection by extracting byte code of the application and generating embedding	LSTM	No	No	Hybrid analysis	Yes	No	Not stated	Android, IoT	Self-generated	98% for Android, 99% for IoT malware
[93]	Malware detection by extracting 11 static behavioral features and transforming them to a multidimensional vector	DBN	No	No	Static analysis	Yes	No	TensorFlow	Android	Self-generated, Drebin, etc.	Up to 99.5%
[94]	Malware detection by extracting API sequence and the methods from the DEX file of the application and generating the hot vector of the API sequence	Bi-LSTM	No	No	Static analysis	Yes	No	Not stated	Android	AMD	97.2%

Table 3: Continued.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Automatic DL algorithm selection (yes/no)	Ability of identifying new malware? (yes/no)	Features engineering method (static/dynamic/hybrid analysis)	DL model needs updating? (yes/no)	Sustainability/resilience against evolution? (yes/no)	Library/framework used	Platform	Dataset used	Accuracy/F_1 score
[95]	Malware detection in the IoT devices by reading the DEX file of the application as an unsigned vector and converting it to a fixed size by image resampling technique	CNN, RNN, GRU, LSTM, Bi-LSTM	No	No	Static analysis	Yes	No	Keras	Android IoT devices	Self-generated	Up to 95.8%
[96]	Malware detection by generating the function call graph from the DEX file of the application and the op-code-level FCG features	LSTM	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	Self-generated	97%
[97]	Malware detection by extracting and vectorizing the manifest features and API calls from the binary file of the app	CNN, GRU, LSTM	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	Drebin, genome, contagio, pwnzen, VirusShare	96.8%
[98]	Malware detection by extracting features like permissions, system events, APIs and data flow from the manifest, DEX and layout xml files	MLP	No	No	Static analysis	Yes	No	TensorFlow	Android	Self-generated	94.9%
[99]	Malware detection by extracting static features (permissions) from the manifest file and then generating feature vector	CNN, DAE	No	No	Static analysis	Yes	No	Keras, TensorFlow	Android	CIC and Mal2017, self-generated	98.2%

Table 4: Summary of the metadata extracted from the literature on IoT-based malware detection.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset used	Accuracy/F_1 score
[100]	IoT and IoBT malware detection through op-code analysis	CNN	Not stated	IoT, IoBT	Self-generated	99.68%
[101]	Behavior-based deep learning framework for detecting malware in IoT environment	SAE	Keras	IoT	Self-generated	About 98.6%
[102]	Detecting pirated software and security threats in internet of things environment	CNN	TensorFlow, Keras	IoT	Not stated	96% for piracy detection, 97.46% for malware detection
[103]	Malware detection in the Internet of things environment by decompiling and extracting op-codes from application samples	LSTM	TensorFlow, Keras, scikit-learn	IoT	Self-generated	98.18%
[104]	Attack detection in industrial Internet of things environment	DAE, DFFNN	Not stated	IoT	NSL-KDD, UNSW-NB15	Up to 98.6%
[105]	Classification of malicious applications in the Internet of things environment by using graph embedding	CNN	Not stated	IoT	Self-generated	88.5%
[106]	Malware detection in industrial IoT devices by extracting DEX file and converting to java class file and extracting the bytecode from the class file	CNN	TensorFlow	IoT	Leopard mobile dataset	98.7%

which are connected by the logical operators "OR" and "AND." For example, different studies may use any of "Convolutional Neural Networks" and "CNN" in their title, so we need to combine these by the "OR" operator. Similarly, we need to have both "Deep Learning" and "Malware Detection" discussed in the abstract or the contents of the paper, so we need to combine these keywords by the "AND" operator. We included different keywords that may appear in the relevant studies, including Deep Learning, Deep Learning algorithms, and malware detection. The final search query we formulated is as follows.

("Deep Learning" OR "Convolutional neural network" OR "Deep belief network" OR "recurrent neural network" OR "CNN" OR "RNN" OR "DBN" OR "LSTM") AND ("malware") AND ("detection" OR "detect" OR "identification" OR "identify" OR "classification")

3.1.3. Searched Libraries.

For searching the relevant studies, we select five popular libraries, as follows.

(i) Google Scholar: https://scholar.google.com/ (accessed 02-07-2022)

(ii) Science Direct: https://www.sciencedirect.com/ (accessed 02-07-2022)

(iii) IEEE Explore: https://ieeexplore.ieee.org/(accessed 03-07-2022)

(iv) ACM: https://dl.acm.org/(accessed 03-01-2021)

(v) Springer Link: https://link.springer.com/(accessed 03-07-2022)

The selected libraries were searched using the formulated search query, filtering the search results to include only the relevant papers published during 2015–2022. The search query returned a total of 935 publications. The detailed search results for each library are shown in Table 7.

3.1.4. Inclusion/Exclusion Criteria.

Not all of the returned results could be relevant to the domain of the study. We need to further refine the results to get a final list of the most relevant publications. The inclusion and exclusion rules we applied are summarized in Table 8.

Our search query returned a total of 935 papers on all five libraries. Firstly, we excluded the papers that did not appear relevant based on the title or abstract, the survey papers, book chapters, and the gray literature. Table 7 summarizes the results after applying inclusion/exclusion criteria.

We downloaded the references of the 290 papers and created an endnote library. There, we further refined the results by removing duplicates, non-journal papers, papers written in a language other than English, and irrelevant papers based on full text. Table 9 summarizes the finally selected papers for critical analysis.

Figure 1 presents a summary of the research methodology used for research articles retrieval.

3.2. Metalevel Critical Analysis of the Literature.

Once the relevant literature is retrieved and filtered, metalevel analysis is performed from different perspectives. In the first phase, to let the readers know of the importance of the proposed study, statistical analysis of the research work done so far is performed. In the second and third phase, respectively, Windows and Mobile platforms-wise metalevel analysis of the malware detection research is performed. These analyses are summarized in the subsequent sections.

Table 5: Summary of the metalevel analysis of the literature on malware detection in platforms other than Windows, Android, and IoT or multiple platforms.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset Used	Accuracy/$F1$ score
[107]	Malware dynamic behavior classification and family clustering algorithm	CNN, GAN	Not stated	Windows, android	DataCon, GreekPwn	Not stated
[108]	Detecting domain generation algorithms (DGAs) and automatically labelling domain names in real traffic	LSTM RNN	Keras, scikit-learn	Not stated	ALexaBamb, Retro	About 98%
[109]	Deep learning-based intrusion detection system for detecting cyber-attacks	Not stated	TensorFlow, Keras, scikit-learn	Internet	KDDCup99, NSL KDD, UNSW-NB15, WSN-DS, CICIDS 2017, Kyoto	85 – 99%
[110]	An intrusion detection system to protect in-vehicle network, the controller area network (CAN) bus	CNN	Not stated	—	Self-generated	99%
[111]	Source-based distributed denial-of-service defense system in fog and cloud computing systems	LSTM	Keras, TensorFlow	Cloud computing	Hogzilla	98.88%
[112]	A hybrid deep learning-based system for detecting botnet	CNN, RNN	Keras, TensorFlow, scikit-learn	Internet	CTU-13, ISOT	99.3% (CUT-13) 99.5% (ISOT)
[113]	Using robust software modeling tool (RSMT) to monitor and characterize the behavior of web based applications	SAE	Keras, TensorFlow, scikit-learn	Internet	Not stated	About 92%.
[114]	Deep multilayer perceptron and RNN-based deep learning system for detecting cloud-based intrusion	RNN, LSTM	Keras, TensorFlow, Theano	Cloud computing	Not stated	86.9%
[115]	Malware detection in PDF files	CNN	Not stated	Multiple	Self-generated	Up to 98.92%
[116]	Ransomware detection and classification by extracting event sequences during a program execution	LSTM, CNN	Keras, TensorFlow	Not stated	Self-generated	99.6%
[117]	Malware detection in cloud platforms by extracting several features of each process, like CPU usage, memory usage, and disk usage	CNN	Not stated	Cloud IaaS	Self-generated	Up to 93%
[118]	Using ML and DL techniques to distinguish normal traffic from cryptomining traffic by extracting the data flow features	Fully connected CNN	Keras, TensorFlow, scikit-learn	Internet	Self-generated mining traffic	99.98%
[119]	Malware detection on various platforms, including Windows, Android, IoT, IoBT, and the Internet by vector embedding	LSTM	Not stated	Windows, Android, IoT, IoBT, Internet	VXHeaven, Drebin, Kaggle	94.1% on average

3.2.1. Statistical Analysis. To answer to RQ2 and know about the peak era of research for malware detection using deep learning methods, the most appropriate place for publishing the work and the most affected platform out of the Windows and Mobile platforms, the literature is statistically summarized and presented in the following sections.

(1) Year-Wise Distribution of the Papers. We observed that, in recent years, there are a lot of research articles published discussing malware and intrusion detection systems using deep learning. We have only one paper in the years 2015 and 2016 while the number increases as we go upward. This shows that malware detection using deep learning algorithms is an active area of research in computer science. Figure 2 shows the year-wise distribution of the research papers published.

(2) Platform-Wise Distribution of the Papers (RQ1). As malware is not limited to a specific platform, different researchers have focused on different platforms for malware

Table 6: Summary of the metalevel analysis of the literature on malware detection during the recent years.

