Android Malware Detection using Machine Learning Techniques KNN-SVM, DBN and GRU

Sk Heena Kauser#1, V.Maria Anu#2,

heenamtech@gmail.com, mariaanu18@gmail.com

[1]Research scholor, Department of Computer Science & Engineering,
Sathyabama Institute of Science and Technology, Chennai, India.
[2]Professor, Department of Computer Science & Engineering,
Sathyabama Institute of Science and Technology, Chennai, India.

Abstract

Android malware is now on the rise, because of the rising interest in the Android operating system. Machine learning models may be used to classify unknown Android malware utilizing characteristics gathered from the dynamic and static analysis of an Android applications. Anti-virus software simply searches for the signs of the virus instance in a specific programme to detect it while scanning. Anti-virus software that competes with it keeps these in large databases and examines each file for all existing virus and malware signatures. The proposed model aims to provide a machine learning method that depend on the malware detection method for Android inability to detect malware apps and improve phone users' security and privacy. This system tracks numerous permission-based characteristics and events collected from Android apps and analyses them using a classifier model to determine whether the program is good ware or malware. This method used the machine learning techniques KNN-SVM, DBN, and GRU in which help to find the accuracy which gives the different values like KNN gives 87.20 percents accuracy, SVM gives 91.40 accuracy, Naive Bayes gives 85.10 and DBN-GRU Gives 97.90. Furthermore, in this paper, we simply employ standard machine learning techniques; but, in future work, we will attempt to improve those machine learning algorithms in order to develop a better detection algorithm.

Keywords

Android Malware, Codes Vulnerability, Malware Detection, Machine Learning, Smartphones.

1. INTRODUCTION

Because of the ease and efficiency of various apps, as well as the ongoing development in the software and hardware on smartphone usage, smart devices, and its related application are fast developing in the modern era. By 2023, it's expected that it will be more than 4.3 billion smartphones users. The most popular smartphone operating system (OS) is Android.It has a market share of 72.2 percent in May 2021. Apple iOS has the second-highest marketplace share of 26.99 percent, with Samsung, KaiOS, and other minor suppliers sharing the remaining 0.81 percent. The official app store for Android smartphones is Google Play. As of May 2021, there were over 2.9 million applications available on it. AppBrain classifies more than 2.6 million of them as standard applications, while 0.5 million are labelled as ordinary apps. Because of its global prominence, Android is a more appealing aim for thieves and is much vulnerable to viruses and malware[1]. Various ways of identifying these assaults have been offered in studies, with machine learning being one of the most prevalent. This is because machine learning algorithms may generate a ordinate from a (restricted) collection of training instances [2].The usage of examples eliminates the required to describe signatures directly while constructing malware detectors. Signature definition necessitates knowledge and time-consuming human engagement, and while specific rules (signatures) do not exist for some attack situations, instances are readily available [3]. Android malware investigation may be done in three distinct methods, according to research. The deployment of static and dynamic content is the first technique. Before loading an application onto any device, the code of the program is examined for harmful components [4].

The second approach entails modifying the Android operating system to include components for tracking and intercepting any unusual behaviour that may occur on the machine, while the third process entails using virtualization to integrate domain detachment ranging from lighter weight application exclusion to running numerous incidents of Android Operating system on the same machine [5]. However, recent research found that machine learning or "abnormality detection" technologies are becoming a dominant and more successful method of combating Android malware [6]. Wi Excepttatic analysis, which entails manually inspecting the Android Manifest.xml file, and Dalvik byte code, source files and source files, which entails working an app in a meticulous setting to observe its behaviour, this behaviour Learning entails learning overall patterns and rules from legitimate and malignant app specimens, and then enabling data-driven prognostications of decisions, like classification [7]. Static characteristics retrieved from an applications are applications machine learning approaches [8]. The static elements of an Android application serve as the foundation for machine learning methodologies, and these static characteristics are meticulously obtained through decrypting. Machine learning approaches have been widely used to classify programmes, with a particular focus on detecting generic malware.

