Sakarya University Journal of Science

ISSN : 2147-835X
Publisher : Sakarya University

Vol. 28, No.3, 517-530, 2024
DOI: https://doi.org/10.16984/saufenbilder.1350839

Research Article

Comparative Analysis of Machine Learning Models for Android Malware Detection

Selma Bulut [1*], Adem Korkmaz[2]

[1]Kirklareli University, Technical Sciences Vocational School, Department of Computer Programming, Kırklareli, Türkiye, selma.bulut@klu.edu.tr
[2]Bandirma University, Gonen Vocational School, Department of Web Design and Coding, Balıkesir, Türkiye, ademkorkmaz@bandirma.edu.tr
[*]Corresponding Author

ARTICLE INFO

ABSTRACT

Keywords:
Android malware detection
Machine learning algorithms
Naticusdroid dataset
Comparative analysis
Data integrity

The rapid growth of Android devices has led to increased security concerns, especially from malicious software. This study extensively compares machine-learning algorithms for effective Android malware detection. Traditional models, such as random forest (RF) and support vector machines (SVM), alongside advanced approaches, such as convolutional neural networks (CNN) and XGBoost, were evaluated. Leveraging the NATICUSdroid dataset containing 29,332 records and 86 traces, the results highlight the superiority of RF with 97.1% and XGBoost with 97.2% accuracy. However, evolving malware and real-world unpredictability require a cautious interpretation. Promising as they are, our findings stress the need for continuous innovation in malware detection to ensure robust Android user security and data integrity.

Article History:
Received: 27.08.2023
Accepted: 15.03.2024
Online Available: 06.06.2024

1. Introduction

Mobile phones, which entered our lives in the 1990s, were initially produced for texting and talking; however, with the advent of evolving technology and mobile internet, they have allowed us to perform all sorts of tasks efficiently. People use smartphones for various activities, from shopping, reading newspapers, and banking transactions to communicating via social media. Therefore, smartphones have become an indispensable part of our lives.

In smartphones, one of the operating systems such as Android, IOS, Samsung, KaIOS, BlackBerryOS, Tizen, and Windows mobile can be found. By the end of 2022, with a 71.75% market share, Android has been recognized as the most widely used operating system. In 2023, it is anticipated that there will be approximately 3.6 billion active Android smartphone users dispersed across 190 nations. Android has attained a 70.94% share of the global mobile operating system market, while Apple's iOS has secured a 28.33% share [1].

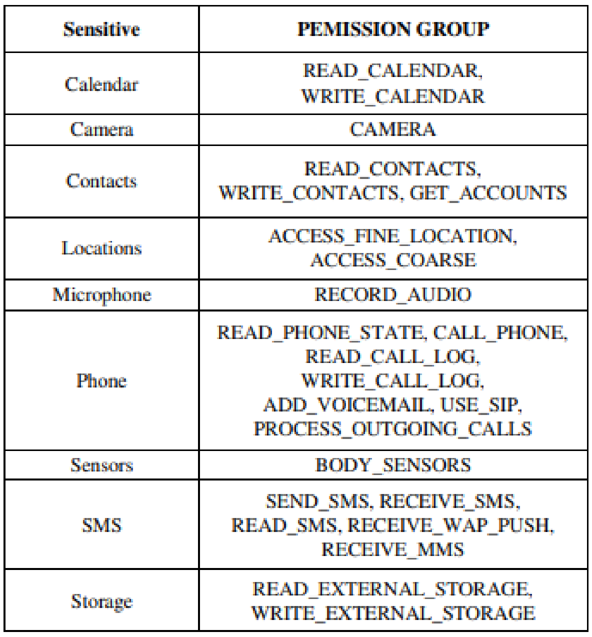
Google developed the Android operating system. It is the fastest-growing, open-source, and fully customizable mobile operating system software in smartphone operating systems. Android extends an open-source platform, offering unrestricted access and managerial control to Original Equipment Manufacturers (OEMs), encompassing entities such as Samsung, Xiaomi, Oppo, Vivo, Huawei, Motorola, and Google. Subsequently, these manufacturers have commercialized their devices at markedly economical price points, particularly when contrasted with the average sale price of Apple iOS devices, quantified at 261 dollars in the fiscal year 2021. This situation reveals the fundamental reason for Android's success today. However, the Android operating system is preferred in smartphones, wearable devices, and smart TVs [2, 3].

With the increase in the number of smartphone applications downloaded and used through app stores (Google Play, App Store), security issues have emerged as a problem. Malware that we encounter on computers is now taking over our smartphones. Malicious software that permeates smartphones can precipitate a spectrum of harmful consequences, including unauthorized access to users' personal information, surveillance of user activities and geographical locations, unauthorized intrusion into social media accounts, penetration into banking accounts, dissemination of unauthorized messages, and diminution of memory and battery longevity [4]. The rapid increase in Android applications and being the most preferred operating system has made it a target for malicious software.

According to the Kaspersky Security Network, 4,948,522 mobile malicious software, adware, and risky software attacks were prevented in the first quarter of 2023. Advertising software is the most common threat to mobile devices, accounting for 34.8% of all detected threats [5]. In recent years, the Android operating system (AOS) has released several updates to address various security vulnerabilities [6].

The primary protection mechanism in the Android operating system is Google Play Protect, which identifies malicious software applications in the Google Play Store. However, there are many third-party app stores where malicious software applications can be downloaded. Another security element is a permission-based resource access system that prevents applications from unauthorized access to resources such as cameras, microphones, and internal file storage [7].
The AOS's Android Market Security Model operates similarly to the Linux security model. In this model, permissions granted to files are user-based. A user cannot read, modify, and execute another user's file unless that user gives permission. When applications run, they must request permission from the user once during installation based on the resources they will use and the areas they will access. Permissions are defined within the AndroidManifest.xml file in the APK (Android application package) [4].

