Machine Learning Based Android Malware Detection: A State of Art

Walid Abdul Hakim
Computer Engineering Department
Istanbul Kultur University
34158 Bakırköy, Istanbul, Turkey
walidhemati@gmail.com

***Abstract*—The worldwide use of Android Operating System has made them a popular target for malware attacks, which can compromise sensitive data of its users. Despite the introduction of many security mechanisms in the Android system, malware remains a significant threat to users which makes traditional malware detection techniques ineffective against a huge number of its applications. Machine Learning algorithms have shown a great promise in Android malware detection. Machine learning plays a crucial role in malware detection by enabling automated analysis and classification of potentially malicious software. Many researches have explored various approaches using supervised, unsupervised, deep learning, and online learning techniques. Many studies have also focused on using machine learning to prevent malware attacks on Android operating systems especially mobile devices as they contain most of a persons sensitive information from their account passwords to their banking details. The purpose of this review is to help academics gain a basic understanding of Machine learning based Android Malware Detection.**

***Index Terms*—Malware, android, detection, machine learning**

I. Introduction

Smartphones have grown exponentially in recent years and are widely used for various tasks such as calling, taking photos, payments, booking, messaging ,etc. which has increased the popularity of them being target for malware attacks. Among several mobile OS, Android is the dominant OS with over 72% world market share [1]. The open source nature of the Android platform makes it relatively simple for malware developers to create specialized software containing malware to harm the users by robbing their private information which includes their social network, online banking, passwords and many more. Obviously, this raises serious concerns regarding the security of devices and individual privacy.

Malware, short for "malicious software," refers to any software or code specifically designed to infiltrate, damage, disrupt, or gain access without authorization to computational devices and networks. It encompasses a broad range of harmful programs, including viruses, worms, Trojans, ransomware, spyware, adware, and other types of malicious applications [2]. Malware is typically created with malicious intent, aiming to compromise the integrity, confidentiality, or availability of data and systems, steal sensitive information, perform unauthorized actions, or cause general havoc and disruption. It can be distributed through various means, such as email attachments, malicious websites, infected software downloads, or exploited vulnerabilities in operating systems or applications.

Since malware is one of the primary means by which smartphones are attacked, anti malware is one of the most important tools in a smartphone's protection system. Malware attackers and Anti-Malware developers are engaged in an ongoing conflict it becomes a race against time to see who can adapt to new technology first as both proteges do.

As attackers have become more intelligent and creative in creating Malware which could easily bypass traditional signature-based detection solutions, these solutions have become largely ineffective in detecting most cases these days. An approach is required, that can be trained and would learn by itself, where its ultimate goal is to differentiate between the good-ware and the malware in Android OS.

Malware applications usually have two types.

1) Repacking: one of the most prevalent tactics used to spread malware via exploiting popular programs in which the attacker disassembles the code, injects it with malicious code, reassembles it, and ultimately uploads it.
2) Downloading: The most popular strategy in which the attacker induces consumers to download intriguing and beneficial programs, which generally include harmful virus that will harm their devices.

According to the company G DATA in there latest report found that there was an average of 5 apps been published every minute that contain malware which targets Android OS and would also the attackers mainly targeted large systems such as universities and companies due to their huge database containing valuable information of their users [3].

This paper presents the findings of 12 articles / journals that were done to compare the performance of several machine learning techniques for detecting Android malware. The results will provide important insights into the strengths and limits of different machine learning based Android malware detection and highlight opportunities for future works.

The remaining sections of the paper are organized as follows. The next section provides background information on malware detection and machine learning, offering an overview of the key concepts and techniques involved in these domains. This background information sets the stage for understanding the subsequent discussions and analyses. Following the background section, a comprehensive literature survey is presented. This survey aims to provide a thorough examination of existing research papers and studies related to malware detection using machine learning. The survey highlights the differences and distinguishing factors among the various approaches, methodologies, and algorithms employed in these studies. To aid in understanding and comparison, the differences are summarized in a table format, allowing for easy reference and analysis.

Finally, the paper concludes by outlining future directions and potential areas of research in the field of malware detection using machine learning. This section provides insights into the unresolved challenges and opportunities for improvement in the current state of the art. By drawing attention to these areas, the paper encourages further exploration and investigation to enhance the effectiveness and efficiency of malware detection techniques. The identified future works serve as a valuable guide for researchers and practitioners interested in advancing the field and addressing emerging threats in the realm of malware detection.