Manuscript received July 5, 2023
Manuscript revised July 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.7.23

As a result, approaches for detecting these errors, in totalling to malware recognition techniques, are critical [9]. The analysis of Android Application Package (APKs) to generate an appropriate set of structures, training machines and the deep learning (DL) algorithms on the resulting features to recognise malicious APK recognize two primary aspects of malware recognition using the machine learning [10]. As a result, an overview of the various APK analysis methodologies is given, including static, dynamic, and hybrid analysis. Vulnerability identification in computer code, like malware detection, includes two major phases: feature creation through static analysis and the training machine learning on derived structures to detect susceptible code segments. As a result, these 2 features are incorporated in the taxonomy of the review [11].The unrivalled threat of Android malware is at the root of a host of internet security problems, and it's an ongoing challenge for researchers and cyber security experts [12]. The only way to eradicate this threat is to detect and kill malware samples as soon as possible. The key to doing so is having a basic awareness of the many types and categories of Android malware. Finally, the report offers some mitigation and preventive measures for Android malware. They detect potentially harmful applications and alert the user, as well as take efforts to eradicate that affection. Antivirus detection rates have risen in lockstep with the threat level.

2. LITERATURE REVIEW

Oluwakemi Christiana Abikoye et al. discussed about the Android Operating System has been widely adopted by a variety of developers. This dynamic has resulted in an exponential growth of Anid-based smartphones across many areas of the business. Even though this development has resulted in significant technological advances and the ease of conducting business (e-commerce) and interpersonal relationships, they have also become powerful platforms for unchecked cyber-attacks and covert operations against company infrastructures and individual people of these portable devices. Various cyber-attack approaches exist, however, malicious application assaults have surpassed previous attack methods such as social engineering. Android malware has grown in sophistication and awareness to the extent where it is now very different from traditional detection methods, notably signature depend systems. The machine learning technologies have developed as a more competent choice for contending new Android malware's uniqueness and complexity. Machine learning (ML) prototypes work by first learning current malware behaviours and then utilising that information to discriminate or recognise any behaviour similar in novel threats. As reported in recent research, this study presented a complete evaluation of machine learning algorithms and their applications in the Android malware detection [13].

Janaka Senanayake et al. explained about a malware assault are increasing as the usage of mobile devices grows, particularly on the Android phones, in which accounting for 72.2 percent of the overall market share. The hackers use a variety of tactics to target cell phones, including credential theft, spying, and the malicious advertising. The (ML) based approaches have shown to be useful in detecting these assaults, since they may construct a classifier from the sets of training instances, removing the requirement for an unambiguous specification of the signatures for developing malware indicators. This paper presents a comprehensive overview of machine learning depend android malware detection method. It assesses 106 carefully chosen articles and identifies their fortes and flaws, and areas for growth. So finally, the ML-based approaches for perceiving source codes susceptibilities are described, because adding security after the program has been released may be more challenging. As a result, the goal of this study is to help academics get a deeper understanding of the topic and suggest prospective future research and development possibilities[11].

Talal A. Abdullah et al. explained about the number of Android-based mobile devices on the market is growing, these devices are becoming the prime targets for malicious software. Several Android malware programs have been developed in recent years to execute various unauthorised and dangerous operations on mobile devices. To identify such Android malicious applications, specialized tools and the anti-virus agendas employed traditional signature-based methodologies. The most current Android malware applications, such as the zero-day, however, are not detectable using traditional approaches that rely on fixed signatures or IDs. As a result of their capacity to study and use prevailing knowledge to detect new Android malware applications, the most lately released research papers have proposed machine learning methods as an alternate strategy to detect the Android malware. This article covers the fundamentals of malware, Android architecture, and permission aspects that may be used as malware predictors[14].

More importantly, this article empirically examines the different the achievement of five learning algorithms usually used in the literature for the detecting malware apps: K-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Data Bus Network (DBN), Naive Bayes (NB) and Gated Recurrent Units (GRU).