Ref.	Description: method and features used to train and evaluate model	DL algorithm used	Library/ framework used	Targeted platform	Dataset used	Accuracy/F_1 score
[120]	Visualizing malware binaries as two-dimensional images and feeding to classifier that uses reweighted class-balanced loss function	Densely connected CNN with ReLU	Keras	Windows	Malimg, BIG 2015, MaleVis	98.46%
[121]	Two-stage hybrid malware detection by extracting op-code by static analysis and then performing dynamic analysis to classify benign files	Bi-LSTM, CNN	Not stated	IoT	KISA 2019	Up to 95%
[122]	Malware detection by representing the application as image, extracting the dex file, and grouping the sequence of bytes into grayscale pixel	CNN	CUDA, TensorFlow	Android	Argus Cyber Security Lab	97%
[123]	Malware detection by using text classification method, using the text sequence of APPs analysis and exploring information	CNN	Keras	Android	Various datasets	96.6%
[124]	Malware detection using dynamic analysis by generating dynamic analysis logs for an APK and transforming the features into a feature vector	CNN with leaky ReLU	Not stated	Android	Self-generated	98%
[64]	Malware detection by visualizing malware as RGB color images using both static and dynamic as well as hybrid analysis	CNN (VGG16)	Not stated	Windows	Dataset by VirusSign	94.7%
[125]	Detection of Java bytecode malware using static analysis of the Java program and extracting interprocedural control flow graph from bytecode file	CNN	Not stated	Platforms capable of running Java programs	Self-generated	98.4%
[126]	Analysis of behavior of malicious programs based on API call graphs. The detection is based on analyzed patterns of the API calls	CNN (used only for discovering common features)	Not stated	Android	Apps from playstore and VirusShare	93.2%
[127]	Classification and detection of malware using executable and linkable format (ELF) binary file, making use of static, dynamic, and hybrid analysis	Bi-GRU-CNN	Keras, TensorFlow, scikit-learn	IoT	Collected from various sources	98% (detect) 100% (classify)
[128]	Malware classification by converting the bytecode of methods of the malware into grayscale feature image and analyzing its feasibility based on reconstruction error of AE	AE based on CNN	TensorFlow	Android	Apps from playstore and VirusShare	96.2%
[129]	Distributed deep learning-based model for malware detection using both static and dynamic analysis	CNN-BiLSTM	Not stated	Windows	Apps from various sources	97%
[130]	Using DL and model-checking to detect malware by converting source code to format of the model-checker, using both static and dynamic analysis	CNN	PyTorch	IoT	Not stated	95%
[131]	Malware detection using static analysis, emphasizing on features extraction from PE files	Not stated	Keras	Windows	EMBER	97.5%

detection. To answer the RQ1 (which platforms are affected the most?), analytics have been calculated from the selected papers and the insights are summarized in Figure 3, stating that android and windows are the most widely used platforms on mobile devices and personal computers, respectively, and hence affected the most. Most of the researchers

Table 7: Summary of the literature extracted from the five scholarly libraries and the process of filtration.

Library	Library URL	Returned papers	SLR/book/gray literature	Irrelevant papers	Total
Google Scholar	https://www.scholar.google.com/	333	74	70	180
Science Direct	https://www.sciencedirect.com/	306	114	146	46
IEEE Explore	https://ieeexplore.ieee.org/	144	12	85	47
ACM	https://dl.acm.org/	9	2	1	6
Springer	https://link.springer.com/	143	45	87	11
	Total	**935**	**247**	**398**	**290**

Table 8: The inclusion/exclusion criteria for the deep learning methods.

	Inclusion criteria
1	The paper must discuss a deep learning-based system for malware or intrusion detection.
2	The paper must be published in a scholarly journal or be a preprint.
3	The paper is published from January 2015 till 2022.
	Exclusion criteria
1	Papers focusing on economic, business, or legal impacts of malware detection and intrusion detection systems
2	Gray literature, such as blogs or reports
3	Papers written in a language other than English
4	Review papers
5	Duplicate papers
6	Papers not published in any scholarly journal, such as conference preceding
7	The studies that do not focus on deep learning for malware detection

Table 9: Summary of the selected literature for review.

Total papers	Nonjournal papers	Duplicates	Non-English	Full text exclusion	Final total
290	101	64	7	11	107

focused on these two platforms. However, the Internet of things environment and web-based malware received less attention despite the fact that malware developers target these platforms frequently. Web-based malicious programs are especially a big threat to the Internet and data security. Figure 3 shows the platform-wise distribution of the papers. Based on the findings of RQ1, Sections 4 and 5 have been given more attention, and hence in-depth critical analysis is performed.

(3) Journal-Wise Distribution of the Papers. We retrieved relevant publications from a large number of journals using the selected libraries. The number of papers from individual journals varies from 1 to 13, including arXiv preprints and journals from other publishers like Hindawi, IEEE, ACM, and more. Table 10 contains the journal-wise distribution data of the papers.

4. Windows Malware Detection

In the Windows platform, researchers have extensively worked on the subject matter to protect personal computers (PC) against cyber-attacks. In this section, we analyze the research work that focuses on Windows malware detection and present a summary in Table 2.

Ni et al. [26] proposed an algorithm "Malware Classification using SimHash and CNN" (MCSC) based on convolutional neural networks. They disassemble the code of the malware and convert it to gray images to identify its family. They apply locality-sensitive hashing (LSH) to convert similar malware code into similar hash values. These hash values are converted to gray images to train neural networks. They claim about 98% accuracy.

Zhao et al. [27] proposed MalDeep, a deep learning-based malware detection system that uses the binary file of the malware. They convert the binary file to gray image and use convolutional neural networks to classify the malware. The strength of their system has a high accuracy of over 99%.

Zhang et al. [28] proposed a deep learning system that uses sensitive system calls for malware detection. The application is monitored in Cuckoo sandbox to retrieve system calls data and train the neural networks. Their system achieves an accuracy of over 95%.

Zhang et al. [29] proposed a convolutional neural network-based malware detection system that includes unpacking the application to retrieve its op-codes and API calls, generating structured data to represent each binary and obtaining PCA-initialized op-code bi-gram matrix and PCA-initialized API frequency vector which are then fed to CNN and BPNN to train a feature embedding model. Their proposed system achieved an accuracy of 95%.

Zhong and Gu [30] proposed a multilevel deep learning system that selects important features from the dynamic and static features set, partitioned the set into many one-level clusters using K-means algorithm, generated cluster subtrees

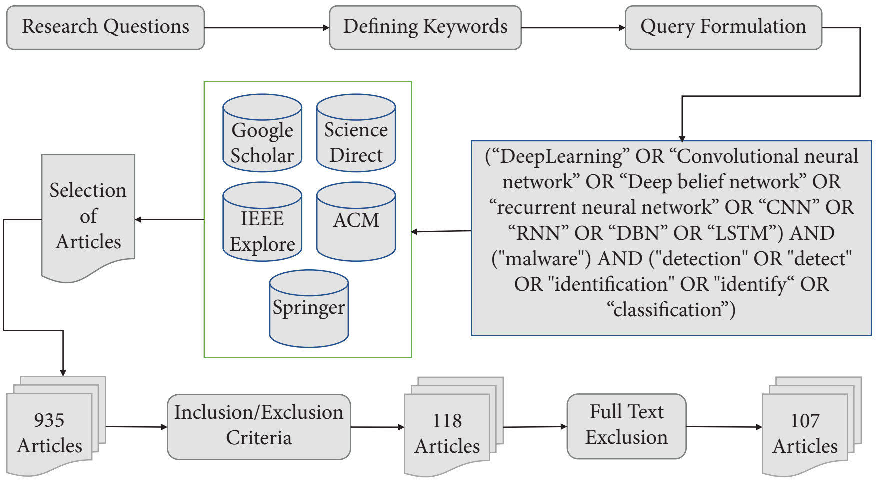
Research Questions
Defining Keywords
Query Formulation
("DeepLearning" OR "Convolutional neural network" OR "Deep belief network" OR "recurrent neural network" OR "CNN" OR "RNN" OR "DBN" OR "LSTM") AND ("malware") AND ("detection" OR "detect" OR "identification" OR "identify" OR "classification")
Google Scholar
Science Direct
IEEE Explore
ACM
Springer
Selection of Articles
935 Articles
Inclusion/Exclusion Criteria
118 Articles
Full Text Exclusion
107 Articles

Figure 1: Research articles retrieval methodology of the systematic literature review.

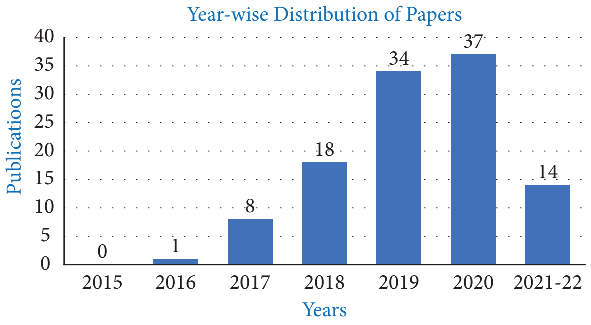

Figure 2: Year-wise distribution of the research work on the subject area malware detection using deep learning.

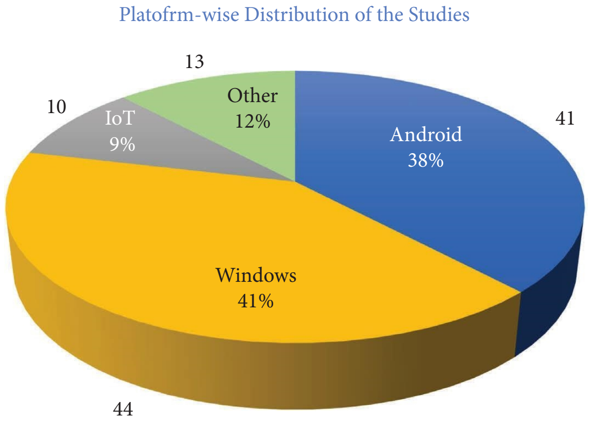

Figure 3: Platform-wise distribution of the research papers on the subject area and malware detection using deep learning.

and combined decision values of deep learning models in the tree for classification of the application as malware or benign.

Zhang et al. [31] proposed a ransomware detection system that transforms the op-code data and ransomware family label to numeric tensors to be used as input to the neural network. They use self-attention powered convolutional neural networks (SA-CNN) in their proposed method. The weakness of their system is the comparatively lower accuracy of about 90%.

Yuxin and Siyi [32] developed a deep belief network-based system that extracts the op-code of malware and used the neural network to detect it. Their system consists of a PE parser that transforms the PE file to op-code sequences, a feature extractor that selects n-grams that have strong classification power and to represent a PE file as n-grams vector, and a malware detection module. Their proposed model achieved about 98% accuracy.

Yue [33] proposes a weighted softmax loss (combination of softmax regression and entropy loss) for deep convolutional networks on malware image classification. It is claimed that this would resolve the issues that are caused by the imbalance of malware families.

Ye et al. [34] proposed a malware detection system that performs directly on Windows PE file. Their proposed system consists of a feature extractor that decompresses the file and parses PE code to extract the API calls from the file. Then they use unsupervised heterogeneous, autoencoder, and RBM-based deep learning model for malware detection.

Xiaofeng et al. [35] combined machine learning and deep learning and proposed an LSTM RNN-based malware detection system that uses API calls sequence and statistical features of malware. They run the malware in a sandbox to get the API calls and use random forest model to classify the system call sequence, which is then processed to get a feature vector that is used as input to the deep learning model for classification.

Vinayakumar et al. [36] proposed a distributed system ScalMalNet, which collects malware samples from different sources and processes the malware samples in real time or on demand basis in a distributed manner. They proposed an image processing framework for malware detection and classification using static and dynamic analysis. They applied various shallow learning and deep learning techniques for malware detection and experimentally shown that deep learning-based malware detection systems work much better than traditional ML-based systems.

Table 10: Journal-wise distribution of the papers on malware detection using deep learning.