II. Background

A. Android Architecture

Android an open-source OS is mainly used for mobile devices which is a modified version of Linux kernel. it was developed by Google and is currently used by a larger number of companies to power their smart devices. The architecture stack consists of C/C++libraries, Linux kernel,modules and ART. ART converts codes like Java and Kotlin to Dex byte-code which is then run on a virtual machine. The kernel offers security measures to the OS and also is in charge of managing the devices memory,drivers, and power [4].

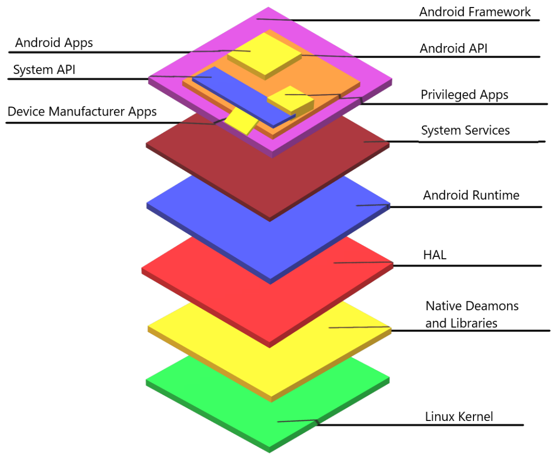

Fig. 1. Android system Architecture

As the name suggests ART(Android Runtime Environment) is where the Android applications are executed. Users may access all of feature sets using APIs, which were primarily built in Java. The Java API layer often offers the user APIs & serves as the foundation for nearly all the Android applications. The system apps layer contains all the essential programs, including calendars, email, the Google browser, and many more. Since these programs have no special rights, any other third-party program that performs the same job might take their place. Linux kernel is the one who is responsible in providing security to the Android OS. Typical in Linux user resources are kept separate from each other to ensure that they cannot be accessed by other users, upholding a certain amount of privacy [4].

B. Malware

Malware short for ”malicious software” is a phrase to describe a file or piece of code that may efficiently do any action desired by the attacker, such as exploring, infecting, performing operations, and stealing. Furthermore, because there are so many types of malware, there are several ways to attack. Despite its varied form and capabilities, malware seeks to get one of the following:

- Allow an attacker to get remote access to a compromised machine.
- From the infected device, send spam to unknown targets.
- Examine the afflicted user’s local network.
- Take vital information.

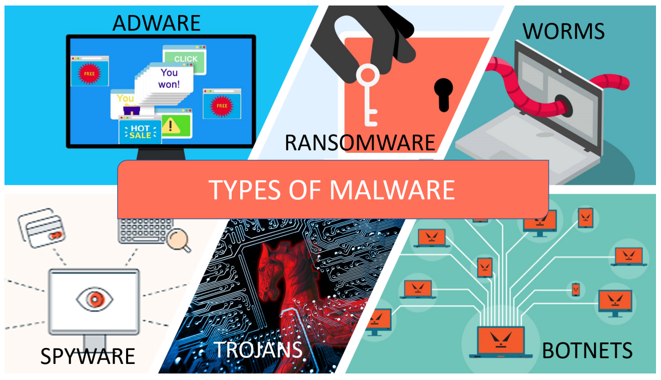

Fig. 2. Types of Malware

C. Popular Detection models for Android

In the world of Android malware detection there is 3 main detection models which are: I. Static analysis: Is a method of analysing the code of apps and programs without executing it and static analysis can detect the presence of a few certain patterns in codes that might be malicious. The main aim of Static Analysis is to identify any potential security risks that can be found in codes that might lead to malware attacks [23]. II. Dynamic analysis: Is a method that involves in running a program in a controlled environment where its behaviour can be monitored.The main aim of Dynamic analysis is to detect malware that may not be found through just the analysis of the code alone. III. Machine learning: ML is a subfield of AI that has a goal of building algorithms and models that allow computers to mimic how the human brain learn and steadily improve their accuracy in finding malware.