Research Question:

- What is the requirement of Android malware detection?
- How ML technique help to detect Android Malware Recognition?

•

3. METHODOLOGY

On Android, the malware detection may be accomplished in two ways: identification based on signatures and identification based on behaviour [15]. The signature-based exposure technique is simple, effective, and produces minimal fabricated positives. The digital signal of the programme is analysed to patterns in a known malware collection. This technique, on the other hand, excludes the identification of ransom-ware. As a result, the anomaly-based detection technique is the most often utilized. On Android, malware detection may be accomplished in two types: signature-based detection and behaviour detection [16]. The system's binary code is examined to patterns in a list of known malware. However, this method does not permit for the identification of the unknown malware. The most often used detection approach is behaviour-based/ the anomaly detection.

3.1. Design:

Both dynamic and static analysis approaches converge successfully in hybrid analysis. The suggested approach retrieves the dynamic and static characteristics of the Android application. The methodology design focused on permission and application programming interface (API) characteristics of the Android application in the static detection, then filtered features depending on the weight derived by the term frequency-inverse document frequency (TF-IDF) approach. Each feature in static detection is assigned a weight based on the Naive Bayes method. They also presented experimental results from three ensemble algorithms for dynamic detection, indicating that the XGBoost algorithm beat the others. Finally, they demonstrated that their technique has a detection accuracy of 94.6 percent for 8000 applications (4000 benign apps and 4000 malicious apps), which is higher than the static detection accuracy of 85.3 percent and the dynamic detection accuracy of 94.1 percent for the same number of applications.

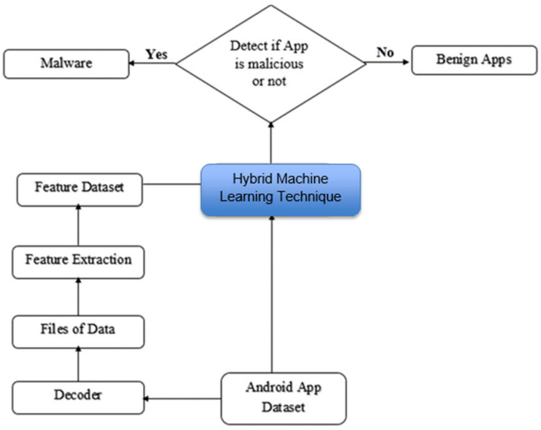

Figure 1: The malware is detected by applying an algorithm such as a hybrid machines learning algorithm.

As shown in given Figure 1, the virus is initially discovered by using an algorithm such as a hybrid machines learning technique and delivering the application's data set. A specific sort of data decoder is then used to decode the data collection. Following the decoding of the files, the characteristics are taken from the data and stored in a dataset over which the appropriate algorithm will be run to identify the virus.

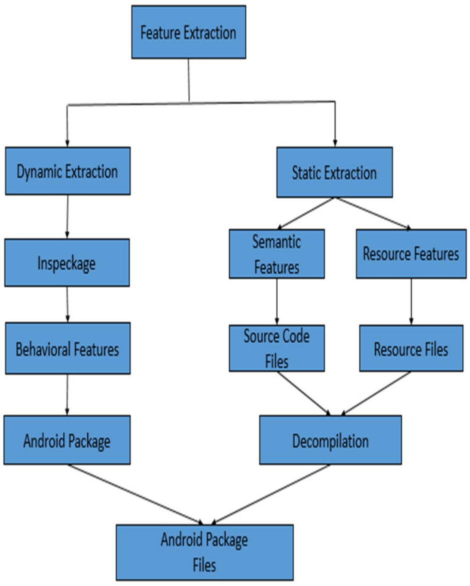

Figure 2: This shows the feature extraction of the dynamic and static extraction.

Figure 2 depicts the features extraction strategy used for Android malware, which combines dynamic and static analysis. Decompiling the APK files yields static characteristics such as resource features and semantic features. Through one-hot in encoding, the static structures form a binary feature vectors. The dynamic structures are retrieved by observing the associated API calls while the APK is executing. The entity implanting approach is required to build features of vector for dynamic characteristics connected with time series.