S. no.	Journal	No. of papers	References
1	ArXiv preprints	11	[33, 40, 50, 52, 59, 66, 69, 94, 105, 107, 117]
2	Academia	1	[65]
3	ACM SIGCOMM computer communication review	1	[71]
4	Ad hoc networks, 2020—Elsevier	2	[95, 106]
5	Alexandria Engineering Journal	1	[76]
6	Cluster Computing	1	[79]
7	Computer Communications	1	[63]
8	Computers and Security	13	[26, 29, 39, 41, 42, 54, 60, 61, 87, 88, 122, 125, 127]
9	Cybersecurity	1	[28]
10	Digital Investigation	1	[43]
11	Engineering Applications of Artificial Intelligence	1	[80]
12	Expert Systems with Applications	1	[30]
13	Future Generation Computer Systems	4	[31, 86, 103, 116]
14	IEEE Access	8	[36, 99, 102, 108, 109, 114, 118, 128]
15	IEEE Transactions on Industrial Informatics	1	[48]
16	IEEE Transactions on Emerging Topics in Computational Intelligence	1	[119]
17	IEEE Transactions on Network Science and Engineering	1	[98]
18	IEEE Transactions on Sustainable Computing	1	[100]
19	IEEE Transactions on Information Forensics and Security	2	[68, 97]
20	IEICE Transactions on Information and Systems	1	[62]
21	IET Information Security	1	[38]
22	Information Sciences	1	[44]
23	International Journal of Advance Soft Computing	1	[82]
24	International Journal of Digital Crimes and Forensics	1	[45]
25	International Journal of Emerging Technology and Computer Science	1	[67]
26	International Journal of Performability Engineering	1	[73]
27	Journal of Ambient Intelligence and Humanized Computing	1	[74]
28	Journal of computer Virology and Hacking Techniques	2	[47, 91]
29	Journal of Computers	1	[81]
30	Journal of Grid Computing	1	[58]
31	Journal of Information Security and Applications	2	[37, 104]
32	Journal of Internet Services and Applications	1	[113]
33	Journal of King Saud University—Computer and Information Sciences	1	[111]
34	Journal of Parallel and Distributed Computing	3	[46, 49, 84]
35	Journal of Signal Processing Systems, 2021—Springer	1	[64]
36	Knowledge and Information Systems	1	[34]
37	KSII Transactions on Internet and Information Systems	2	[75, 85]
38	Mathematical Problems in Engineering	1	[101]
39	Microelectronics Reliability	1	[72]
40	Neural Computing and Applications	3	[32, 92, 112]
41	Neurocomputing	1	[78]
42	PeerJ Computer Science, 2020	1	[56]
43	Pertanika Journal of Science and Technology	1	[83]
44	PloS One, 2020	1	[57]
45	Procedia Computer Science	2	[35, 51]
46	Security and Communication Networks	3	[27, 89, 115]
47	Sensors	2	[96, 130]
48	Soft Computing	2	[77, 93]
49	TechRxiv	1	[90]
50	Tsinghua Science and Technology	1	[70]
51	Vehicular Communications	1	[110]
52	Entropy	1	[120]
53	Human-Centric Computing and Information Sciences	1	[121]
54	Applied Soft Computing	1	[123]
55	Journal of Internet Services and Information Security	1	[124]
56	Signal Processing Systems	1	[64]

Table 10: Continued.

S. no.	Journal	No. of papers	References
57	International Journal of Information Security	1	[126]
58	Cybernetics and Systems	1	[129]
59	International Journal of Computer Network & Information Security	1	[131]

Venkatraman et al. [37] investigated the use of image-based techniques for detecting suspicious behavior and proposed their own image-based malware detection technique by transforming the binary code of malware samples to grayscale images. They use both CNN and LSTM and develop a self-learning system that is capable of detecting the known malware as well as unknown malware.

Tang and Qian [38] proposed a CNN-based malware classification system that extracts API calls sequence of the application using dynamic analysis and generating feature image. They run the malware sample in a sandbox and extract the API call sequences and generate feature images using color mapping rules, category of the API and the number of times the category occurs in unit time. These images are used to train the convolutional neural network for detecting unknown malware. The strength of their system is the claimed accuracy of over 99% in most cases.

Rhode et al. [39] proposed an early detection system based on a recurrent neural network that works within the first five seconds of execution of a program and detects malicious behavior based on behavioral data. They generate machine activity data metrics based on the initial dynamic data and use them as feature input to the model. The features they used were CPU usage, packets sent and received, bytes sent and received, swap use, memory usage, number of current processes, and the maximum process ID assigned. They compared their RNN model with traditional machine learning algorithms and showed that deep learning performed much better, achieving an accuracy of 96%.

Rafique et al. [40] proposed a malware detection technique that uses the byte and ASM files to extract static features for classification. They use a convolutional neural network to extract features from the byte files, while for extracting features from ASM files, a wrapper-based technique is used. Then, they use feature space to train multilayer perceptron, which classifies the different malware categories of the BIG 2015 dataset.

Nguyen et al. [41] proposed a CNN-based deep learning system for malware detection that uses a modified form of control flow graph called lazy-binding CFG. They generate CFG from the binary code of the malware by using lazy-binding instead of early-binding for more precise results and convert the CFG to pixel image by transforming the CFG to adjacency matrix. By this way, variants of a specific malware are represented by closely similar objects, which are then inputted to the deep learning model for malware detection.

Namavar Jahromi et al. [42] proposed two-hidden-layer-based extreme learning machine (ELM) for malware detection. The system uses dependencies between malware features, like op-codes and API, calls to train the deep learning model. The extreme learning model has a different connection of input to the first hidden layer and is partially connected. They compared their proposed method with various deep learning methods and showed that their method achieved a better accuracy of above 99% in most cases.

Le et al. [43] proposed a malware classification system that is based on transforming the malware binaries to grayscale images. They use a convolutional neural network based deep learning model to classify malware, training their model using Microsoft Kaggle dataset. They developed three different models, one with CNN, another one with CNN and LSTM, and the third one with CNN and biLSTM achieving the highest accuracy of 98.2% with the third model.

Kim et al. [44] used transferred deep convolutional generative adversarial network (tDCGAN) to detect zero-day (a type of malware) by creating fake malware and feeding it to the modal to learn to distinguish a real malware.

Kalash et al. [45] proposed a CNN-based data-independent system for malware detection that uses the grayscale representation of the malware sample. The system reads the malware binary file in a vector of 8-bit integers and converts the binary value to the decimal equivalent and a new decimal vector representation is generated. Then, they represent this decimal vector as a two-dimensional matrix and transforms it to a grayscale image. The strength of their system is the high accuracy rate of 99.97%.

Huda et al. [46] proposed a deep belief networks-based system for detecting threats in the cloud-assisted Internet of things environment. To collect data, they execute the malware in a virtual sandbox, observe the change of states and collect any operations performed by it, and generate a report that includes the API calls and their parameters, which are used to prepare a frequency list of the APIs. Next, they train the deep belief networks-based model for malware detection.

Gibert et al. [47] proposed a convolutional neural networks-based system for malware classification that visualizes the malware as a grayscale image to extract features. They interpret each byte of the malware sample as a pixel and visualize the resulting two-dimensional array as a grayscale image. Then, they use a convolutional neural network-based system for extracting features from the images. Next, they use CNN with a softmax layer to classify the malware samples.

Cui et al. [48] proposed a CNN-based malware variants detection system. They first split the malware binary bit string into 8-bit substrings and consider each of the substrings as a pixel to visualize the image as a grayscale image. Then, they use a convolutional neural network based deep learning system that consists of an input layer, a convolutional layer, a subsampling layer, and several fully connected layers to classify malware.

Cui et al. [49] proposed a CNN-based approach for malware detection that uses grayscale images generated from the malware executable. They also split the malware binary bit string into 8-bit substrings each of which is considered as a pixel. This way they visualize the malware as a grayscale image. Then, they use a CNN-based deep learning framework that consists of two convolution layers, one pooling layer, and two dense layers to classify malware samples.

Chen [50] proposed a deep transfer learning based method for malware detection. The malware binary was mapped into an integer in the range 0 to 255 to generate pixels and visualize the malware. Next, deep transfer learning-based model was used to classify malware samples. This approach was compared with several shallow learning approaches, and it was shown that deep learning achieved much better results.

Andrade et al. [51] developed an LSTM-based system for detecting five different families of malware, including rootkit, virus, Trojan, worm, and backdoor. They rely only on static analysis (or code analysis) of the malware to extract features of the malware file. Their model consists of an input layer, an LSTM layer, a dropout layer, and a dense layer. A weakness of their system is the average accuracy of 90%.

Agarap [52] proposed an SVM and CNN-based system for detecting malware. First, the binary string of the malware sample was transformed to 8-bit vectors, which are further processed and transformed to a grayscale image. Both multilayer perceptron and CNN were used for experiments and achieved better results with MLP; however, a below-average accuracy rate of about 80% was achieved.

Other papers that focus on Windows-based malware detection include [53–64].

5. Android Malware Detection

Like Windows platform, in android platform, researchers have worked on malware and intrusion detection using deep learning. This section analyzes the research work performed over android platform, extracting meta-information highlighted in research questions (RQ7 and RQ8), in addition to the information considered in Windows-based malware detection literature. Summary of android-based malware detection techniques is shown in Table 3.

Devi [65] proposed a permission based android malware detection system. They extracted the manifest files and permissions from the android packages and generated feature vectors and trained their model using neural networks and *k*-means clustering algorithm. The weakness of this approach is comparatively lesser accuracy of 88%.

Karbab et al. [66] proposed MalDozer, an android malware detection system based on the API method call sequence of the applications. MalDozer disassembles the classes.dex file of an android package to extract API method calls and discretize them by replacing each API method by an identifier and generates the semantic vectors. Then, they train the neural networks to predict android malware. The strength of their system is the high F_1 score achieved on various datasets.

Khedkar et al. [67] also proposed a permission-based android malware detection system. The FAST algorithm they designed use graph clustering method to cluster the features of the application and construct a trained dataset to classify new malware.

Kim et al. [68] used multiple features for malware detection including API methods, op-code features, permission features, share library function op-code features, component features, and environment features to generate feature vectors for each feature. These vectors are then fed to the classification model to predict malware. The strength of their approach is the usage of multiple features unlike most of the others who used a single or two features.

Milosevic and Huang [69] proposed a deep learning-based malware prediction system that uses CPU, memory, and battery usage to predict malware. Their unsupervised method is based on encoder-decoder and LSTM networks, using different applications to retrieve data, like CPU and battery usage. The weakness of their system is comparatively lower F_1 score of about 80%.