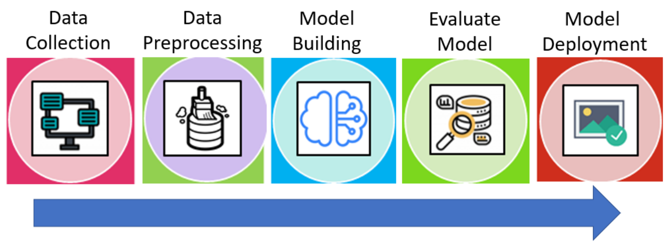

Fig. 3. Machine learning steps

D. Machine Learning (ML)

ML is a subfield of AI that has a goal of building algorithms and models that allow computers to mimic how the human brain learn and steadily improve their accuracy. The growth of malware has made traditional signature-based detection solutions less practical . Which led to ML being implemented by commercial anti-malware companies and researchers more and more. ML-based malware detection systems work leaps and bounds higher than traditional signature-based detection solutions. And with the help of ML they can even detect several iterations of malware that already exists and even completely brand-new ones.

*1) **Supervised Learning**:* is a type of ML that involves training the model with existing labelled data to predict incoming new data [5]. As supervised learning require labeled data it very important that the data used must very accurately labeled otherwise the model predictions would be very inaccurate and would also not improve its accuracy as it wouldn't be able to learn properly.

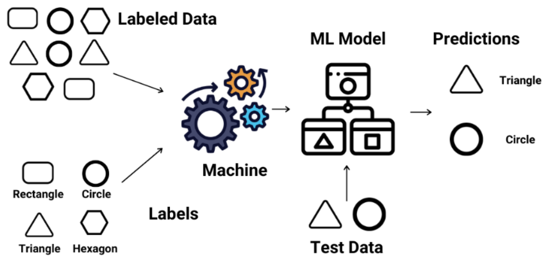

Fig. 4. Supervised Learning

Supervised learning is primarily utilized in Android malware detection to predict whether an App is malware or not. There is a huge library of predictive models available to be used in ML but only a very few of them are commonly used which are:

I. Logistic Regression: Logistic regression is a statistical model used to model the relationship between a binary response variable , one or more predictor variables and is commonly used in binary classification problems in various fields, including finance, healthcare, and marketing.

II. Decision Tree: is a modeling algorithm where it resembles a tree containing root nodes which represents a feature or attribute, branches which represents a decision rule and leafs which represents a predicted outcome or a class label.

III. Support Vector Machines (SVM): SVMs along side random forests are the most widely used classifiers and are used for classification, regression analysis and Android malware detection [5]. The main goal of SVMs are to find the optimal higher dimensional plane to separate the classes of data while maximizing the margins between them.

IV. Random Forest: is a tree based classifier that creates a collection of decision trees at training, and outputs the class that is the node of the classes of the trees. Each tree is trained on random subset of the training data and at each node which results in a reduction of the variance and over-fitting of the model, making it more generalizable to unseen data [5].

V. Naive Bayes: It is occasionally used in Android malware detection because NV is a straightforward classifier model that models conditional probabilities across features, making it particularly helpful on big data sets [5].

VI. K-Nearest Neighbour (KNN): Also known as a lazy learning algorithm where it does not build its own internal representation of the data in simple terms it doesn't build a model or even train on the data. The data points are classified based on the class to which they are the nearest to. Then the algorithm assigns the class to the data based on which class majority of its K-nearest neighbors are located, K is the user defined parameter.

*2) **Unsupervised Learning**:* Unsupervised learning is a sort of machine learning in which the model is trained without previously labeled data to predict incoming new data [5].In other words, it can detect or attempt to detect patterns and correlations in data on its own. Its major goal is to investigate the structure of the data, looking for similarities and contrasts as well as hidden patterns.

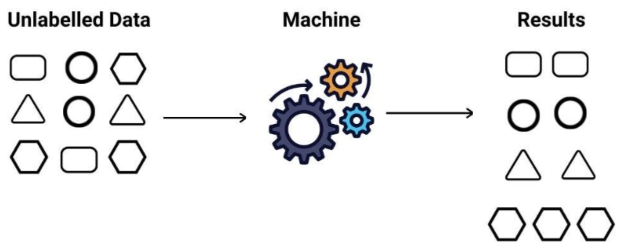

Fig. 5. Unsupervised Learning

Unsupervised learning is very useful where labeled data is not available and is very commonly used in applications such as Malware detection, customer segmentation , image and speech recognition.