3.2. Instrument:

3.2.1. Features Extraction:

Static characteristics although extraction is debauched and uses slight system assets, making it suited for the large-scale feature extractions, it cannot identify obfuscated of Android malware. This study employs the hybrid detection approach combining dynamic and static analysis a whole of 353 features were retrieved, including 293 static and 54 dynamic features.

3.2.1.1. Static Feature Extraction:

The resource characteristics and semantic structures are among the retrieved static characteristics.

3.2.1.1.1. Resource Features:

The term "resource features" refers to features retrieved from APK resource files. The APK's inconsistency in structure and logic serve as the basic foundation for obtaining resource attributes. Inconsistent alludes to the artefacts left behind because of hidden hazardous components, culminating in an APK file with an abnormal architecture. When malicious software is bundled as a benevolent application, it generally leaves evidence behind, which is referred to as contradictory logic.

3.2.1.1.2. Semantic Features:

The semantic characteristics of the APK code file are extracted. Semantic characteristics include common static feature such the penetrating API and authorization. Author suggest several modern semantic characteristics, such as express purpose and other meta-data-mined features.

3.2.1.2. Dynamic Features Extraction:

Data decryption and encryption, file writing and reading, network data transfer, SMS, call, geographic position, and entrée to sensitive informations are examples of dynamic features. These behaviours might indicate the functions and goals of the application. In packages a simple application program that includes regularly used dynamic analytic methods, and built-in the web server may give users with nice interactive interface. In spin package not only collect fundamental information like permissions, shared libraries, components, User Identifier (UID), and so on, but it can also watch the application's activity in real-time can alter the bent API, which means it can adapt dynamic behaviour necessary for observing, which is also the tool's main benefit.

3.2.2. Features Encoding:

3.2.2.1. Static Features Encoding:

The majority of static characteristics are binary, with just a limited number of discrete features, and there is no link between them. As a result, the deep learning algorithms DBN is well suited to static features. Because the DBN's input is the binary vector, the static characteristics are encoded into binary vectors using one-hot encoding. The method of one-hot encoding involves converting discrete characteristics into binary structures. The discrete numbers 0, 1, 2, 3 are encoded to binary orders 0001, 0010, 0100, 1000, for example. If one-hot encoding is used for discrete structures with a wide value ranges, the features vector will be scant. In this situation, the value ranges can be finely categorized after one-hot encoding to lower the feature dimension. Following one-hot encoding, entirely static characteristics are concatenated hooked on the binary vector, in which serves as the DBN's inputs.

3.2.2.2. Dynamic Feature Encoding:

After collecting the dynamic characteristics of an Android applications, the dynamic structures are molded into a chronological operating order. Since the dynamic characteristics are associated in with time, the neural network that can better suited for the dynamic behaviour of the Android applications software. The GRU network's input is a dynamic feature vector. Entity implanting is the data demonstration technique. It encodes organised discrete variable and attempts to keep the continuous link between data in the data representation.

3.3. The Requirements of Analysis:

3.3.1. Functional Necessities:

The system generates malware detection for a given group of APK datasets. The detection's output must meet the following conditions:

- Depending on the input data, the outputs ought to be possible to perceive malware.
- The technology must be opoptimizedor the complexity of time and space.
- The system will be able to notice new/ unknown malware variants.
- The system will be able to know the outcome of harmful programs based on their previous behaviour
- The system should be able to choose the appropriate feature set to include unknown behaviour stem should be capable of detecting malware from any domain.

3.3.2. Non- Functional Necessities:

3.3.2.1. User interface:

There has to be a convenient and straightforward solution that allows the user to detect viruses using the APK dataset. The model is constructed using deep learning using the chosen features as input. A comparison analysis is performed, and a classification report is produced.

3.3.2.2. Hardware:

The system's effective implementation does not necessitate the use of any unique hardware interfaces.