Table 1. Android Sensitive Permissions [8].



Sensitive	PEMISSION GROUP
Calendar	READ_CALENDAR, WRITE_CALENDAR
Camera	CAMERA
Contacts	READ_CONTACTS, WRITE_CONTACTS, GET_ACCOUNTS
Locations	ACCESS_FINE_LOCATION, ACCESS_COARSE
Microphone	RECORD_AUDIO
Phone	READ_PHONE_STATE, CALL_PHONE, READ_CALL_LOG, WRITE_CALL_LOG, ADD_VOICEMAIL, USE_SIP, PROCESS_OUTGOING_CALLS
Sensors	BODY_SENSORS
SMS	SEND_SMS, RECEIVE_SMS, READ_SMS, RECEIVE_WAP_PUSH, RECEIVE_MMS
Storage	READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE

Table 1 displays the list of these sensitive permissions. These permissions are divided into four protection levels: i) Normal, ii) Dangerous, iii) Signature, and iv) Signature|Privileged [9] Permissions classified as 'Dangerous' are the most sensitive among these, as they manage users' personal information, and when used with malicious intent, they can jeopardize users' security and privacy. Therefore, user consent is sought for these permissions [7]. For instance, SEND_SMS permission is essential for communication and social media apps that allow text messages to be sent.

Malicious software can use this permission to communicate with their centers or send messages to premium numbers. This can lead to unexpected billing charges. While granting two permissions separately might be harmless, giving them simultaneously can significantly increase privacy and security risks. For example, while the INTERNET and READ SMS permissions are benign when taken separately, they can be used for an app that can read your text messages and send them to a third party [10].

Malicious software such as Trojans, ransomware, spyware, and worms exploit users unfamiliar with Android's permission system, jeopardizing their data. Hence, there is a need to educate users about Android permissions. In the AOS security model, the person installing the app must consciously grant these permissions. Third-generation app developers sometimes use these permissions either knowingly or unwittingly. The primary responsibility lies with the user, the person installing the app. One must decide if the requested permission is necessary for the application and grant permissions accordingly.

The principal objective of this study is to meticulously evaluate the efficacy of various machine learning algorithms in the context of Android permissions analysis, focusing on detecting potential security threats posed by malicious applications. Given the pervasive nature of mobile devices in daily life and the consequent escalation of security risks, the research endeavors to scrutinize and compare the predictive capabilities of a broad spectrum of models—including Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), Random Forest (RF), k-Nearest Neighbors (k-NN),

Support Vector Machines (SVM), CatBoostClassifier, and XGBoost. This comparative analysis aims to identify the most effective algorithms in terms of accuracy, F1 score, and computational efficiency and contribute to the development of robust, scalable solutions for enhancing the security of Android operating systems. Through a comprehensive assessment of the NATICUSdroid dataset, encompassing 29,332 records across 86 permissions, this study seeks to advance our understanding of how machine learning techniques can be leveraged to fortify defenses against the ever-evolving landscape of mobile malware threats, thereby providing invaluable insights for both academic research and practical applications in cybersecurity.

2. Related Work

Machine learning-based Android malicious software detection studies are categorized into static and dynamic analyses. The static analysis includes notifications, permissions, API calls, and intents. It can be obtained without running the malicious software. On the other hand, dynamic analysis focuses on monitoring an application's activity, such as logcat errors, shared memory corruption, system calls, and processes [11]. Dynamic analysis can be obtained by running the malicious software. The most commonly used method is static analysis, but it lacks accuracy. Dynamic analysis is more effective but requires a virtual environment or an Android device [12]. Hybrid analysis combines static and dynamic features [13].

Various datasets for malicious software analysis have become available in recent years. Using these datasets, researchers have tried all supervised, unsupervised, and deep learning strategies to detect Android malicious software [14]. The datasets under scrutiny may encapsulate static attributes, such as Application Programming Interface (API) invocations, intentional actions, permission requests, and dynamic characteristics, including logcat error manifestations, shared memory allocations, and system call interactions. Studies conducted for this purpose have been examined.

Y. Zhou and X. Jiang [15] conducted a significant study examining the characterization and evolution of Android malicious software. This research investigated how specific permissions and behaviors could be used to detect malicious applications by analyzing malware behaviors. S. Y. Yerima and S. Khan [16] employed static attributes, encompassing permissions, intents, API invocations, and instantiation dates, extracted from benign and malicious software datasets furnished with date labels to scrutinize the efficacy of assorted machine-learning classifiers. They preferred machine learning methodologies such as NB, SVM, and RF.

A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. Gagnon, and F. Massicotte [17] proffered a methodology entailing an image-based deep neural network to systematically classify and characterize software exemplars, derived from a malware dataset that encompasses 12 paramount malware categories and 191 noteworthy malicious software entities.

J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi [18] propounded MAPAS, a malware detection system that furnishes elevated accuracy while enabling adaptable deployment of computational resources. MAPAS scrutinizes the behavioral attributes of malevolent applications by analyzing API call graphs, employing convolutional neural networks (CNN). The authors juxtaposed MAPAS with an alternative detection methodology, MaMaDroid, to evaluate its relative performance and efficacy. F. Giannakas, V. Kouliaridis, and G. Kambourakis [19] employed shallow and deep machine learning techniques to predict malicious software on the Android platform. This involved researching, optimizing, and evaluating the performance of 28 different machine learning algorithms, including a DNN model.