Yuan et al. [70] developed an online android malware detection system. The proposed system extracts three features, required permissions, sensitive API calls, and dynamic behavior and then use deep belief networks to detect malware in an application. Their deep learning model has two phases, an unsupervised learning phase and a supervised back propagation phase.

Yuan et al. [71] proposed a deep learning-based method that includes extracting features like permissions, sensitive API calls, and dynamic behavior. They used more than 200 different features in their proposed framework and used deep belief networks for malware detection. Their claimed accuracy is over 96%.

Yen and Sun [72] proposed a system that use the importance of words in apk file for malware detection. They extract the classes from apk file and convert them to java files and find the importance value of each word in the code. Then, they generate images by using the words importance from code using Term Frequency-Inverse Document Frequency (TF-IDF), a text mining and information retrieval method. These images were used to train and text their CNN based model.

Xie et al. [73] proposed a CNN-based approach, which includes extracting seven different malware features: API calls, hardware features, filtered intent, requested permissions, used permission, and restricted API calls. The framework consists of Dataset Construction, which includes collecting samples, labelling and features extraction, and classification process in which feature vectors are transformed to matrices and the dataset is divided to training set and validation set. The strength of their system is the claimed accuracy of 99.25%.

Wang et al. [74] combined deep autoencoder with a modified model of convolutional neural network they called CNN-S and proposed an android malware detection system that uses seven different features of applications to train their model. The features include restricted API calls, suspicious API calls, permissions, requested permissions, hardware features, filtered intents, and code-related patterns. The strength of their system is the claimed high accuracy rate of 99.82%.

Luo et al. [75] proposed ITMF (image texture median filter) to analyze and detect android malware. Median filter is a filtering technique for removing or reducing noise from images and signals to improve processing and results. They obtain the malware binary file and convert it to a vector which is then transformed to grayscale image, which is then inputted to the ITMF. They extract features including API calls, used permissions, URL, and activity and train deep belief network for malware detection. They compared their model with shallow learning techniques and achieved better results with deep learning.

Saif et al. [76] proposed a deep belief network-based android malware detection system. They used both static analysis and dynamic analysis of android application and extracted features like manifest components, API calls, dynamic behavior of the application, and system calls and generated feature vector. They applied relief feature selection by using relief algorithm, which outputs another vector with the quality measurement of features. This vector is inputted to the deep neural network.

Pektaş and Acarman [77] proposed an android malware detection system that uses API calls graph. They build an API call graph for each execution path; the API call number is selected to generate graph embedding if it is equal to or greater than a threshold value and the graph embedding features are processed to be interpreted numerically. Then, the embedding vectors are inputted to the CNN-based deep learning model to classify malware.

Pektaş and Acarman [78] built an android malware detection system that examines all the execution paths and detect malware by using features extracted from instruction call graph. Their method consists of pseudodynamic analysis of the application in which call graphs and execution paths are extracted in terms of op-codes. Then, they construct a flow graph for each execution path and process the graphs to be interpreted numerically and to generate vectors. The vectors are then inputted to the LSTM RNN-based deep learning model to determine the probability of being benign or malware. They compared their approach with traditional machine learning approaches and showed that deep learning achieves better accuracy rate.

Nauman et al. [79] used different deep learning methods, including CNN, DBN, LSTM, and autoencoders on large-scale dataset for detecting android malware. They used the features from manifest file and those extracted through static analysis including requested permissions, components, filtered intents and restricted API calls, and so on as input to the deep learning model and evaluated the performance of different deep learning methods.

Martín et al. [80] proposed a deep learning-based system CANDYMAN, which classifies malware by combining dynamic analysis and Markov chains. They use DroidBox tool to run the application and extract dynamic behavior. The information gathered include network data, read/write operations, services, loaded classes, file, and SMS services and permissions, which is reported in a JSON file. In the next step, the data from the JSON file is represented in terms of Markov chains. Finally, the Markov chains are transformed into feature vectors that are then fed into deep learning networks for malware classification. They performed experiments using different machine learning and deep learning algorithms; however, they achieved a lower accuracy of around 81%.

Shiqi et al. [81] presented an attention-CNN-LSTM-based deep learning system for android malware detection. They use deep belief networks to extract texture fingerprint features and the malware activity embedding in vector space and then the malicious code is converted to grayscale image. The malware texture fingerprint features and the activity embedding in vector space are fed to the attention-CNN-LSTM-based deep learning model for malware classification. They compared their model to traditional machine learning algorithms and showed that they achieved a better accuracy with deep learning.

Halim et al. [82] proposed an android malware detection system that uses Bag of Words (BOW) model to extract various features of the application, including hardware components, used permissions, requested permissions, application components, filtered intents, restricted API calls, suspicious API calls, and network addresses. They used two different deep learning models for malware detection, a CNN-LSTM model in which CNN is stacked over LSTM and an LSTM-CNN model in which LSTM is stacked on top of CNN. They achieved an accuracy of 96.76% with the CNN-LSTM model while 98.53% with LSTM-CNN model.

Elsersy and Anuar [83] proposed a deep belief network-based deep learning system for android malware detection. They used Lasso features shrinkage and selection technique, which is used for features selection by means of absolute regularization penalty and evaluated traditional machine learning technique (K-NN classifier) and deep learning technique (DBN) for malware detection. They achieved better accuracy with deep learning technique; however, the accuracy rate they achieved was below average, 85.22%.

D'Angelo et al. [84] generated sparse matrices from the sequence of the API calls to be used for malware detection. Their autoencoders based system represents the temporal behavior of the application by using the sequence of sparse matrices and extract features from the sparse matrices, which are then used to classify the application as malware or benign-ware.

Chen et al. [85] proposed an android malware detection system that uses features like permissions and sensitive API calls extracted from the APK file. They model the features as a document and generate k-dimensional word vectors using word2vec. Finally, they use deep belief networks based deep learning system for malware classification.

Amin et al. [86] proposed an android malware detection system based on various deep learning methods. They extract the .dex file (dalvik executable file) from the APK file and further use it to extract the byte code. This byte code is given as input to the deep learning model for training, feature engineering, and classification of the sample as malware or benign. They used different deep learning methods, including DAE, DBN, LSTM, BiLSTM, CNN, and RNN, and claimed to have achieved an accuracy of up to 99.9%.

Alzaylaee et al. [87] proposed a malware detection system for android platform that runs the application to extract its features. They use DynaLog, a platform that runs a large number of android applications in sequence to log and extract dynamic features, such as API calls, actions, events, and permissions. They extract 178 features and rank them using InfoGain to select the top 120 of them for experiments. Other research articles that focused on android malware detection include [88–99].

6. IoT/IoBT Malware Detection

A number of studies focused on malware detection in the Internet of Things and Internet of Battlefield (Military) Things (IoBT/IoMT) environments. These studies are analyzed and summarized in the following section.

Azmoodeh et al. [100] proposed a two-phase method for malware detection. They first generate the Op-code sequence graph by using the selected features and then use deep eigenspace learning to classify Internet-of-things and Internet of (battlefield) things (IoBT) malware. The strength of their system is the claimed accuracy of over 99%.

Xiao et al. [101] also combined machine learning and deep learning, combining DT, NB, SVM, and KNN with autoencoders. They proposed a behavior-based deep learning framework for malware detection in the Internet of Things environment. Their proposed model consists of IoT environment, which includes local computers and smart devices and a cloud platform module (CP module). CP provides storage space, constructs behavior graph, and transforms the API call graphs to binary vectors. These vectors are used as input to the stacked autoencoders-based deep learning model.

Ullah et al. [102] proposed a convolutional neural network-based system for detecting pirated software applications and files infected by malware in the IoT environment. Their system consists of a preprocessing module that transforms the malware binary file to grayscale image, a convolutional neural network to which the training images are inputted so that the classifier identifies the respective malware families using the images, a convolution layer that is used to extract meaningful features, and a pooling layer that is used to minimize the consequences of image distortion and increase CNN functioning. They claimed to have a better accuracy rate of 96% (piracy detection) and 97.46% (malware detection) as compared to other traditional machine learning techniques.

Haddadpajouh et al. [103] proposed an LSTM-based system for detecting malware in the Internet of things environment. They first collected samples of malware and benignware and decompiled them using object-dump tool. They used a Linux hash script to extract op-codes from the samples and used text mining techniques to generate features from the op-codes. Next, they used LSTM network with two hidden networks for malware detection. They compared their approach with several traditional learning methods and showed that they achieved much better accuracy rate with deep learning.

Al-Hawawreh et al. [104] proposed a deep autoencoder and deep feed-forward neural network-based system for detecting malicious activities in IoT environment. The deep autoencoder-based model learns to use normal network observations and creates initialization parameters and learns the representation of normal behavior. The parameters created at this stage are used as input to the DFFNN-based model for detecting new attacks.

Abusnaina et al. [105] used graph embedding to classify malicious programs in the IoT environment.

Naeem et al. [106] extracted bytecode from the java class file of the malicious software and used this bytecode for detecting malware in the Internet of things environment.

7. Other Platforms

Some of the studies focused on malware in the cloud environment, web applications or did not even mention what operating system or platform they targeted, or targeted multiple platforms at a time. These studies are analyzed in the following section and summarized in Table 5.

Lu et al. [107] constructed their own deep neural network for malware classification, which they named MalDeepNet. They also used several features, like API features, PID features, RET features, EXINFO features, and reboot features, and implemented TB-MalNet (Text-based MalNet) and IB-MalNet (Image-based MalNet) for malware prediction.

Yu et al. [108] investigated the use of deep learning algorithms for Domain Generation Algorithms (DGAs) and developed an LSTM-based deep learning model for DGA detection trained with weakly labelled data obtained from real traffic. They achieved an accuracy of around 98% on different datasets.

Vinayakumar et al. [109] proposed a deep learning-based distributed system for detecting cyber-attacks. The proposed system was built using big data processing frameworks Apache Hadoop YARN and Apache Spark. They used various shallow learning algorithms, like NB, RF, SVM, LR, KNN, and so on, and deep neural networks with different layers and evaluated their performance on different datasets and experimentally shown that deep neural networks perform better than traditional machine learning algorithms.

Song et al. [110] proposed an intrusion detection system for the controller area network in vehicles to protect the CAN bus of the vehicle. Their CNN-based system retrieves CAN IDs from the logged CAN and assembles data frames, each consisting of 29 sequential IDs. The data frames are then processed and classified as attack or nonattack. They also compared their proposed system with traditional machine learning techniques and experimentally showed that DL performed better. The strength of their system is the claimed F_1 score of above 99%.

Priyadarshini and Barik [111] proposed a DDoS defense system that is capable of detecting and mitigating denial of service attacks in fog and cloud computing environment. They use Hogzilla dataset to train their LSTM-based deep learning model for DDoS defense and achieve a high accuracy rate of 98.88%.