3.3.2.3. Software:

- Tools: Android Studio, Anaconda3

- Programming Languages: Java, Python
- Dataset: APK dataset
- Operating System: Linux/Windows/ Macintosh

The detection technique or classifier is used to identify whether or not a programs malware. The classifier determines if software is a malware or benign ware based on its attributes. Machine learning is required by the popular of classifiers. Machine learning classifiers employ one or more classifiers. In the detecting procedure, layered classifiers may also be utilized. In this example, there are 2 or 3 levels, with every layer including identification to increase the detection system's accuracy. There are numerous individual classifiers in parallel classifiers. As indicated in the figure below, the outputs of various classifiers are merged to achieve improved accuracy. Another classifiers, like analytic hierarchy process (AHP) and punishment computation, do not employ machine learning. Figure 3 depicts collective machine learning classifier used in the Android mobile malware exposure. The deep learning models based on a mix of the deep belief networks and the gates recurrent units is presented based on the varied characteristics of the dynamic and static properties of Android applications.

The advantage of employing the DBN is that static structures of the Android applications learn earlier and perform better. GRU outperforms typical machine learning models in coping with lengthier time operations sequences, with rarer parameters, quicker training speeds, and fewer data necessary to obtain a satisfactory generalisation effect. As a result, the GRU neural networks is more suited to handling the dynamic elements of Android applications. Figure 2 depicts the DBN-GRU hybrid models for the Android malware detection. The dynamic and static features vectors are utilized to sequence the DBN-GRU, individually, and the outputs vectors are fed into the completely connected layers. So, the softmax functions transfers several neurons' outputs to the intermission (0, 1) and produces classification outputs in the procedure of possibility. The SVM algorithm is parts of back propagation algorithm that is charity to fine-tune the DBN and GRU variables. Figure 4 depicts a hybrid feature that may be created using DBN, KNN-SVM, and GRU.

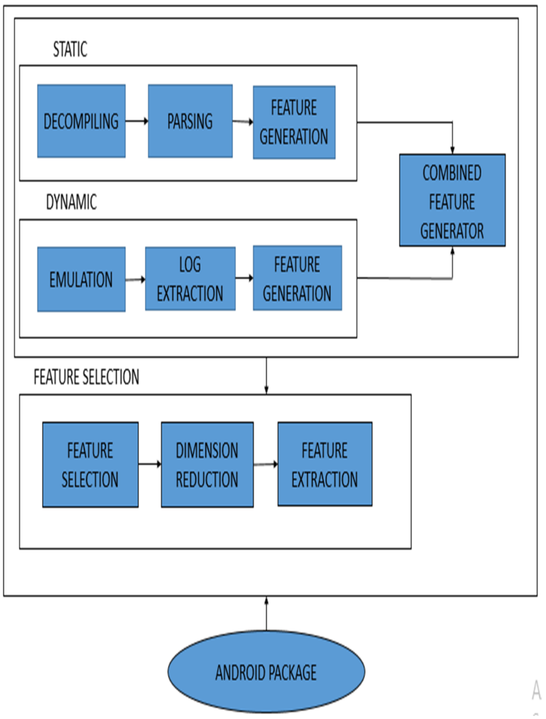

Figure 3: The above diagram shows the android package which helps to select the feature for the static and dynamic state.

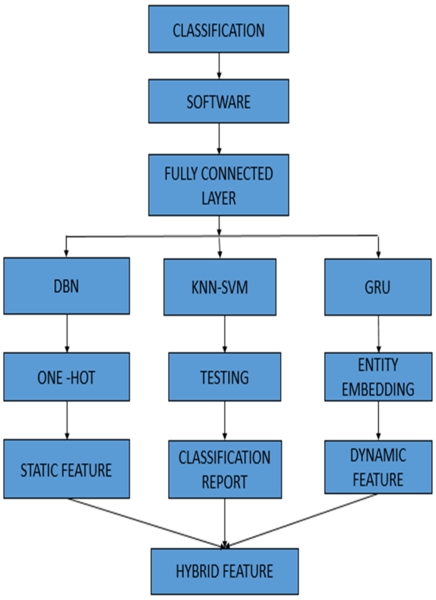

Figure 4: The above diagram shows the hybrid feature which can be made with the help of DBN, KNN-SVM, and GRU.