K. Liu, G. Zhang, X. Chen, Q. Liu, L. Peng, and L. Yurui [10] summarized the process of sample collection, data preprocessing, feature selection, machine learning models, algorithms, and evaluation of detection efficiency using machine learning. They also assessed future expectations for research based on machine learning detection of malicious Android software. C. D. Nguyen, N. H. Khoa, K. N. D. Doan, and N. T. Cam [20] instantiated machine learning and deep learning algorithms to categorize malware into respective families and categories, leveraging multiple datasets. The researchers conducted a comprehensive evaluation and elected appropriate methodologies, ensuring optimal alignment with each dataset.

C. Ding, N. Luktarhan, B. Lu, and W. Zhang [21] applied deep learning techniques, formulating a classification schema predicated upon permission and intent features discerned through static and network traffic attributes identified through dynamic analysis. S. Shi, S. Tian, B. Wang, T. Zhou, and G. Chen [13] introduced SFCGDroid, a malware detection method that uses precise function call graphs to identify malicious behaviors. SFCGDroid utilizes both static and dynamic features to identify malicious activities. The process achieved high accuracy and F1 scores on a broad dataset of Android software.

R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud [11] executed a classification of Android malicious software, employing an optimal feature selection methodology in tandem with an ensemble machine learning approach, aiming to enhance the precision and reliability of the categorization process.

In the research conducted by M. N. U. R. Chowdhury, A. Haque, H. Soliman, M. S. Hossen, T. Fatima, I. Ahmed [22], an examination of various machine learning approaches—spanning supervised, unsupervised, and deep learning paradigms—utilized for Android malware detection was undertaken. Moreover, a comparative analysis of the performance of assorted Android malware detection methodologies was proffered, and the evaluative metrics employed to gauge their efficacy were explored in the discourse. Conclusively, the discourse also illuminated the detriments and challenges inherent to contemporary methodologies.

H. Rathore, S. Chari, N. Verma, S. K. Sahay, and M. Sewak [23] elucidate a comprehensive investigation predicated on data mining techniques for static malware detection. The authors proffer an exhaustive analysis of each phase inherent to data mining-based malware detection, including data aggregation, preprocessing, feature extraction, application of learning algorithms, and evaluative procedures, while also dialoguing upon the evolution of Android malware and extant detection techniques.

3. Material and Method

3.1. NATICUSdroid (Android Permissions) Dataset

The dataset is in binary format and describes the permissions each application may be using. Each row represents an app, and each column represents a specific Android permission. A '1' in a cell indicates that the corresponding application uses this permission, while a '0' does not. The data set contains 29332 records of 86 permissions on Android phones, such as android.permission.CAMERA, and android.permission.READ_CONTACTS [24].

Figure 1 shows the Android Permission Classification attribute data distribution.

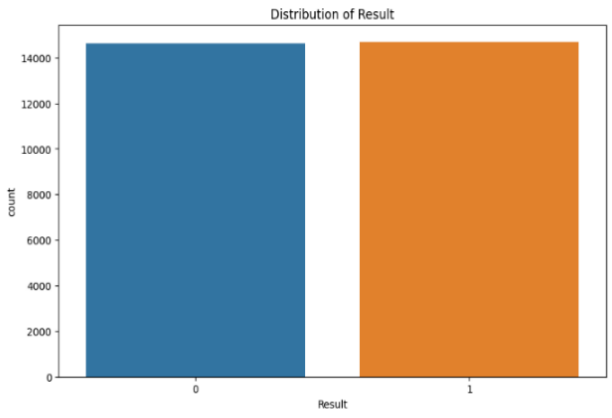


Figure 1. Android Permissions Classification attribute data distribution

3.2. Data analysis

Before initiating the study, an examination of the data for preprocessing purposes revealed that data balancing measures were deemed unnecessary due to the relatively even distribution of the Android Permission Classification attribute, which comprised 14,700 records for class 1 and 14,632 for class 0. Given the unconditional nature of the Android permission attributes, represented as 1 and 0, normalization procedures were not required. Furthermore, the dataset was found to be complete, with no instances of missing or incomplete data. This thorough data assessment ensured the dataset was adequately prepared for the subsequent analysis without additional preprocessing steps such as balancing or normalization.

In the study, an extensive classification task was conducted utilizing a myriad of algorithms, including Convolutional Neural Networks (CNN), Artificial Neural Network (ANN), Random Forest (RF), k-nearest Neighbors (k-NN), Support Vector Machines (SVM), CatBoostClassifier, and XGBoost. Each algorithm, ranging from the spatial hierarchy-utilizing CNNs to the gradient-boosted precision of XGBoost, was rigorously trained and subsequently evaluated based on standard classification metrics like accuracy, F1 score, and AUC-ROC. A comprehensive comparative analysis spotlighted the superior performers, considering varied facets such as training duration, model interpretability, and predictive prowess. The culmination of the study offered invaluable insights, underscoring the most efficacious algorithms and proffering recommendations for practical deployments or prospective research avenues. The sequential process followed in the research study is shown in Figure 2.