*3) **Deep Learning**:* is a subset of ML which uses Artificial neural network structure that is inspired by the human brain, and its capabilities outperforms any traditional algorithms in laterally every situations [5]. Now lets look at some of the most commonly used deep learning approaches:

I. The Multi-layer Perceptron (MLP) is an artificial neural network (ANN) that consists of multiple layers of interconnected neurons. This architecture is commonly used in machine learning and is particularly effective for solving complex problems. The MLP operates as a feed-forward neural network, meaning that data flows in a single direction from the input layer to the output layer. The input layer receives the initial data, which is then processed through the hidden layers before reaching the output layer. The hidden layers, located between the input and output layers, play a vital role in the model's overall functioning.

II. Convolutional Neural Network (CNN): A feed-forward network that is made up of one or more convolutional , fully linked, and pooling layers. It is inspired by the human brain's ability to recognize patterns. CNNs have demonstrated outstanding performance in image recognition applications such as object identification, segmentation, and classification.

III. Restricted Boltzmann Machine (RBM): a neural network created at random that uses the probability distribution in the input data to teach itself new things during unsupervised learning [9]. An RBM consists of a hidden and visible layer, where the input data is the the visible layer and the neuron in the hidden layer is the learned features.

IV. Deep Belief Network (DBN): DNB is a type of probability GNN that is used for unsupervised learning. It is composed of multiple layers of RBMs that are stacked on each other. But has a high cost in creating the model. And works very well with high dimensional inputs. DNB can be used with and without labelled data which makes it very good in classification as it can combine the best of two worlds.

V. Recurrent Neural Networks (RNNs): Just as their name suggests, RNNs have loops in their architecture that allow them to use previous outputs as inputs to the next time step [5]. Although RNNs are used for supervised learning there is a RNN architecture called LSTM that can be used for unsupervised learning.

E. Datasets

Datasets used in the researches were many and the most common in them were Drebin and AMD. Drebin was released in 2014 created at North Carolina State University and contains more than 5,000 malware samples. The name was from the character Lt. Frank Drebin from the movie ”The Naked Gun”. It contains a range of malware families, such as spyware, trojans, and viruses, and each sample is labeled with its corresponding malware family.

Android Malware Dataset (The Argus Lab) or AMD was released on 2017 by University of Louisiana at Lafayette it was created as part of a research project on Android malware detection and classification. The dataset contained a collection of over 5,000 Android malware samples which had malware samples from different malware families, such as trojans, adware, and spyware.

The Table I below contains many of the datasets used in the researches and their information:

F. Advanced Optimization

In the use of deep learning and machine learning models the effect of the hyper parameters are important. Although this type of values can be tuned with a static approaches, new trends focus on the use of evolutionary algorithms such as genetic algorithms as used in different problem solving environments [6], [7]. For example in [8] the authors made a comparative performance analysis reveals that SMOTE for oversampling techniques and genetic algorithms for optimization yield improved results, as observed on real datasets.

III. Literature Survey

The papers [11]–[16], [19]–[22] used various machine learning algorithms such as SVM, KNN, Naives Bayes, BNB, RF, DT and many more while using public data sets like Drebin, MoDroid, AMD, CICMalDroid2020 , etc. where as [4], [10], [19] papers used either own data sets or a hybrid dataset containing their own with another public dataset. While the evaluation metrics were the same for all which were:

- The confusion matrix:
  1) True Positive (TP): Program is harmful, as was rightly foreseen.
  2) False Positive (FP): Program is safe but is mistakenly classified as malicious.
  3) True Negative (TN): Program is harmless and this was accurately foreseen.
  4) False Negative (FN): Program is harmful yet is incorrectly labeled as benign.
- Accuracy: The percentage of accurate forecasts across all application test sets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- Precision: The ratio of successfully classified malware from all forecasts of malware.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- Recall: the percentage of real malware that is accurately categorized.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- F1-Score: the harmonic mean of the two is calculated in order to combine accuracy and recall.

$$\begin{aligned} F1 - Score &= \frac{2 * Precision * Recall}{Precision + Recall} \\ &= \frac{2 * TP}{2 * TP + FP + FN} \end{aligned} \tag{4}$$

The paper [10] provided a multidimensional, kernel feature-based framework in which they investigated 112 kernel features of executing the task data structure in the Android system and assessed detection accuracy using a variety of datasets of varied dimensions. Their method of execution got them an accuracy of between 93.07% - 99.23% using various machine learning algorithms.