4. RESULT AND DISCUSSION

Static features are those that can be extracted without operating the program. A static feature extractor is a technique that extracts the static features. Some studies concentrate on a single category of static properties, while others examine a large number of them. Permissions, API calls, extracted strings, XML components, native commands, meta data, intents, usage scenarios, components, and other static properties are all prevalent. To extract static features, you can utilise the document, dex file, or byte code. The APK tool is the most extensively used. Researchers get the APK file, Smali file, classes.dex and Manifest file from the APK tool. Table 1 shows the type, sources, and number of models in dataset. A total of 7,000 benign samples were obtained through web crawler from the Google Play and the APK mobile applications stores. The malware collection contains 6,298 samples, all of which were obtained from communal malware distribution websites. The malware samples dataset is split into 2 parts based as to whether the trials use misdirection technology: each part is the nonobfuscated malware set of data available for download from VirusShare, as well as the other parts is the obscured malware dataset available for download from the PRAGuard acquired by distorting the Contagion Minimum and MalGenome sets of data with five different evasion techniques.

The most often utilised aspects of obfuscated malware sample and the nonobfuscated malware sample are compared. Figures 5 and Figure 6 depict the top ten commonly utilised attributes of the two types of samples. Permission-related structures (such as Write SMS, Read SMS, and so on) of both example kinds are widely employed since permission structures are hard to conceal, and obfuscating authorisation features destroys the basic edifice of APK. However, several sensitive API structures (such as Telephony manager get devised, etc.) are commonly utilised in non-obfuscated malware sample but extremely seldom in the obfuscated malware sample, indicating that malware sample after obfuscation might evade associated detections when accessing sensitive APIs. It checks if the certificate's creation time and the time it's used to sign the APK are the same. This feature appears often, suggesting that automated repackaging is used to build the bulk of obfuscated malware variants.

Table 1: This table shows the benign and android malware type.

Type	Originator	Number	Total
Malware	Praguard	4500	7360
	Virus	2860	
Benign	Android package	5300	8300
	Google Play Apps	3000	

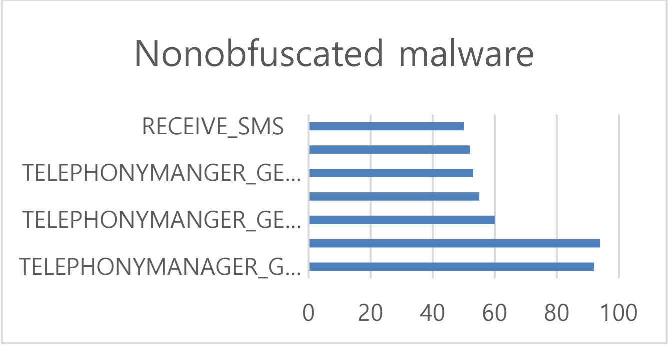

Figure 5: The above graph shows the nonobfuscated malware feature.

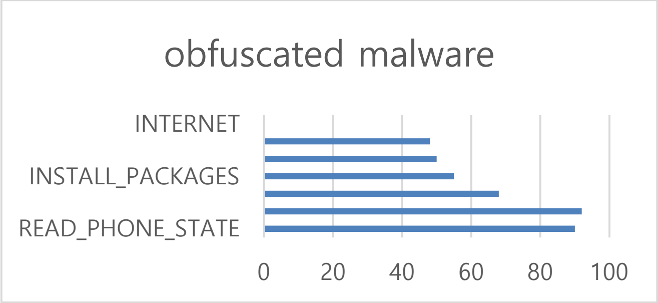

Figure 6: The above graph shows the obfuscated malware feature.