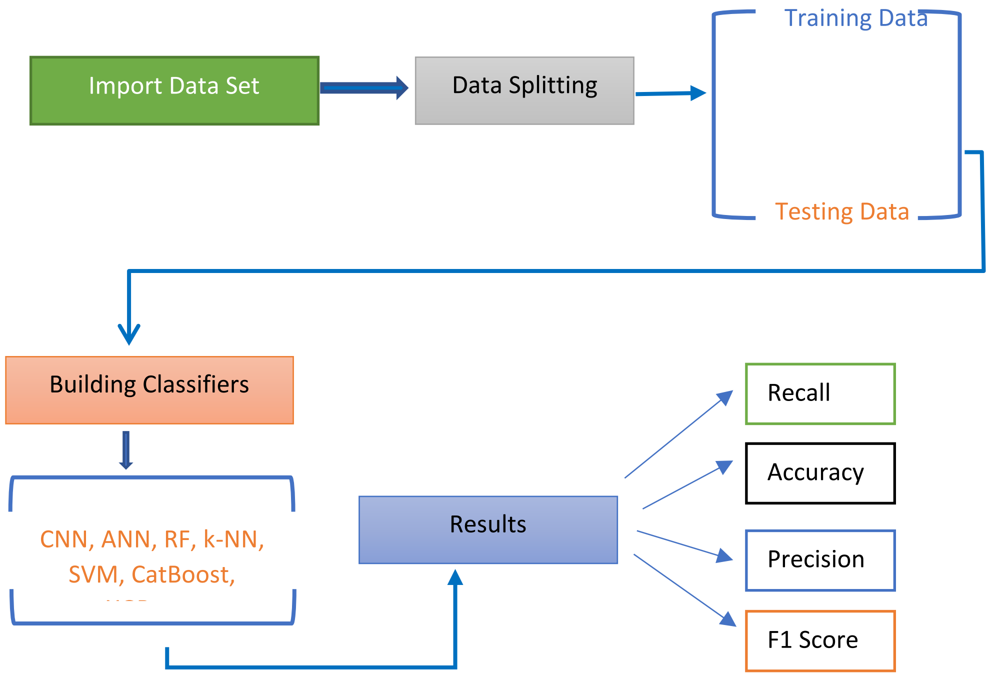


Figure 2. Research methodology steps

3.2.1. Artificial neural networks

Artificial Neural Networks (ANNs) represent a category of machine learning models conceptualized and developed by drawing inspiration from biological neural networks' architectural structure and functional dynamics. ANNs are composed of interlinked artificial neurons, which are systematically arranged in layers and responsible for processing and transmitting information via connections that are weighted and adjusted during the learning process [25]. The artificial neuron mimics a biological neuron's input, processing, and output properties. Figure 3 shows the results produced by the network: The net input obtained by multiplying the information entered into the network by its weights (W) is processed with the transfer function and taken from the output layer [26-28].

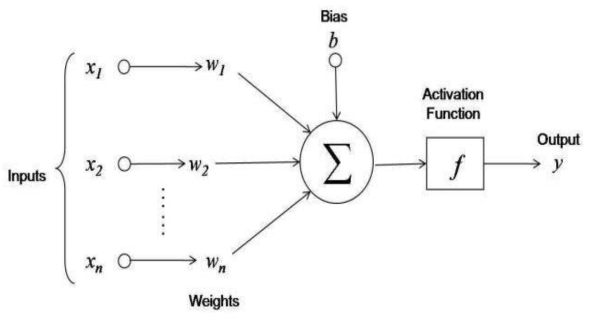


Figure 3. Artificial neuron network structure [29]

Classification with artificial neural networks (ANNs) involves training a neural network model to classify data into different classes or categories. ANNs have been ubiquitously utilized for classification endeavors, attributed to their capacity to decipher intricate patterns and interrelations within data. The network comprises an input layer, an intermediate hidden layer hosting ten neurons, a subsequent hidden layer furnished with five neurons, and an output layer endowed with a neuron count equivalent to the number of classes inherent in the classification task. About the activation functions, the hidden layers employ the Rectified Linear Unit (ReLU), whereas the sigmoid activation function is implemented in the output layer.

In case there is no linear relationship between the inputs and outputs of artificial neural networks, Multilayer Perceptrons (Multilayer Perceptron) are used to learn. Therefore, this method was used in the study.

3.2.2. Convolutional neural networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized category of deep neural networks predominantly applied to computer vision tasks, designed explicitly to process grid-like topology data, such as images [30]. These networks use convolutional layers with filters or kernels that traverse the input data, producing a feature map that emphasizes critical features, making them particularly adept at identifying local patterns ranging from simple edges to complex image structures [31]. A significant component, the pooling layer, typically follows the convolutional layer, aiming to down-sample the spatial dimensions of the data, enhancing the model's robustness and reducing computational demands [32]. As one progresses more profoundly into a CNN, the architecture detects intricate structures, with the concluding fully connected layers classifying the discerned high-level features into categories. Their hierarchical design, enabling the adaptive learning of spatial hierarchies from input images, has solidified CNNs as a premier choice for numerous computer vision applications.

3.2.3. Random forest (RF)

The Random Forest (RF) algorithm efficiently amalgamates multiple randomized decision trees, producing results by averaging their predictions. Particularly powerful when variables outnumber observations, each tree within the RF is trained on a random data subset. Only a randomized subset of features is considered at every decision node, mitigating overfitting and enhancing generalization [33]. RF is a versatile supervised learning method suitable for classification and regression, building trees on random data samples, and finalizing predictions through a majority vote [34]. These decision trees recursively partition data based on specific criteria until a set stopping point, with tree splits determined by preset criteria [35].

3.2.4. K-Nearest neighbor (k-NN)

The K-nearest neighbor (KNN) algorithm is a supervised learning method predominantly used for classification. By measuring the similarity of data points to the nearest instances in the training set, KNN classifies them based on the most frequent class among its "K" neighbors. The effectiveness of KNN relies on several parameters, such as the choice of "K," distance metrics like Euclidean or Manhattan, and the normalization of data [36, 37] Introduced by Fix and Hodges (1952), the algorithm's computational demands increase with larger datasets [38]. Normalizing training data is pivotal to its accuracy [39].