Pektaş and Acarman [112] presented a hybrid, RNN- and CNN-based, deep learning system for detecting botnet. They extract flow features from the network traffic and transform them to multidimensional feature vector. The feature vector is inputted to the classification model for detecting whether the element is normal or botnet. They use the connection patterns created due to the data transmission between botnets and servers and split network traffic between endpoints and represent them as graph to extract features. The strength of their system is the high accuracy rate of nearly 99.4% on average.

Pan et al. [113] proposed a deep learning-based system for detecting attack in the web traffic by analyzing web applications. Their system uses robust software modeling tool (RSMT), which is a tool that targets languages that run on JVM, and extracts traces of program execution and generates models of behavior of the running application. RSMT captures features that represent program behavior, which are used as input to the Stacked Autoencoders-based deep learning model for detecting anomalies in web applications.

Loukas et al. [114] used deep multilayer perceptron and recurrent neural networks and built an intrusion detection system in the cloud environment. They used a robotic vehicle to evaluate their system by detecting various type attacks, including denial-of-service attack, command injection attack, and malware attack. They tested various traditional learning models and achieved much better average accuracy of 87% with deep learning.

Jeong et al. [115] proposed a system for detecting malware in PDF files. Their CNN-based model consists of one embedding layer, two convolutional layers, one pooling layer, one fully connected layer, and one output layer. The first layer is used to represent contextual meaning of the byte values and generate E-dimensional vectors, which are then given as input to the convolutional layers.

Homayoun et al. [116] proposed an LSTM and CNN-based deep learning approach for ransomware detection and classification. They use a deep feature extractor and a one-class classifier. It records the executed events when an application is started and transforms the sequence of the events to a numerical form and combines the input datasets to a single dataset. They use two different deep learning tasks for ransomware detection and classification, respectively. Other literature focusing on different types of malware and vulnerabilities detection include [117–119].

8. Research during the Recent Years

As obvious from Figure 2, the use of deep learning methods for malware and intrusion detection system is on the rise and has been increasing each year. In this section, we have selected a few important and most cited studies from the recent years (i.e., 2021–22) that the readers and researchers would be more interested in. Table 6 summarizes these studies.

9. Major Deep Learning Algorithms Used in Malware Detection

In order to answer RQ4 (What are the major DL algorithms used in the domain of malware detection?), we collected information about the usage of different DL algorithms in any form by the researchers. From the summary results shown in Figure 4, it is evident that Convolutional Neural Networks were used in most of the studies by the researchers which represents more than 50% of the publications surveyed, while LSTM-based neural networks in different forms were used by 25 researchers, which make up 25%. Similarly, DBN-based and AE-based algorithms were used in 13 and 11 publications that form 12% and 10.3%, respectively, of the total publications reviewed. Like many other domains, in malware detection and classification, most of the researchers have preferred convolutional neural networks when choosing a deep learning model. CNN is one of the most popular deep learning networks, which is capable of detecting the significant features without supervision. It is widely used for classification tasks, such as plant diseases, object detection, medical image analysis and computer vision, and so on. It is especially reported effective in image classification and image/object detection.

10. Discussion

10.1. Effectiveness of Deep Learning in Malware Detection. Deep learning produces best results with unstructured data. As most of the data produced by various systems is unstructured and in various formats, we either need to structure the data or have systems that have the capability to process unstructured data. Deep learning enables us to develop malware detection systems that can produce better results with unstructured and unlabeled data as well. Moreover, a deep learning algorithm can perform thousands of complex and repetitive tasks in very short time when trained once and produces accurate results as long as the raw data provided represents the problem.

Traditional malware detection techniques do not use machine learning or deep learning algorithms, and their performance is quite limited when it comes to detecting new types of malware. They rely on regularly updating their "malware definitions," which are used to detect threats. On the other hand, machine learning and deep learning-based algorithms can discover complex structures in structured and unstructured data when once trained and are very useful in developing effective malware detection systems. Hackers are developing malware that can change their code when propagated and thus hard to detect with the traditional pattern matching techniques. These malware can also deceive the traditional pattern matching-based systems easily in an intelligent manner. The different behavioral patterns that malware share could be used to detect unknown malware using ML and DL techniques.

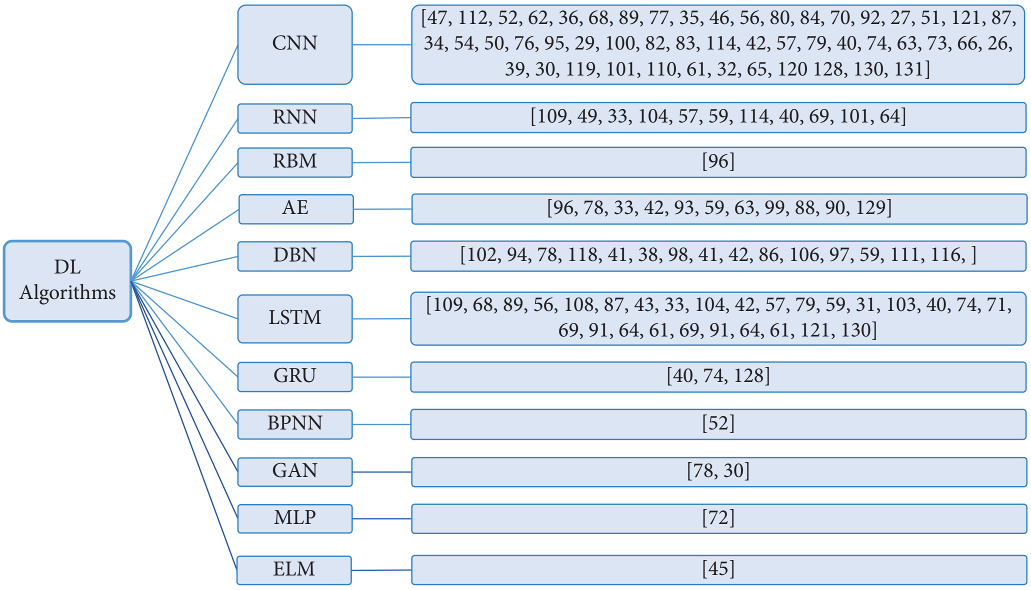

FIGURE 4: Major deep learning algorithms used in the domain of malware detection.

10.2. Performance of DL Compared with Traditional Learning. Deep learning algorithms have become popular as they can deliver more accurate results when trained with large amounts of data as compared to traditional learning algorithms. These algorithms can learn high-level features from data and mostly do not need domain expertise and hard-core feature extraction. The authors of some of the papers we reviewed used both deep learning and traditional learning algorithms for malware detection and experimentally showed that deep learning algorithms performed better than machine learning algorithms. Rhode et al. [39] proposed an early-stage malware detection system that is intended to detect malware within a few seconds of execution of a program. They used RNN and compared it with traditional learning algorithms, such as SVM. SVM achieved considerable accuracy of 80%, but RNN outperformed it after 1 second of execution and achieved an accuracy of 96% at 19 seconds into execution. Random Forest classifier achieved accuracy of 92% while Decision Trees achieved 92.6% accuracy. Haddadpajouh et al. [103] achieved an accuracy of 98.18% using RNN-LSTM for threat hunting in the Internet of things environment. They also used traditional machine learning algorithms and achieved the highest accuracy of 94% using KNN. Loukas et al. [114] developed an intrusion detection system for detecting cyber-attacks against vehicles, which achieved an accuracy of 86.9% with RNN. They also used several machine learning algorithms, including Logistic Regression, DT, RF, and SVM, achieving accuracy of 73.3%, 74%, 77.3%, and 79.9%, respectively. The cyber security threats detection system developed by Ullah et al. [102] performed better compared to other systems that used traditional learning algorithms and achieved much higher accuracy of 96%. Vinayakumar et al. [109] developed a deep neural network-based intrusion detection system and achieved an accuracy of up to 99.2%. They achieved a much lower average accuracy of around 80% with classical machine learning algorithms. Luo et al. [81] worked on detecting android malware and used Attention-CNN-LSTM in their system. They compared their deep learning-based model with SVM-based and KNN-based models and achieved a higher average accuracy of 96% compared to 95% with KNN and 94% with SVM. Similarly, Pektaş and Acarman [78] proposed an android malware detection system using instruction calls graphs and achieved 91.4% accuracy. They compared the proposed method with traditional learning algorithms, including KNN, Logistic Regression, SVN, and RF, and achieved accuracy of 80%, 70%, 79%, and 89%, respectively. Schranko de Oliveira and Sassi [90] acheived an accuracy of 91% with their deep neural network-based android malware detection system outperforming several ML algorithms, including SVM, RF, Logistic Regression, Extra Trees, and KNN. On the contrary, Jain et al. [55] achieved better accuracy of 97.7% using ELM with just one hidden layer as compared to 96.3% with CNN based architecture. Similarly, Pastor et al. [118] tested various traditional learning algorithms and CNN for detecting cryptomining traffic and achieved equal or better results with traditional learning algorithms.

In most of the cases, deep learning models performed much better than traditional learning methods. All these statistics show that deep learning algorithms are more capable of detecting and hunting malware or other threats and using shallow learning techniques may not lead us to a scalable solution with significant accuracy. However, it is not guaranteed that DL algorithms will always perform better than ML algorithms as some of the studies and our experiment's results depict. In our case, we compared the performance of Deep Autoencoders with different ML algorithms and achieved a higher accuracy rate with traditional learning models. Table 11 summarizes the results of our experiments. Our AE-based model could outperform only the Naïve Bayes ML algorithm.

Table 11: Analysis of the sustainability of android-based malware detection techniques.

Ref.	Evolvability/ability of identifying new malware? (yes/no)	DL model need updating/ retraining? (yes/no)	Sustainability/ resilience against evolution? (yes/no)	Sustainable up to years/accuracy	Initial accuracy/ F_1 score
Towards sustainable android malware detection [132]	Yes	After 5 years	Yes	5 (82%)	Above 93%
MaMaDroid [133]	Yes	After 2 years	Yes	2 (87%)	99%
Frequency analysis model (FAM)—a variant of MaMaDroid [134]	Yes	After several years	Yes	Several (76%)	81%
DroidSpan [135]	Yes	After 7 years	Yes	1–7 (21–37% superior than competitor)	6–32% superior than competitor
RevealDroid [136]	Yes	After 3 years	Yes	3 (87%)	98%
DroidCat [137]	Yes	After 9 years	Yes	9 (97%)	97%
G-Droid [138]	Yes	—	Yes	—	98.99%

10.3. Challenges in Malware Detection Using Deep Learning

10.3.1. The Issue of Data Unavailability. Machine learning and deep learning algorithms need a huge amount of training data to start with. In malware and threat detection, one of the biggest challenges is to provide a sufficient amount of malware and benign samples to the algorithm. Many datasets are available for use in research, but they need to be updated frequently so that the most recent malware samples could be used for training models.