The Hybrid Deep Learning Model's (HDLM) Detection Effects Table 2 shows the results of evaluating the detection impact of the (HDLM) and (DBN-GRU) on Android malware using the indicators of recall, precision, and accuracy. Standard machine learning models (such as SVM, Nave Bayes, and KNN) are much worse than deep learning models and Figure 7 shown the accuracy after applying the machine learning methods for detecting the malware in android.

3.4. Evaluation Parameters:

The True Positive Rate (TPR) defines the percentage of benign apps identified accurately, where *i. TPR = TP (TP+FN)*

TP is the number of properly recognised benign applications, whereas FN denotes the amount of erroneously identified kind apps. The False Positive Rate (FPR) is the percentage of malware programmes that are wrongly identified, where

3.4.1. FPR = FP/(TN+FP)

The numbers of mistakenly identified malware is denoted by FP, while the number of appropriately identified malware is denoted by TN. Accuracy (ACC) is a performance statistic that is used to quantify overall performance. The percentage of accurately identified apps is referred to as accuracy, where

3.4.2. ACC=(TP+TN)/(TP+TN+FP+FN)

Table 2: Detecting the effects of the different machine learning algorithm.

	MALWARE SAMPLE		**BENING SAMPLES**		**EXACTNESS**
	ACCURACY	**RECALL**	**ACCURACY**	**RECALL**	
NAÏVE BAYES	87.60	87.90	87.01	80.10	85.10
KNN	86.93	84.59	85.50	87.50	87.20
SVM	94.80	94.87	93.40	89.90	91.40
DBN-GRU	98.80	98.70	98.84	97.40	97.90

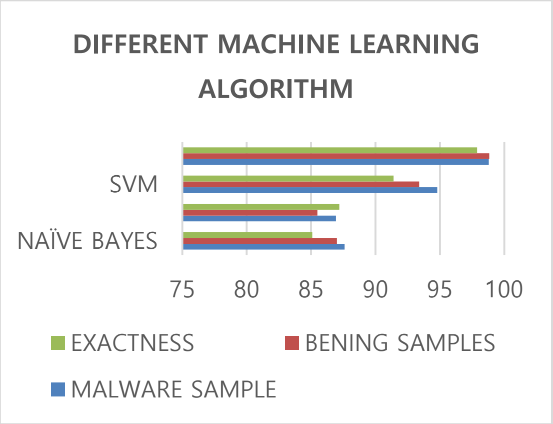

Figure 7: The above graph shows the different accuracy with different machine learning approaches.

5. CONCLUSION

To identify mobile malware and malicious activity, researchers employed static, dynamic, and hybrid approaches. The key interest of researchers is accuracy levels, and the majority of research articles use accuracy measures to explain the efficacy of their detection technique. Performance overhead should be addressed for mobile device operating systems, since better precision may result in more overhead. To make the detection procedure efficient, accuracy and performance overhead must be effectively matched. The static characteristic is created by examining the sample's formats, then collecting the hash value, string data, function data, header file data, and metadata data. However, when stationary camera is packed, encoded, or compacted, static characteristics are unable to effectively identify malware, making it difficult for static characteristics to communicate the genuine intent of malware, lowering detection rate. The behaviour of the sample operation and the characteristics of the debugging record, such as file actions, process formation and destruction, as well as other dynamic behaviours, are examples of dynamic characteristics. The extracted dynamic characteristics give a more accurate characterization than the static characteristics since the dangerous activities of malware during dynamic runtime cannot be hidden. However, dynamic extracted features must be performed in a virtual world that will be restored and returned to its previous condition after each harmful sample is evaluated, ensuring that the virtual world is a legitimate user scenario. As a result, the efficiency of extracting features is substantially lower than that for classifiers. Researcher look at the current research projects in 3 groups: dynamic, static, and hybrid analysis.