3.2.5. Support vector machines (SVM)

The Support Vector Machine (SVM) is a powerful supervised machine learning technique introduced by Vapnik et al. in 1997, rooted in statistical learning theory [40]. Designed for classification and regression tasks, SVMs excel in diverse applications such as learning, clustering, and density estimation. The algorithm is exceptionally versatile, addressing both binary and multi-class classification problems. The crux of SVM lies in identifying support vectors—the data points nearest to the class boundaries—and maximizing the distance between these vectors and the separating hyperplane. While various hyperplanes might separate the data, SVM chooses the one with the maximum distance from both classes, ensuring optimal and robust classification [41].

3.2.6. CatBoostClassifier

The CatBoostClassifier is a gradient-boosting algorithm designed to work effectively with categorical features. Developed by Yandex, a Russian search engine company, it utilizes a collection of decision trees to make predictions. A standout capability of the CatBoostClassifier is its ability to handle categorical features without needing one-hot or label encoding. It employs a technique called "ordered boosting," which considers the order of categories, enhancing the algorithm's performance [42].

3.2.7. XGBoost

XGBoost, for eXtreme Gradient Boosting, is a widely-used machine learning algorithm suitable for regression and classification tasks. It operates within a gradient-boosting framework, combining multiple weak predictive models, typically decision trees, to form a robust predictive model. XGBoost differentiates itself by building trees using numerous cores and organizing data to minimize search times. Such efficiency measures reduce model training times, improving performance [43, 44].

3.3. Model performance and evaluation

The TP, TN, FP, and FN Confusion matrix metrics provide values for correct or incorrect classification of packets in the firewall. These values were used to calculate precision, recall, f-measure, and accuracy metrics as follows [45]:

$$\text{Precision} = \frac{TP}{(TP+FP)} \tag{1}$$

$$\text{Recall} = \frac{TP}{(TP+FN)} \tag{2}$$

$$\text{F-measure} = \frac{(2*precision*recall)}{(precision+recall)} \tag{3}$$

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{4}$$

The Sigmoid and ReLU activation functions used in the artificial neural network are calculated as follows:

$$\text{Sigmoid} : f(a) = \frac{1}{1+e^{-a}} = \frac{e^{a}}{1+e^{a}} \tag{5}$$

$$\text{ReLU} : (0,a)=0,\text{if } a<0;\ (0,a)=a,\text{if } a\geq 0 \tag{6}$$

Table 2 gives the confusion matrix table.

Table 2 Confusion matrix

Predict Class			
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

3.4. Limitations

This study, while comprehensive in its approach, acknowledges several limitations. Our analysis primarily relied on the NATICUSdroid dataset, which, despite its breadth, might not encapsulate the entire Android ecosystem's nuances. The range of algorithms employed, from conventional techniques like RF and SVM to advanced ones like CNN and ANN, leaves out potential hybrid models and other sophisticated methodologies. Our focus on 86 permissions as features might not capture the full spectrum of signals beneficial for malware detection, such as API calls or code patterns. Furthermore, the rapid evolution of malware techniques poses a challenge, suggesting that today's effective models might struggle with tomorrow's threats. Generalizing our promising results, especially from the RF model, to real-world scenarios requires caution due to the inherent unpredictability of malware distribution in live environments. The "black box" nature of some models, intense learning ones, presents a transparency challenge, and a more granular comparative analysis among models could further enrich our insights. These recognized limitations pave the way for future research aiming for a more holistic view of malware detection.

4. Results

Our extensive analysis of Android permissions using various machine-learning algorithms observed notable distinctions in performance metrics across the models.

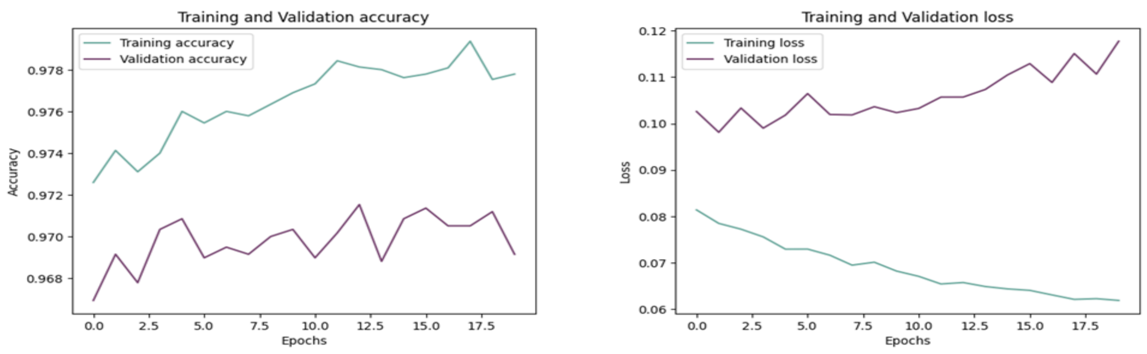


Figure 4. CNN Accuracy and loss of training and validation values

As seen in Figure 4, the 20-epoch CNN results show a model with stable performance but signs of overfitting. While the training accuracy increased slightly from 97.73% to 97.94% and the loss decreased, the validation metrics were less consistent. Validation loss rose from 0.1161 to 0.1307, and accuracy hovered in the 96-97% range. This divergence between training and validation suggests the model is overfitting, excelling on training data but not generalizing effectively to new data. Implementing dropout layers, data augmentation, or regularization might be beneficial to improve performance.