10.3.2. The Issue of Data Noise and Model Overfitting. Another big challenge is the risk of wrongly labelled and noisy data, which may result in "overtraining the model" that leads to incorrect results.

10.3.3. The Issue of Model Validation and Response to New Threats. Many of the studies achieved high accuracy rates. However, they did not provide experimental results demonstrating how their systems would perform if a new type of malware attacked. New types of malware are being created across the world with passage of time and the malware detection systems should not only be able to detect variants of the malware samples that are used for training but also new types of malware to actually get the advantage of deep learning over the traditional threat detection systems.

10.3.4. The Issue of Evolution. Since a key issue in the android ecosystem is its fast evolution and various problems caused by the evolution [139, 140], the development of a flexible model that could be used at all times for the detection of new types of threats in the future is required. In the case the model does not need frequent retraining to cope with the situation of arising new malware but needs to update the model after a few years with very low degradation in the model performance, an evolvable model leads to sustainability in the model.

10.3.5. The Issue of Automatic Selection of DL Algorithm. In literature, a large number of DL methods can be found, which deal with complex problems, like malware classification over the big data. However, like ML algorithm (s) selection problem [141], the researchers always find it difficult to decide which method to pick for their problems in hand without frequently training, testing, and adopting the model.

10.3.6. The Issue of Sustainability. The frequent changes and continuous evolution of android malware demands for frequent retaining of the supervised malware detection models, which is a challenging job [142–144]. This requires building a sustainable malware detection model to update itself over the time in an effective and scalable manner. In case of declaring a model as sustainable, the frequency of retaining, duration for which the model is sustainable, and degradation in performance after the declared period are a few characteristics that need to be considered. For further details, Table 11 summarizes a few sustainable models with their key characteristics.

10.3.7. The Issue of Automatic Features Engineering and Analysis for Robust Modeling. One of the key challenges is how to pick or automatically learn, in the case of using deep learning for automatic feature engineering, features that stand the best of the time and future without frequent retraining. In literature, static, dynamic, and hybrid analysis methods have been used which automatically extract features for learning the DL model [145–147]. Column 2, "Automatic DL algorithm selection (yes/no)," of Table 3 summarizes these methods.

10.4. Data Quality for Malware Detection Using Deep Learning. Data quality is an essential component for machine and deep learning tools used for malware detection. Hence, along with technical approaches, availability of a sizable and informative dataset plays an essential role in the predictive accuracy of such systems. Therefore, the

Table 12: Android-based malware detection datasets for research community.

Reference	Dataset: description, size, type
[150]	15,451 benign apps and 15,183 malware
AndroZoo [151]	More than three million apps
AndroCT [152]	A large-scale dataset on the run-time traces of function calls in 35,974 benign and malicious android apps from ten historical years (2010 through 2019)
Rmvdroid [153]	Malware dataset containing 9,133 samples that belong to 56 malware families over the four years of 2014–18
[154]	17,664 apps sampled from the apps developed in each of the past eight years (2012–21)
AndroZooOpen [155]	AndroZooOpen, currently contains over 45,000 app artefacts, a representative picture of Github-hosted android apps
Deep ground [156]	Dataset (containing 24,650 malware apps)
DREBIN	In an evaluation with 123,453 applications and 5,560 malware samples DREBIN
Malgenome [157]	1,200 malware samples that cover the majority of existing android malware families, ranging from their debut in August 2010 to recent ones in October 2011

Table 13: Windows-based malware datasets.

Reference	Dataset: description, size, type
API Call dataset [149]	7107 different malicious software belonging to various families, such as virus, backdoor, trojan, and so on, have been analyzed, categorized into its different families, and made available for researchers to work on.
EMBER [158]	A labelled benchmark dataset for training machine learning models to statically detect malicious Windows portable executable files. This dataset includes features extracted from 1.1 M binary files.
SOREL-20 M [159]	A large-scale dataset consisting of nearly 20 million files with preextracted features and metadata, high-quality labels derived from multiple sources, information about vendor detections of the malware samples at the time of collection, and additional “tags” related to each malware sample to serve as additional targets.

quality of dataset should be carefully considered when creating and developing predictive models and tools for malware detection [148]. Such datasets are created by the research community to serve as a source of research for empirical analysis and extracting new insights about apps. In case of malware detection, data quality may be read as the number and types of apps used in android-based malware detection and operating system’s application programming interfaces (API) calls in case of Windows-based malware detection [149]. The details are explained in the subsequent sections.

10.5. Android-Based Malware Detection Datasets. In literature, a number of android-based malware datasets have been curated and made publicly available for researchers to perform their research activities. The details of a few most important datasets are made available here, in Table 12, with brief description of their characteristics.

10.6. Windows-Based Malware Datasets. Like android-based malware detections datasets, different malicious programs belong to different families, such as backdoor, adware, downloader, dropper, spyware, trojan, virus and worms, and so on, have been studied by researchers in windows environment in the form of API calls. These are collected into different datasets, which can be used by analysts to build intelligent automatic malware detection systems in windows environment. A few examples of such datasets have been provided in Table 13.

11. Recommendations and Future Perspective

In this section, on the basis of our study and observations, we make some recommendations for the readers and the future researchers in the domain of security and malware threat detection using ML and DL techniques. A large number of researchers have focused on developing intelligent systems for malware threat detection and classification; however, very few of them have considered using big data analytics tools.

(i) With the growth of the Internet, the enormous amount of data being generated could not be handled using traditional data processing techniques. Big data analytics frameworks, such as Apache Hadoop and Apache Spark, are being adopted by organizations and websites to handle the big data generated in an efficient manner in relatively lesser time, which would not be possible otherwise.

(ii) The same is the case with security systems and threat hunting software, which need to deal with huge amounts of data within the machines and across the web. DL- and ML-based systems, integrated with big data processing tools, may be much more efficient and cost effective, especially in the domain of Internet security.

(iii) It is a big question whether the different DL-based security systems proposed in these studies would perform as good with big data as they perform theoretically.

(iv) We also observe that the deep learning-based security systems have mostly focused on windows and android platforms as compared to web-based security systems. Internet security is not less important than securing a computer offline or a smartphone.

(v) The results of this study lead us to the conclusion that developing deep learning-based intelligent Internet security systems is one of the areas in DL we need to focus on.

(vi) As obvious from the results of this study, many researchers have achieved a very high accuracy rate of up to 99.9% in malware detection; however, we need to have these malware hunting systems operating in real world performing as perfectly as these statistics show. Future researchers should focus on how to enable the users to easily and effectively use deep learning for protection against malware.

(vii) The scholars are recommended to work on sustainable and self-evolvable models that do not require frequent retraining.

12. Conclusion

In this review paper, we extensively studied the recent research publications that aimed at using deep learning for malware detection on various platforms, like Windows, smartphones, IoT, and the Internet. We searched five different libraries, including Google Scholar, Springer, Science Direct, IEEE Explore, and ACM Digital Library, to retrieve the relevant literature published during the last six years. We collected a total of 290 studies and then carefully studied all of them to select the studies to include in this survey. We excluded duplicates, non-English literature, book chapters, SLRs, and conference papers and finally selected a total of 107 publications to review. A lot of work has been done in the field of DL techniques for malware detection and classification, and various systems have been developed that have achieved accuracy as high as 99.9%. The statistics show that CNN, AE, RNN, and LSTM are used by most of the researchers. Python and Python-based libraries, like TensorFlow, Keras, and scikit-learn, are widely used to implement the DL models. However, there is less or no information about how these DL-based security systems would perform when applied in real-world scenarios and the question remains unanswered whether these systems would be able to handle the huge amounts of data being produced in organizations and online.

The current study presents a big picture of the deep learning-based models used for threats classification and detection on a number of platforms, including Windows, Android, IoT, cloud computing, and the Internet. In the future, we intend to conduct extensive reviews for each of these platforms that will be useful for the researchers focusing on a specific platform.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported with research funds from Research Incentive Funds (R20090), Zayed University, United Arab Emirates.

References

[1] N. Idika and A. P. Mathur, “A survey of malware detection techniques,” *Purdue University*, vol. 48, no. 2, 2007.

[2] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, “Semantics-aware malware detection,” in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P’05)*, pp. 32–46, IEEE, Oakland, CA, USA, 2005 May.

[4] E. Gandotra, D. Bansal, and S. Sofat, “Malware analysis and classification: a survey,” *Journal of Information Security*, vol. 05, no. 02, pp. 56–64, 2014.

[5] A. S. Bist, “A survey of deep learning algorithms for malware detection,” *International Journal of Computer Science and Information Security*, vol. 16, no. 3, 2018.

[6] A. Naway and Y. Li, “A review on the use of deep learning in android malware detection,” 2018, https://arxiv.org/abs/1812.10360.

[7] T. Sonali Kothari and V. Khedkar, “Analysis of recent trends in malware attacks on android phone: a survey using scopus database,” *Library Philosophy and Practice*, pp. 1–20, 2021.

[8] B. Yadav and S. Tokekar, “Recent innovations and comparison of deep learning techniques in malware classification: a review,” *International Journal of Information Security Science*, vol. 9, no. 4, pp. 230–247, 2021.

[9] M. V. R. Kumar, S. Anand Kumar, A. Bando, S. R. Gs, H. Shah, and S. C. Reddy, “A survey of deep learning techniques for malware analysis,” *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 6031–6042, 2020.

[10] H. Lubuva, Q. Huang, and G. C. Msonde, “A review of static malware detection for Android apps permission based on deep learning,” *International Journal of Computer Networks and Applications*, vol. 6, no. 5, pp. 80–91, 2019.

[11] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, “A survey of deep learning-based network anomaly detection,” *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2019.

[12] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, “Real-time big data processing for anomaly detection: a survey,” *International Journal of Information Management*, vol. 45, pp. 289–307, 2019.

[13] A. Souri and R. Hosseini, “A state-of-the-art survey of malware detection approaches using data mining techniques,” *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 3–22, 2018.

[14] S. Sharma and A. Kaul, “A survey on Intrusion Detection Systems and Honeypot based proactive security mechanisms in VANETs and VANET Cloud,” *Vehicular communications*, vol. 12, pp. 138–164, 2018.

[15] J. Martínez Torres, C. Iglesias Comesaña, and P. J. Garcia-Nieto, “Review: machine learning techniques applied to cybersecurity,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2823–2836, 2019.

[16] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, “Intrusion detection systems in the Internet of things: a

comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019.