In the study [11] the authors implemented on the NB, SVM, and KNN machine learning methods against a comprehensive new Android malware dataset (CICInvesAndMal2019), in an effort to increase malware detection rates and contribute to research and actions aimed at safeguarding the growth of

TABLE I
DATASETS AND THEIR INFORMATION

Name	**Ref**	***Year***	***Record***	***# of Families***	***Type***	***Based***
CICMalDroid2017	[34]	2018	10,854 samples	42	Dynamic	Net-Traffic
CICInvesAndMal2019	[33]	2019	426 malware and 5,065 benign	58	Dynamic	Net-Traffic
Malgenome	[31]	2021	1260 malware	49	Static	binary
AMD	[30]	1999	4,354 malware	71	Dynamic	Net-Traffic
Drebin	[32]	2014	5,560 malware	179	Static	APK
M0Droid	[35]	2016	Unknown	153	Static	APK

mobile information access. The methodology used was a very straight forward as it had 4 phases,The first was data pre-processing, and the second was selecting the Android API calls and permission characteristics from the dataset, therefore distinguishing both feature sets. Importing the selected essential characteristics to the ML classifiers is the third process.Finally, in the last step, the findings are shown. Their method of execution got them an accuracy of between 84.33% - 94.36% using 3 machine learning algorithms.

The paper [13] developed a machine learning model based on the co-existence of static features for Android malware detection. The suggested approach assumes that malware for Android seeks an unusually large number of permissions.Several machine learning algorithms were used.The highest accuracy attained was 98% using Random Forest algorithm using Malgenome dataset and 95% using Drebin dataset.

A research group proposed a highly reliable classifier for Android Malware detection based on a Factorization Machine architecture and the extraction of Android app features [15]. The model used Drebin dataset and AMD dataset and several algorithms like SVM, NB-Bernoulli (NB-B), NB-Multinomial (NB-M), NB-Gaussian (NB-G), MLP and FM where the highest accuracy was from MLP 99.73% on Derbian and 99.05% for MLP and FM on the AMD dataset. The paper also claim to be 50 times faster than several commercial antivirus engines.which can be seen from the table V.

The Android operating system's widespread usage on mobile devices, combined with its financial incentives, has attracted malicious actors who create malware specifically targeting these devices. As a result, there has been a significant rise in the number of Android malware applications. In response to this security threat, signature-based detection, also known as static detection, has become a popular approach due to its simplicity of implementation and fast identification capabilities. Several studies [30], [31] have focused on developing advanced, effective, and reliable malware detection systems using deep learning algorithms. In this particular study, the authors aimed to evaluate the performance of RNN-based LSTM, BiLSTM, and GRU algorithms on the CICInvesAndMal2019 dataset. This dataset contains 8115 static features that are utilized for malware detection. The objective was to identify the most suitable deep learning algorithm for this task. Through their experiments, the authors discovered that the BiLSTM model consistently outperformed the other RNN-based deep learning methods. It achieved an impressive accuracy rate of 98.85%, showcasing its effectiveness in detecting Android malware.

The paper referenced as [16] employs multiple linear regression techniques to present two permission-based Android malware classifiers. These classifiers are then compared with fundamental machine learning algorithms including Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Naive Bayes (NB), and Decision Trees (DT) on four distinct datasets. The study follows a comprehensive 7-step methodology, beginning with dataset collection and subsequent feature extraction. Next, the features are carefully selected, followed by model training. The study further narrows down the models and combines the selected ones, ultimately resulting in the creation of Ensemble-1 and Ensemble-2. The combined model is then tested, and the obtained results are documented and presented in Table VI.

During the testing phase, every application is evaluated using the developed models [16]. A prediction is made for each application using a majority vote approach, where each model's output is considered, and the most frequent prediction is selected. This ensemble learning approach is referred to as "Ensemble-1." Additionally, a second ensemble learning model, known as "Ensemble-2," is developed. In this model, the training dataset is randomly divided into five subsets. Two of these subsets are used for training a linear Support Vector Machine (SVM), two subsets are used for training a Decision Tree (DT), and the remaining subset is used for training a linear regression model.