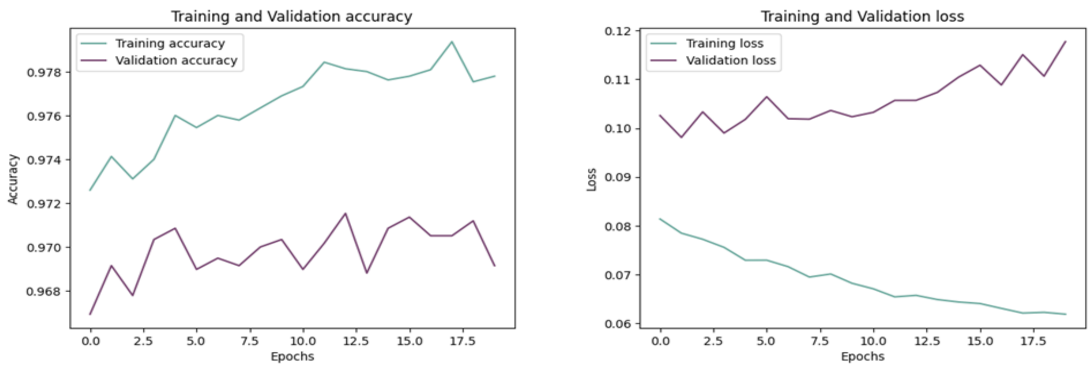


Figure 5. ANN Accuracy and loss of training and validation values

As seen in Figure 5, the results from the 20-epoch ANN algorithm display consistent model improvement. The training loss began at 0.2052 and decreased to 0.0824 by the end, while the training accuracy rose from 92.70% to 97.34%. Validation metrics also showed progress, with accuracy starting at 96.20% and finishing at 97.17%. However, after the 10th epoch, the validation metrics slightly oscillated, hinting at possible overfitting. The closeness of the training and validation metrics suggests good model generalization, but future training beyond 20 epochs should be approached with caution to avoid overfitting. Regularization or dropout might be considered for enhanced robustness in extended training.

Table 3. Analysis results of the Confusion Matrix

		CNN		ANN		RF		k-NN		SVM		CatBoostClassifier		XGBoost	
		Predicted													
Actual	0	2849	95	2879	65	2876	68	2864	80	2800	156	2877	67	2874	70
	1	100	2823	115	2808	100	2823	125	2798	116	2795	101	2822	90	2833

The Confusion Matrix is a diagnostic tool for classification models, delineating the nuances of their predictive performance. Each algorithm in the confusion matrix in Table 3 reveals its strengths and potential areas for improvement. XGBoost distinctly emerges as the frontrunner, delivering a harmonious blend of high true positives and one of the lowest false positives, underscoring its adeptness at accurately distinguishing both classes. Conversely, the SVM exhibits a propensity for a higher rate of false negatives, indicating occasional oversights in identifying positive instances. The k-NN algorithm, while proficient, sometimes misrepresents negative instances as positive, as denoted by its elevated false positive count. CatBoostClassifier and RF, both robust, closely trail XGBoost's commendable performance. Although slightly lagging behind the top trio of XGBoost, CatBoostClassifier, and RF, CNN and ANN still showcase admirable proficiency. Collectively, these results underscore the quintessential role of algorithm selection and optimization dictated by the dataset's inherent characteristics and distribution.

Table 4. Analysis results of the dataset

	CNN	ANN	RF	k-NN	SVM	CatBoostClassifier	XGBoost
Accuracy	0.966	0.969	0.971	0.952	0.953	0.971	0.972
F1 Score	0.966	0.969	0.971	0.961	0.953	0.971	0.972
Times	98sn	48sn	3sn	17sn	21sn	59sn	3sn

In the analysis presented in Table 4, seven classification algorithms are compared based on Accuracy, F1 Score, and running time, revealing nuanced insights into their performance. The XGBoost model achieves the highest accuracy and F1 score at 0.972, closely followed by the Random Forest and CatBoostClassifier models at 0.971. This suggests that ensemble methods, particularly those based on decision tree ensembles like XGBoost, Random Forest, and CatBoost, perform exceptionally well in accuracy and maintain a balance between precision and recall, as reflected by the F1 scores.

CNN and ANN also show strong performance with accuracy and F1 scores above 0.96, indicating their capability to capture complex data patterns. However, they require significantly more computation time, with CNN being the most time-consuming at 98 seconds and ANN at 48 seconds. This highlights a trade-off between performance and computational efficiency when employing deep learning models.

K-NN and SVM exhibit the lowest accuracy and F1 scores among the models, with k-NN at 0.952 and SVM at 0.953 for accuracy and a slightly better F1 score for k-NN at 0.961 than SVM's F1 score. Despite their lower performance metrics, k-NN and SVM are relatively more computationally efficient than CNN and ANN but less so than RF and XGBoost, which only require 3 seconds to compute, making them highly efficient choices with superior accuracy.

In summary, ensemble methods like XGBoost, Random Forest, and CatBoostClassifier offer an optimal blend of high accuracy, excellent F1 scores, and computational efficiency. In contrast, CNN and ANN, while powerful in model performance, demand significantly higher computation times, potentially limiting their applicability in scenarios where rapid model inference is critical. Meanwhile, traditional machine learning models like k-NN and SVM provide a decent balance between computational demand and model performance but do not match the superior metrics of ensemble methods.