[17] R. Coulter and L. Pan, "Intelligent agents defending for an IoT world: a review," *Computers & Security*, vol. 73, pp. 439–458, 2018.

[18] E. Benavides, W. Fuertes, S. Sanchez, and M. Sanchez, "Classification of phishing attack solutions by employing deep learning techniques: a systematic literature review," *Developments and advances in defense and security*, pp. 51–64, 2020.

[19] K. Bakour, H. M. Ünver, and R. Ghanem, "The Android malware detection systems between hope and reality," *SN Applied Sciences*, vol. 1, no. 9, pp. 1120–1142, 2019.

[20] M. Aly, F. Khomh, M. Haoues, A. Quintero, and S. Yacout, "Enforcing security in Internet of Things frameworks: a systematic literature review," *Internet of Things*, vol. 6, Article ID 100050, 2019.

[21] A. M. Aleesa, B. B. Zaidan, A. A. Zaidan, and N. M. Sahar, "Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions," *Neural Computing & Applications*, vol. 32, no. 14, pp. 9827–9858, 2020.

[22] Z. Wang, Q. Liu, and Y. Chi, "Review of android malware detection based on deep learning," *IEEE Access*, vol. 8, pp. 181102–181126, 2020.

[23] B. Kitchenham, "Procedures for Performing Systematic Reviews," *Keele UK Keele University*, vol. 33, pp. 1–26, 2004.

[24] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," 2007.

[25] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," vol. 5, EBSE, Mumbai, India, 2007, Technical report Ver. 2.3 EBSE Technical Report.

[26] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.

[27] Y. Zhao, C. Xu, B. Bo, and Y. Feng, "Maldeep: a deep learning classification framework against malware variants based on texture visualization," *Security and Communication Networks*, vol. 2019, Article ID 4895984, 11 pages, 2019.

[28] J. Zhang, K. Zhang, Z. Qin, H. Yin, and Q. Wu, "Sensitive system calls based packed malware variants detection using principal component initialized MultiLayers neural networks," *Cybersecurity*, vol. 1, no. 1, pp. 10–13, 2018.

[29] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," *Computers & Security*, vol. 84, pp. 376–392, 2019.

[30] W. Zhong and F. Gu, "A multi-level deep learning system for malware detection," *Expert Systems with Applications*, vol. 133, pp. 151–162, 2019.

[31] B. Zhang, W. Xiao, Xi Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," *Future Generation Computer Systems*, vol. 110, pp. 708–720, 2020.

[32] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, pp. 461–472, 2017.

[33] S. Yue, "Imbalanced malware images classification: a cnn based approach," 2017, https://arxiv.org/abs/1708.08042.

[34] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: a heterogeneous deep learning framework for intelligent malware detection," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 265–285, 2018.

[35] L. Xiaofeng, Z. Xiao, J. Fangshuo, Y. Shengwei, and S. Jing, "ASSCA: API based sequence and statistics features combined malware detection architecture," *Procedia Computer Science*, vol. 129, pp. 248–256, 2018.

[36] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.

[37] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications*, vol. 47, pp. 377–389, 2019.

[38] M. Tang and Q. Qian, "Dynamic API call sequence visualisation for malware classification," *IET Information Security*, vol. 13, no. 4, pp. 367–377, 2019.

[39] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Computers & Security*, vol. 77, pp. 578–594, 2018.

[40] M. F. Rafique, M. Ali, A. S. Qureshi, A. Khan, and A. M. Mirza, "Malware classification using deep learning based feature extraction and wrapper based feature selection technique," 2019, https://arxiv.org/abs/1910.10958.

[41] M. H. Nguyen, D. L. Nguyen, X. M. Nguyen, and T. T. Quan, "Auto-detection of sophisticated malware using lazy-binding control flow graph and deep learning," *Computers & Security*, vol. 76, pp. 128–155, 2018.

[42] A. Namavar Jahromi, S. Hashemi, A. Dehghantanha et al., "An improved two-hidden-layer extreme learning machine for malware hunting," *Computers & Security*, vol. 89, Article ID 101655, 2020.

[43] Q. Le, O. Boydell, B. Mac Namee, and M. Scanlon, "Deep learning at the shallow end: malware classification for non-domain experts," *Digital Investigation*, vol. 26, pp. S118–S126, 2018.

[44] J. Y. Kim, S. J. Bu, and S. B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460-461, pp. 83–102, 2018.

[45] M. Kalash, M. Rochan, N. Mohammed, N. Bruce, Y. Wang, and F. Iqbal, "A deep learning framework for malware classification," *International Journal of Digital Crime and Forensics*, vol. 12, no. 1, pp. 90–108, 2020.

[46] S. Huda, S. Miah, J. Yearwood, S. Alyahya, H. Al-Dossari, and R. Doss, "A malicious threat detection model for cloud assisted internet of things (CoT) based industrial control system (ICS) networks using deep belief network," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 23–31, 2018.

[47] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 15–28, 2019, \.

[48] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[49] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50–58, 2019.

[50] L. Chen, "Deep transfer learning for static malware classification," 2018, https://arxiv.org/pdf/1812.07606.

[51] E. D. O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin, and F. C. Bernardini, "A model based on lstm neural networks to identify five different types of malware," *Procedia Computer Science*, vol. 159, pp. 182–191, 2019.

[52] A. F. Agarap, "Towards building an intelligent anti-malware system: a deep learning approach using support vector machine (SVM) for malware classification," 2017, https://arxiv.org/abs/1801.00318.

[53] E. Masabo, K. S. Kaawaase, and J. Sansa-Otim, "Big data: deep learning for detecting malware," in *Proceedings of the 2018 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, pp. 20–26, IEEE, Gothenburg, Sweden, 2018 May.

[54] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on Markov images and deep learning," *Computers & Security*, vol. 92, Article ID 101740, 2020.

[55] M. Jain, W. Andreopoulos, and M. Stamp, "CNN vs ELM for Image-Based Malware Classification," 2020, http://arXiv.org/2103.13820.

[56] F. O. Catak, A. F. Yazı, O. Elezaj, and J. Ahmed, "Deep learning based Sequential model for malware analysis using Windows exe API Calls," *PeerJ Computer Science*, vol. 6, Article ID e285, 2020.

[57] Y. Fang, Y. Zeng, B. Li, L. Liu, and L. Zhang, "DeepDetectNet vs RLAttackNet: an adversarial method to improve deep learning-based static malware detection model," *PLoS One*, vol. 15, no. 4, Article ID e0231626, 2020.

[58] H. Darabian, S. Homayounoot, A. Dehghantanha et al., "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, no. 2, pp. 293–303, 2020.

[59] R. Mitsuhashi and T. Shinagawa, "High-accuracy malware classification with a malware-optimized deep learning model," 2020, https://arxiv.org/abs/2004.05258.

[60] D. Gibert, C. Mateu, and J. Planes, "HYDRA: a multimodal deep learning framework for malware classification," *Computers & Security*, vol. 95, Article ID 101873, 2020.

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

[62] M. Cho, J. S. Kim, J. Shin, and I. Shin, "Mal2d: 2d based deep learning model for malware detection using black and white binary image," *IEICE - Transactions on Info and Systems*, vol. 103, no. 4, pp. 896–900, 2020.

[63] Y. Sung, S. Jang, Y.-S. Jeong, and J. H. J. J. Park, "Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations," *Computer Communications*, vol. 153, pp. 342–348, 2020.

[64] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *Journal of Signal Processing Systems*, vol. 93, no. 2-3, pp. 265–273, 2021.

[65] K. Devi, "Android malware detection using deep learning," *International Research Journal of Engineering and Technology (IRJET)*, Academia, vol. 06, no. 05, 2019.

[66] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Android malware detection using deep learning on api method sequences," 2017, https://arxiv.org/abs/1712.08996.

[67] Y. Suleiman, S. Sezer, and I. Muttik, "Android malware detection: An eigenspace analysis approach," in *Proceedings of the 2015 Science and Information Conference (SAI)*, pp. 1236–1242, 2015.

[68] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2019.

[69] N. Milosevic and J. Huang, "Deep learning guided Android malware and anomaly detection," 2019, https://arxiv.org/abs/1910.10660.

[70] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[71] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 371-372, Chicago, IL, USA, 2014 August.

[72] Y. S. Yen and H. M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectronics Reliability*, vol. 93, pp. 109–114, 2019.

[73] N. Xie, X. Di, X. Wang, and J. Zhao, "Andro_MD: android malware detection based on convolutional neural networks," *International Journal of Performability Engineering*, vol. 14, no. 3, p. 547, 2018.

[74] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3035–3043, 2019.

[75] S. Q. Luo, B. Ni, P. Jiang, S. W. Tian, L. Yu, and R. J. Wang, "Deep learning in Drebin: android malware image texture median filter analysis and detection," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 7, pp. 3654–3670, 2019.

[76] D. Saif, S. Elgokhy, and E. A. Sallam, "Deep Belief Networks-based framework for malware detection in Android systems," *Alexandria Engineering Journal*, vol. 57, no. 4, pp. 4049–4057, 2018.

[77] A. Pektaş and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," *Soft Computing*, vol. 24, no. 2, pp. 1027–1043, 2020.

[78] A. Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, 2020.

[79] M. Nauman, T. Ali, S. Khan, and T. A. Syed, "Deep Neural Architectures for Large Scale Android Malware Analysis," *Cluster Computing*, vol. 21, pp. 1–20, 2018.

[80] A. Martín, V. Rodríguez-Fernández, and D. Camacho, "CANDYMAN: classifying Android malware families by modelling dynamic traces with Markov chains," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 121–133, 2018.

[81] L. Shiqi, Z. Liu, B. Ni, H. Wang, H. Sun, and Y. Yuan, "Android malware analysis and detection based on attention-CNN-LSTM," *Journal of Computers*, vol. 14, no. 1, pp. 31–43, 2019.

[82] M. A. Halim, A. Abdullah, and K. A. Z. Ariffin, "Recurrent neural network for malware detection," *Int. J. Advance Softwarw Computing. Appl*, vol. 11, no. 1, pp. 43–63, 2019.

[83] W. F. Elsersy and N. B. Anuar, "Android malware detection using deep belief network," *PERTANIKA JOURNAL OF SCIENCE AND TECHNOLOGY*, vol. 25, pp. 143–150, 2017.

[84] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on Autoencoders and API-

images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, 2020.

[85] T. Chen, Q. Mao, M. Lv, H. Cheng, and Y. Li, "Droidvecdeep: android malware detection based on Word2Vec and deep belief network," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 4, pp. 2180–2197, 2019.

[86] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, "Static malware detection and attribution in android byte-code through an end-to-end deep system," *Future Generation Computer Systems*, vol. 102, pp. 112–126, 2020.

[87] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.