A similar work in the literature proposed the utilization of Reverse Engineered Android application features and Machine Learning techniques for detecting vulnerabilities in the Android platform [19]. The researchers initially put forward a model that leverages extensive datasets of malware samples along with innovative static feature sets. They further enhance the model's performance by employing ensemble learning and various machine learning methods. Remarkably, the results obtained from their experiments were highly promising, achieving an impressive accuracy rate of 96.24% in effectively detecting malware within Android applications.

The paper [20] proposed a method based on Random Forest algorithm, which used three different feature selection methods. It was conducted on Drebin dataset. Additionally, baselines for various techniques were compared to the candidate feature selection model. The article first utilized three feature selection procedures as pre-processing: effective, high weight, and effective group feature selection. The random forest algorithm then uses the chosen attributes to identify Android malware.

A survey paper in [21] reviewed of 42 highly cited papers that covered their use of ML methods, the features engineered, the dimensionality reduction techniques used, the training datasets used, and their evaluation and explanation strategies over the course of a decade of research (from 2011 to 2021). They had a 3 step methodology where their first step was to gather literature then review them in-depth and finally build a taxonomy according to their finds. The highest accuracy they found was using Extra Trees with an accuracy of 99.64% followed by 99% by using FNN, ANN, WANN, KMNN, LR, SVM and Decision Tree using various feature extraction and selection methods from extraction ranging from Static, Dynamic or being a hybrid using both and feature selection ranging from none selected , manual, to many other ways like fisher score, Mean decrease impurity, etc.

TABLE II
COMPARISON BETWEEN PAPERS

Paper	Year	Datasets	Algorithms	Acc.
[10]	2021	Private	DT, NB, NN & KNN	93.07
[11]	2022	CICInvesAndMal2019	KNN, NB & SVM	94.36
[13]	2023	Malgenome & Drebin	RF, DT, LR SVM & KNN	98.00
[15]	2019	AMD & Drebin	SVM, NB-G NM-B, NB-M MLP & FM LinRegDroid1 LinRegDroid2	99.73
[30], [31]	2022, 2023	CICInvesAndMal2019	RNN LSTM BiLSTM GRU	98.02 98.75 98.85 98.10
[16]	2022	AMD, M0Droid, Arslan & Lopez	KNN,DT,nm-NB SVM,mvnm-NB Ensemble-1 & Ensemble-2	98.53
[19]	2022	Mix/Hybrid	RBF, NB, SVM KNN, DT, Adaboost, & Proposed AL	96.24
[20]	2022	Drebin	KNN(k=n) SL, NB, DT RF10, RF50 RF100, RF-EF Plnfroid, SensDroid, & RNPDroid	99.80
[22]	2023	Private	DT, RF, LR NB, SVM, KNN Adaboost CNN & MLP	98.70%
[23]	2023	Drebin CICMaldroid2020	RF ET DNN 1D-CNN	97.07 97.67 98.26 98.2

A approach for producing feature vectors based on Huffman encoding is proposed in the work [22]. System call frequencies are used as indicators of malware's dynamic activity patterns. According to deep learning and machine learning assessments, the proposed model using the Random Forest classifier beats various current processes with an accuracy of 98.70%. The proposed technique was based on dynamic analysis, with system call frequencies serving as characteristics.

This research paper of [23] proposes the utilization of machine learning and deep learning approaches to classify malware families and categories, employing diverse datasets for evaluation and selecting suitable methods for each dataset. The findings of this study demonstrate that, following feature extraction and selection, along with training and evaluation using machine learning models, the Drebin and CICMaldroid2020 datasets, classified by family and category respectively, yield highly accurate results with a low false positive rate. Furthermore, they compared their outcomes with several prior studies, emphasizing the significance of findings.

From this we can clearly see improvements in nearly all models while using the proposed Huffman technique. Table II presents the results and details of several studies that were done on machine learning based Android Malware detection. The table includes several papers with there year of publish, Datasets used, Algorithms used and finally the highest accuracy attained by that study. Each of the studies have there own pros and cons from being very simple to very complex, 50 times faster then several commercial antivirus engines to being very time consuming , one being the best at one algorithm and other been good at all, etc, it all depends on the application of the methodology and the requirement of the researcher.