Table 5. Previous similar studies and their results

Ref / Year	Dataset	Number of features	Applied models	Results-Accuracy
[6] / 2021	NATICUSdroid	29,000 benign	RF	97%
[13] / 2023	SFCGDroid	26,939 Android software datasets		98.22%
[17] / 2020	Didroid		Deep Learning	93.36%
[20] / 2023	Drebin CICMaldroid2020	Drebin: 204 features CICMaldroid:337 features	RF, ET, DNN, 1D-CNN	Drebin 1D-CNN and RF 99.6% CICMaldroid DNN 98.26%, 1D-CNN 98.2%
[46] / 2014	Drebin	123,453 applications and 5,560 malware samples	SVM	94%
[47] / 2022	CICMaldroid2020	17,341samples	semi-supervised DNN	97.7%
[48] / 2021	CCCS-CIC-AndMal-2020	14 malware categories and 180 malware families	J48, NB, SVM, AB, LR, KNN, RF, MLP	Malware Category over 96 % Malware Family over 99%
[49] / 2019	TFDroid		SVM	93.7%
[18] / 2022	MaMaDroid MAPAS		CNN	MAPAS 91.27% MaMaDroid 84.99%
Our Study	**NATICUSdroid**	**29332 records of 86 permissions**	**CNN, ANN, RF, k-NN, SVM, CatBoostClassifier, XGBoost**	**XGBoost 97.2% RF 97.1%**

Table 5 reviews studies from 2014 to 2023 on malware detection across various datasets. Mathur et al. [6] worked on the NATICUSdroid dataset with 29,000 benign features, achieving a 97% accuracy using Random Forest (RF). Similarly, Shi et al. [13] reported a 98.22% accuracy and 98.20% F1 score from the SFCGDroid dataset with 26,939 Android software. On the Didroid dataset, Rahali et al. [17] used deep learning to reach 93.36% accuracy. A multi-model approach was taken by Nguyen et al. [20] on two datasets (Drebin and CICMaldroid2020), with the highest accuracy being 99.6% on Drebin using 1D-CNN and RF. Arp et al. [46] also analyzed the Drebin dataset, achieving 94% accuracy using SVM. Employing a semi-supervised DNN on the CICMaldroid2020 dataset, Mahdavifar et al. [47] reported a 97.7% accuracy. Fiky et al. [48] showcased a broad model application on the CCCS-CIC-AndMal-2020 dataset, with results surpassing 96% for malware category accuracy and over 99% for malware family accuracy. CNN models were applied by Kim et al. [18] on the MaMaDroid and MAPAS datasets, obtaining accuracies of 84.99% and 91.27%, respectively. Lastly, a multi-algorithm study labeled "Our Study" was conducted on the NATICUSdroid dataset, wherein the RF model yielded an accuracy of 97.1%.

5. Conclusion and Discussion

The landscape of malware detection has undergone significant advancements, with studies spanning from 2014 to 2023 adopting various datasets and leveraging diverse machine learning and deep learning algorithms. Our comprehensive review of these studies showcases the consistent effort toward achieving higher accuracy rates and understanding the nuances of malware categorization and family detection.

Upon comparison, it is evident that Random Forest (RF) consistently performed well across different datasets, as noted in studies by Mathur et al. [6] and our own, registering accuracy rates of around 97%. However, Nguyen et al. [20] demonstrated that with a suitable dataset and model combination, particularly 1D-CNN on the Drebin dataset, accuracy could skyrocket to 99.6%. Such high-accuracy figures emphasize the potential of hybrid approaches, blending traditional machine-learning techniques with deep-learning structures.

However, it is also essential to acknowledge the robust results derived from singular models. The SVM, as utilized by Arp et al. [46] and Lou et al. [49], generated commendable results, highlighting the continued relevance of foundational machine learning methods amidst the surge of deep learning models. Kim et al. [18] focus on CNNs underscores the increasing reliance on deep learning for complex classification tasks, especially given the intricate nature of malware detection.

Our study's multi-algorithm approach to the NATICUSdroid dataset was insightful, revealing each model's strengths and limitations in the dataset's context. While XGBoost emerged as the top performer, it was enlightening to juxtapose its results against algorithms like CNN, ANN, k-NN, SVM, CatBoostClassifier, and RF.

In conclusion, while the overarching goal across studies remains consistent—achieving higher accuracy in malware detection—the path to that end varies. It is essential to focus on the accuracy figures and consider factors like the false-positive rate, F1 score, and the intricacies of the dataset in use. As malware continues to evolve, so too must our methodologies, urging a blend of both foundational and avant-garde approaches to stay ahead in the ever-evolving cyber landscape.

Article Information Form

Funding
The authors has no received any financial support for the research, authorship or publication of this study.

Authors' Contribution
The authors contributed equally

The Declaration of Conflict of Interest/ Common Interest
No conflict of interest or common interest has been declared by the authors.

The Declaration of Ethics Committee Approval
This study does not require ethics committee permission or any special permission.

The Declaration of Research and Publication Ethics
The authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

References

[1] A. Turner. (2022, Jan 12). How many Android users are there? Global statistics. [Online]. Available: https://www.bankmycell.com/blog/how-many-android-users-are-there

[2] Google. (2023, Aug 26). Wear OS by Google. [Online]. Available: https://wearos.google.com

[3] Android. (2023, Aug 25). Android TV. [Online]. Available: https://www.android.com/tv/

[4] S. Büyükgöze, “Mobil uygulama marketlerinin güvenlik modeli incelemeleri,” Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 12(1), pp.9-18. 2019.

[5] A. Kivva, (2023, Jun 07). IT threat evolution Q1 2023. Mobile statistics. [Online]. Available: https://securelist.com/it-threat-evolution-q1-2023-mobile-statistics/109893/

[6] A. Mathur, L. M. Podila, K. Kulkarni, Q. Niyaz, A. Y. Javaid, “NATICUSdroid: A malware detection framework for Android using native and custom permissions,” Journal of Information Security and Applications, vol. 58, no. 102696, p. 102696, 2021.