The data collection, features, characteristic selection process, detection method, and accuracy are all represented in these research. Authors also discussed the gap in the literature and the limits of present studies. As a result, author were able to identify the suspicious feature lists that malware authors frequently utilise. For Android malware detection, this research integrates dynamic and static analytic technologies and constructs a hybrid deep learning method depend on GRU, DBN, KNN and SVM. To contract with concealment technology, modern stable features with stout antiobfuscation abilities have been introduced, and dynamic characteristics of the software applications at the given runtime have been removed to expand the Android malware set of structures. A hybrid deep learning model containing DBN, SVM, GRU, and KNN is utilised for learning based on the varied characteristics of dynamic and static data, and the detection impact of this model is proven by comparison tests. There are various research questions that need to be answered in the future.

REFERENCES

[1] I. Martín, J. A. Hernández, A. Muñoz, and A. Guzmán, "Android Malware Characterization Using Metadata and Machine Learning Techniques," *Secur. Commun. Networks*, 2018, doi: 10.1155/2018/5749481.

[2] S. Priyadharshini and S. Shanthi, "A Survey on Detecting Android Malware Using Machine Learning Technique," 2021, doi: 10.1109/ICACCS51430.2021.9441712.

[3] 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 Trans. Emerg. Top. Comput. Intell.*, 2020, doi: 10.1109/TETCI.2019.2910243.

[4] J. Garcia, M. Hammad, and S. Malek, "Lightweight, obfuscation-Resilient detection and family identification of android malware," *ACM Trans. Softw. Eng. Methodol.*, 2018, doi: 10.1145/3162625.

[5] S. Gupta, S. Sethi, S. Chaudhary, and A. Arora, "Blockchain Based Detection of Android Malware using Ranked Permissions," *Int. J. Eng. Adv. Technol.*, 2021, doi: 10.35940/ijeat.e2593.0610521.

[6] M. Melis *et al.*, "Do gradient-based explanations tell anything about adversarial robustness to android malware?," *Int. J. Mach. Learn. Cybern.*, 2021, doi: 10.1007/s13042-021-01393-7.

[7] S. Y. Yerima, S. Sezer, and I. Muttik, "High accuracy android malware detection using ensemble learning," *IET Inf. Secur.*, 2015, doi: 10.1049/iet-ifs.2014.0099.

[8] H. Zhang, D. Yao, and N. Ramakrishnan, "Causality-based sensemaking of network traffic for android application security," 2016, doi: 10.1145/2996758.2996760.

[9] G. Canfora, F. Mercaldo, E. Medvet, and C. A. Visaggio, "Detecting Android malware using sequences of system calls," 2015, doi: 10.1145/2804345.2804349.

[10] S. I. Imtiaz, S. ur Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network," *Futur. Gener. Comput. Syst.*, 2021, doi: 10.1016/j.future.2020.10.008.

[11] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics (Switzerland)*. 2021, doi: 10.3390/electronics10131606.

[12] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A survey on android malware detection techniques using machine learning Algorithms," 2019, doi: 10.1109/SDS.2019.8768729.

[13] O. C. Abikoye, B. A. Gyunka, and O. N. Akande, "Android malware detection through machine learning techniques: A review," *Int. J. online Biomed. Eng.*, vol. 16, no. 2, pp. 14–30, 2020, doi: 10.3991/ijoe.v16i02.11549.

[14] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, "Empirical study on intelligent android malware detection based on supervised machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 4, pp. 215–224, 2020, doi: 10.14569/IJACSA.2020.0110429.

[15] F. Mercaldo and A. Santone, "Formal Equivalence Checking for Mobile Malware Detection and Family Classification," *IEEE Trans. Softw. Eng.*, 2021, doi: 10.1109/TSE.2021.3067061.

[16] S. M. Shahidi, H. Shakeri, and M. Jalali, "A semantic malware detection model based on the GMDH neural networks," *Comput. Electr. Eng.*, 2021, doi: 10.1016/j.compeleceng.2021.107099.

Sk Heena Kauser
Research scholor,
Department of Computer Science & Engineering,
Sathyabama Institute of Science and Technology, Chennai, India.

Dr.V.Maria Anu
Professor,
Department of Computer Science & Engineering,
Sathyabama Institute of Science and Technology, Chennai, India.