[88] X. Pei, L. Yu, and S. Tian, "AMalNet: a deep learning framework based on graph convolutional networks for malware detection," *Computers & Security*, vol. 93, Article ID 101792, 2020.

[89] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, Article ID 8863617, pp. 1–11, 2020.

[90] A. Schranko de Oliveira and R. J. Sassi, "Chimera: an android malware detection method based on multimodal deep learning and hybrid analysis," 2020.

[91] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 2, pp. 157–171, 2020.

[92] M. Amin, D. Shehwar, A. Ullah, T. Guarda, T. A. Tanveer, and S. Anwar, "A deep learning system for health care IoT and smartphone malware detection," *Neural Computing & Applications*, vol. 34, no. 14, pp. 11283–11294, 2020.

[93] X. Su, W. Shi, X. Qu, Y. Zheng, and X. Liu, "DroidDeep: using Deep Belief Network to characterize and detect android malware," *Soft Computing*, vol. 24, no. 8, pp. 6017–6030, 2020.

[94] Z. Ma, H. Ge, Z. Wang, Y. Liu, and X. Liu, "Droidetec: android malware detection and malicious code localization through deep learning," 2020.

[95] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning," *Ad Hoc Networks*, vol. 101, Article ID 102098, 2020.

[96] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang, and T. Li, "OpCode-level function call graph based android malware classification using deep learning," *Sensors*, vol. 20, no. 13, p. 3645, 2020.

[97] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1563–1578, 2021.

[98] H. Zhu, Y. Li, R. Li, J. Li, Z. H. You, and H. Song, "SEDMDroid: an enhanced stacking ensemble framework for android malware detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 984–994, 2021.

[99] J. Feng, L. Shen, Z. Chen, Y. Wang, and H. Li, "A two-layer deep learning method for android malware detection using network traffic," *IEEE Access*, vol. 8, pp. 125786–125796, 2020.

[100] A. Azmoodeh, A. Dehghantanha, and K. K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE transactions on sustainable computing*, vol. 4, no. 1, pp. 88–95, 2019.

[101] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Mathematical Problems in Engineering*, vol. 2019, Article ID 8195395, pp. 1–10, 2019.

[102] F. Ullah, H. Naeem, S. Jabbar et al., "Cyber security threats detection in internet of things using deep learning approach," *IEEE Access*, vol. 7, pp. 124379–124389, 2019.

[103] H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[104] M. N. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018.

[105] A. Abusnaina, M. Abuhamad, H. Alasmary et al., "A deep learning-based fine-grained hierarchical learning approach for robust malware classification," 2020.

[106] H. Naeem, F. Ullah, M. R. Naeem et al., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, Article ID 102154, 2020.

[107] S. Lu, L. Ying, W. Lin et al., "New era of deeplearning-based malware intrusion detection: the malware detection and prediction based on deep learning," 2019, https://arxiv.org/abs/1907.08356.

[108] B. Yu, J. Pan, D. Gray et al., "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51542–51556, 2019.

[109] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[110] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, Article ID 100198, 2020.

[111] R. Priyadarshini and R. K. Barik, "A deep Learning Based Intelligent Framework to Mitigate DDoS Attack in Fog Environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 825–831, 2019.

[112] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries," *Neural Computing & Applications*, vol. 31, no. 11, pp. 8021–8033, 2019.

[113] Y. Pan, F. Sun, Z. Teng et al., "Detecting web attacks with end-to-end deep learning," *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 16–22, 2019.

[114] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018.

[115] Y. S. Jeong, J. Woo, and A. R. Kang, "Malware detection on byte streams of pdf files using convolutional neural networks," *Security and Communication Networks*, vol. 2019, pp. 1–9, 2019.

[116] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh et al., "DRTHIS: deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94–104, 2019.

[117] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing cnn based behavioural malware detection techniques

on cloud iaas," in *Proceedings of the International Conference on Cloud Computing*, pp. 64–79, Honolulu, HI, USA, 2020, September.

[118] A. Pastor, A. Mozo, S. Vakaruk et al., "Detection of encrypted cryptomining malware connections with machine and deep learning," *IEEE Access*, vol. 8, pp. 158036–158055, 2020.

[119] A. N. Jahromi, S. Hashemi, A. Dehghantanha, R. M. Parizi, and K. K. R. Choo, "An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 630–640, 2020.

[120] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, p. 344, 2021.

[121] S. Baek, J. Jeon, B. Jeong, and Y. S. Jeong, "Two-stage hybrid malware detection using deep learning," *Human-centric Computing and Information Sciences*, vol. 11, no. 27, pp. 10–22967, 2021.

[122] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Towards an interpretable deep learning model for mobile malware detection and family identification," *Computers & Security*, vol. 105, Article ID 102198, 2021.

[123] N. Zhang, Y. A. Tan, C. Yang, and Y. Li, "Deep learning feature exploration for android malware detection," *Applied Soft Computing*, vol. 102, Article ID 107069, 2021.

[124] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-LADY: deep learning based Android malware detection using Dynamic features," *J. Internet Server. Information. Security*.vol. 11, no. 2, pp. 34–45, 2021.

[125] I. Obaidat, M. Sridhar, K. M. Pham, and P. H. Phung, "Jadeite: a novel image-behavior-based approach for java malware detection using deep learning," *Computers & Security*, vol. 113, Article ID 102547, 2022.

[126] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, no. 4, pp. 725–738, 2022.

[127] R. Chaganti, V. Ravi, and T. D. Pham, "Deep learning based cross architecture internet of things malware detection and classification," *Computers & Security*, vol. 120, Article ID 102779, 2022.

[128] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A malware detection approach using autoencoder in deep learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.

[129] M. Kumar, "Scalable malware detection system using distributed deep learning," *Cybernetics & Systems*, pp. 1–29, 2022.

[130] A. A. Hamza, I. T. Abdel Halim, M. A. Sobh, and A. M. Bahaa-Eldin, "HSAS-MD analyzer: a hybrid security analysis system using model-checking technique and deep learning for malware detection in IoT apps," *Sensors*, vol. 22, no. 3, p. 1079, 2022.

[131] S. S Lad and A. C. Adamuthe, "Improved deep learning model for static PE files malware detection and classification," *International Journal of Computer Network and Information Security*, vol. 14, no. 2, pp. 14–26, 2022.

[132] H. Cai and J. Jenkins, "Towards sustainable android malware detection," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, pp. 350-351, Gothenburg Sweden, 2018, May.

[133] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: detecting android malware by building Markov chains of behavioral models," 2016.

[134] L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: detecting android malware by building Markov chains of behavioral models (extended version)," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 2, pp. 1–34, 2019.

[135] W. Li, X. Fu, and H. Cai, "Androct: ten years of app call traces in android," in *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 570–574, IEEE, Madrid, Spain, 2021 May.

[136] J. Garcia, M. Hammad, and S. Malek, "Lightweight, obfuscation-resilient detection and family identification of android malware," *ACM Transactions on Software Engineering and Methodology*, vol. 26, no. 3, pp. 1–29, 2018.

[137] H. Cai, N. Meng, B. Ryder, and D. Yao, "Droidcat: effective android malware detection and categorization via app-level profiling," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1455–1470, 2019.

[138] H. Gao, S. Cheng, and W. Zhang, "GDroid: android malware detection and classification with graph convolutional network," *Computers & Security*, vol. 106, Article ID 102264, 2021.

[139] H. Cai, "Embracing mobile app evolution via continuous ecosystem mining and characterization," in *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, pp. 31–35, 2020, July.

[140] G. Suarez-Tangil and G. Stringhini, "Eight years of rider measurement in the android malware ecosystem: evolution and lessons learned," 2018, https://arxiv.org/abs/1801.08115.

[141] R. Ali, S. Lee, and T. C. Chung, "Accurate multi-criteria decision making methodology for recommending machine learning algorithm," *Expert Systems with Applications*, vol. 71, pp. 257–278, 2017.

[142] X. Fu and H. Cai, "On the deterioration of learning-based malware detectors for Android," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 272-273, IEEE, Montreal, Canada, 2019 May.

[143] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, "DroidEvolver: self-evolving Android malware detection system," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 47–62, IEEE, Stockholm, Sweden, 2019 June.

[144] H. Cai, "Assessing and improving malware detection sustainability through app evolution studies," *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 2, pp. 1–28, 2020.

[145] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys*, vol. 44, no. 2, pp. 1–42, 2012.

[146] M. Akour, I. Alsmadi, and M. Alazab, "The malware detection challenge of accuracy," in *Proceedings of the 2016 2nd International Conference on Open Source Software Computing (OSSCOM)*, pp. 1–6, IEEE, Beirut, Lebanon, 2016 December.

[147] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 29–46, 2017.

[148] V. Sessions and M. Valtorta, "The effects of data quality on machine learning algorithms," *ICIQ*, vol. 6, pp. 485–498, 2006.

[149] F. O. Catak and A. F. Yazı, "A benchmark API call dataset for windows PE malware classification," 2019.

[150] H. Cai, X. Fu, and A. Hamou-Lhadj, "A study of run-time behavioral evolution of benign versus malicious apps in android," *Information and Software Technology*, vol. 122, Article ID 106291, 2020.

[151] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: collecting millions of android apps for the research community," in *Proceedings of the 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pp. 468–471, IEEE, Austin, TX, USA, 2016 May.

[152] W. Li, X. Fu, and H. Cai, "AndroCT: ten years of app call traces in android," in *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pp. 570–574, IEEE, Madrid, Spain, 2021 May.

[153] H. Wang, J. Si, H. Li, and Y. Guo, "Rmvdroid: towards a reliable android malware dataset with app metadata," in *Proceedings of the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pp. 404–408, IEEE, Montreal, QC, Canada, 2019 May.

[154] H. Cai and B. G. Ryder, "A longitudinal study of application structure and behaviors in android," *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2934–2955, 2021.

[155] P. Liu, L. Li, Y. Zhao, X. Sun, and J. Grundy, "Androzooopen: collecting large-scale open source android apps for the research community," in *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 548–552, Republic of Korea, 2020 June.

[156] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *Proceedings of the International Conference on Detection Of Intrusions And Malware, and Vulnerability Assessment*, pp. 252–276.

[157] Y. Zhou and X. Jiang, "Dissecting android malware: characterization and evolution," in *Proceedings of the 2012 IEEE symposium on security and privacy*, pp. 95–109, IEEE, San Francisco, CA, USA, 2012 May.

[158] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," 2018, https://arxiv.org/abs/1804.04637.

[159] R. Harang and E. M. Rudd, "SOREL-20M: a large scale benchmark dataset for malicious PE detection," 2020, https://arxiv.org/abs/2012.07634.