IV. Conclusion and Future Works

In conclusion, machine learning has emerged as a crucial tool in mitigating the escalating threat of Android malware. Traditional methods have proven inadequate in combating the growing number of Android malware attacks. However, the utilization of machine learning-based approaches, with sophisticated algorithms capable of identifying malware patterns, offers hope in the fight against Android malware. This paper provides a comprehensive overview of recent machine learning-based methods for Android malware detection, highlighting their advantages, limitations, and challenges. Some approaches propose novel feature techniques, while others leverage advanced machine learning algorithms, both achieving high accuracy rates. Notably, several techniques and methods have demonstrated exceptional results, with malware detection rates reaching as high as 99%. Machine learning-driven Android malware detection systems are continually evolving to confront the increasing risks posed by malware. The ongoing development and implementation of these systems by major companies will bolster Android security, safeguarding users against the growing menace of Android malware.

Based on the aforementioned findings, it would be beneficial to explore the application of deep learning techniques to further enhance detection accuracy. This could involve developing novel architectures or adapting existing ones to address the challenges of Android malware detection [26]. Additionally, combining multiple existing methods and leveraging GPU technologies for efficient processing of big data could yield promising results [27]. There remains ample room for improvement in machine learning-based Android malware detection, and the experiments conducted in this review paper lay a solid foundation for future research in this field. Furthermore, incorporating different feature selection mechanisms, as mentioned in [28], [29], can help reduce runtime and enhance the efficiency of the proposed models.

References

[1] D. Curry, "Android Statistics (2023)," Business of Apps, 27-Feb-2023. [Online]. Available: https://www.businessofapps.com/data/android-statistics/.

[2] E. C. Bayazit, O. K. Sahingoz and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2020, pp. 1-8, doi: 10.1109/HORA49412.2020.9152840.

[3] "G Data Mobile Security Report: Attacks on smartphones every minute," G DATA CyberDefense AG - Press Center. [Online]. Available: https://presse.gdata.de/news-g-data-mobile-security-report-attacks-on-smartphones-every-minute?id=174612&amp;menueid=28982&amp;l=english, 2023.

[4] A. Muzaffar, H. Ragab Hassen, M. A. Lones, and H. Zantout, "An in-depth review of machine learning based Android Malware Detection," Computers & amp; Security, vol. 121, p. 102833, 2022.

[5] "Topics," IBM. [Online]. Available: https://www.ibm.com/topics/.

[6] S. Birtane Akar, H. Korkmaz and O. K. Sahingoz, "Efficient Deployment of Wireless Sensor Nodes with Evolutionary Approaches," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-7, doi: 10.1109/HORA55278.2022.9800045.

[7] M. Teymournezhad and O. K. Sahingoz, "Path planning of Mobile Robot in Dynamic Environment by Evolutionary Algorithms," 2023 2nd International Conference on Computational Systems and Communication (ICCSC), Thiruvananthapuram, India, 2023, pp. 1-6, doi: 10.1109/ICCSC56913.2023.10142971.

[8] K. Konar, S. Das and S. Das, "Employee attrition prediction for imbalanced data using genetic algorithm-based parameter optimization of XGB Classifier," 2023 International Conference on Computer, Electrical & Communication Engineering (ICCECE), Kolkata, India, 2023, pp. 1-6, doi: 10.1109/ICCECE51049.2023.10085402.

[9] "A computer science portal for geeks," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/.

[10] X. Wang and C. Li, "Android malware detection through machine learning on kernel task structures," Neurocomputing, vol. 435, pp. 126–150, 2021.

[11] A. S. Shatnawi, Q. Yassen, and A. Yateem, "An Android malware detection approach based on static feature analysis using machine learning algorithms," Procedia Computer Science, vol. 201, pp. 653–658, 2022.

[12] V. Syrris and D. Geneiatakis, "On machine learning effectiveness for malware detection in Android OS using Static Analysis Data," Journal of Information Security and Applications, vol. 59, p. 102794, 2021.

[13] E. Odat and Q. M. Yaseen, "A novel machine learning approach for Android malware detection based on the co-existence of features," IEEE Access, vol. 11, pp. 15471–15484, 2023.

[14] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of Android Malware Detection Approaches based on machine learning," IEEE Access, vol. 8, pp. 124579–124607, 2020.

[15] C. Li, K. Mills, D. Niu, R. Zhu, H. Zhang, and H. Kinawi, "Android malware detection based on Factorization Machine," IEEE Access, vol. 7, pp. 184008–184019, 2019.

[16] D. O. Sahin, S. Akleylek, and E. Kilic, "LinRegDroid: Detection of Android malware using multiple linear regression models-based classifiers," IEEE Access, vol. 10, pp. 14246–14259, 2022.

[17] E. C. Bayazit, O. K. Sahingoz and B. Dogan, "A Deep Learning Based Android Malware Detection System with Static Analysis," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-6, doi: 10.1109/HORA55278.2022.9800057.

[18] E. C. Bayazit, O. K. Sahingoz and B. Dogan, "Deep Learning based Malware Detection for Android Systems: A Comparative Analysis," Tehnički vjesnik, Vol. 30 No. 3, 2023, pp. 787-796, doi: 10.17559/TV-20220907113227.

[19] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware detection: A framework for reverse engineered Android applications through Machine Learning Algorithms," IEEE Access, vol. 10, pp. 89031–89050, 2022.

[20] M. R. Keyvanpour, M. Barani Shirzad, and F. Heydarian, "Android malware detection applying feature selection techniques and machine learning," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9517–9531, 2022.

[21] M. Mehrabi Koushki, I. AbuAlhaol, A. D. Raju, Y. Zhou, R. S. Giagone, and H. Shengqiang, "On building machine learning pipelines for android malware detection: A procedural survey of practices, challenges and opportunities," Cybersecurity, vol. 5, no. 1, 2022.

[22] H. H. Manzil and S. Manohar Naik, "Android malware category detection using a novel feature vector-based machine learning model," Cybersecurity, vol. 6, no. 1, 2023.

[23] R. Bellairs, "What is static code analysis? Static Analysis Overview," Perforce Software. [Online]. Available: https://www.perforce.com/blog/sca/what-static-analysis#: :text=Static

[24] Intel, "Dynamic Analysis vs. Static Analysis," Intel. [Online]. Available: https://www.intel.com/content/www/us/en/docs/inspector/user-guide-windows/2022/dynamic-analysis-vs-static-analysis.html#: :text=Dynamic

[25] C. -D. Nguyen, N. H. Khoa, K. N. -D. Doan and N. T. Cam, "Android Malware Category and Family Classification Using Static Analysis," 2023 International Conference on Information Networking (ICOIN), Bangkok, Thailand, 2023, pp. 162-167, doi: 10.1109/ICOIN56518.2023.10049039.

[26] G. Sismanoglu, M. A. Onde, F. Kocer and O. K. Sahingoz, "Deep Learning Based Forecasting in Stock Market with Big Data Analytics," 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741818.

[27] S. I. Baykal, D. Bulut and O. K. Sahingoz, "Comparing deep learning performance on BigData by using CPUs and GPUs," 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), Istanbul, Turkey, 2018, pp. 1-6, doi: 10.1109/EBBT.2018.8391429.

[28] S. J. K., S. Chakravarty and R. K. Varma P., "Feature Selection and Evaluation of Permission-based Android Malware Detection," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 795-799.

[29] M. Korkmaz, O. K. Sahingoz and B. Diri, "Feature Selections for the Classification of Webpages to Detect Phishing Attacks: A Survey," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2020, pp. 1-9, doi: 10.1109/HORA49412.2020.9152934.

[30] 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, Bonn, Germany, 6-7 July 2017, pp. 252-276, Springer, 2017.

[31] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in 2012 IEEE Symposium on Security and Privacy, Piscataway, NJ, USA, 2012, pp. 95-109.

[32] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," NDSS 2014, vol. 14, pp. 23-26.

[33] L. Taheri, A. F. A. Kadir and A. H. Lashkari, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019, pp. 1-8, doi: 10.1109/CCST.2019.8888430.

[34] CICMalDroid: A Benchmark Dataset for Android Malware Detection, Canadian Institute for Cybersecurity (CIC), the University of New Brunswick, Canada.

[35] M. Damshenas, A. Dehghantanha, K.-K. R. Choo, and R. Mahmod, "M0Droid: An Android Behavioral-Based Malware Detection Model," Journal of Information Privacy and Security, vol. 11, pp. 141-157, 2015.