[7] A. Mathur, E. Ewoldt, Q. Niyaz, A. Javaid, X. Yang, “Permission-educator: App for educating users about android permissions,” in Conf. Intelligent Human Computer Interaction, Cham: Springer International Publishing, 2022, pp.361–371.

[8] K. Liu, G. Zhang, X. Chen, Q. Liu, L. Peng, L. Yurui, “Android malware detection based on sensitive patterns,” Telecommunication Systems, vol. 82, no. 4, pp. 435–449, 2023.

[9] Android Developers. (2023, Aug 26). Permissions on android. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview.

[10] E. Georgescu, (2020, Oct 16). The hidden dangers of Android permissions - description and mitigation. [Online]. Available: https://heimdalsecurity.com/blog/android-permissions-full-guide/.

[11] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, M. A. Masud, “Android malware classification using optimum feature selection and ensemble machine learning,” Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100–111, 2023.

[12] Q. Wu, X. Zhu, B. Liu, (2021). “A survey of android malware static detection technology based on machine learning,” Mobile Information Systems, pp. 1-18, 2021.

[13] S. Shi, S. Tian, B. Wang, T. Zhou, G. Chen, “SFCGDroid: android malware detection based on sensitive function call graph,” International Journal of Information Security, pp.1-10, 2023.

[14] L. Zhen, R. Wang, N. Japkowicz, D. Tang, W. Zhang, J. Zhao, “Research on unsupervised feature learning for Android malware detection based on Restricted Boltzmann Machines,” Future Generation Computer Systems, Volume 120, pp.91-108, 2021.

[15] Y. Zhou, X. Jiang, “Dissecting android malware: Characterization and evolution,” in Conf. Security and Privacy, 2012, pp.95-109.

[16] S. Y. Yerima, S. Khan, “Longitudinal performance analysis of machine learning based Android malware detectors,” in Conf. Cyber Security and Protection of Digital Services (Cyber Security), 2019, pp.1-8.

[17] A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. Gagnon, F. Massicotte, “DIDroid: Android malware classification and characterization using deep image learning,” in Conf. Communication and Network Security, 2020, pp.70-82.

[18] J. Kim, Y. Ban, E. Ko, H. Cho, 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.

[19] F. Giannakas, V. Kouliaridis, G. Kambourakis, “A closer look at machine learning effectiveness in Android malware detection,” Information (Basel), vol. 14, no. 1, p. 2, 2022.

[20] C. D. Nguyen, N. H. Khoa, K. N. D. Doan, N. T. Cam, “Android Malware Category and Family Classification Using Static Analysis,” in Conf. Information Networking (ICOIN), IEEE, 2023, pp. 162-167.

[21] C. Ding, N. Luktarhan, B. Lu, W. Zhang, “A hybrid analysis-based approach to android malware family classification,” Entropy, 23(8), 1009, 2021.

[22] M. N. U. R. Chowdhury, A. Haque, H. Soliman, M. S. Hossen, T. Fatima, I. Ahmed, “Android malware Detection using Machine learning: A Review,” arXiv preprint arXiv:2307.02412, 2023.

[23] H. Rathore, S. Chari, N. Verma, S. K. Sahay, M. Sewak, “Android Malware Detection Based on Static Analysis and Data Mining Techniques: A Systematic Literature Review,“ in Conf. Broadband Communications, Networks and Systems Cham: Springer Nature Switzerland, 2023, pp. 51-71.

[24] A. Mathur, NATICUSdroid (Android Permissions) Dataset. UCI Machine Learning Repository, 2022.

[25] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition,” in Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp.770-778.

[26] E. Öztemel, Yapay sinir ağlari, Papatya Yayincilik, ISBN: 978-975-6797-39-6. Istanbul, Turkey, 2023.

[27] S. Haykin, Neural Networks and Learning Machines, Pearson: Upper Saddle River, Neural Networks and Learning Machines, vol. 3, India, 2009.

[28] E. Egrioglu, C. H. Aladag, U. Yolcu, V. R. Uslu, M. A. Basaran, “A new approach based on artificial neural networks for high order multivariate fuzzy time series,” Expert System with Applications, vol. 36, no. 7, pp. 10589–10594, 2009.

[29] U. Porwal, Z. Shi, S. Setlur, Machine learning in handwritten Arabic text recognition, In Handbook of Statistics Vol. 31, pp. 443-469, Elsevier, 2013.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition.”, Proceedings of the IEEE, 86(11), 1998, pp.2278-2324.

[31] A. Krizhevsky, I. Sutskever, G. E. Hinton, “ImageNet classification with deep convolutional neural networks”. In Advances in neural information processing systems, pp. 1097-1105, 2012.

[32] D. Scherer, A. Müller, S. Behnke, “Evaluation of pooling operations in convolutional architectures for object recognition,” in Conf. Artificial Neural Networks (ICANN), 2010, pp. 92-101.

[33] L. Breiman, “Random forests,” Machine learning, 45(1), pp.5-32, 2001.

[34] S. J. Rigatti, “Random forests,” Journal of Insurance Medicine, 47(1), 31-39, 2017.

[35] M. Schonlau, R. Y. Zou, “The random forest algorithm for statistical learning,” The Stata Journal, 20(1), pp.3-29, 2020.

[36] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, “Supervised machine learning: A review of classification techniques,” Emerging artificial intelligence applications in computer engineering, 160(1), pp.3-24, 2007.

[37] Ö. Tomak, Derin Öğrenme Algoritmalarının EKG Aritmilerinin Sınıflandırılmasında Değerlendirilmesi, Karadeniz Teknik Üniversitesi, Trabzon, 2018.

[38] G. Bilgin, “Makine öğrenmesi algoritmaları kullanarak erken dönemde diyabet hastalığı riskinin araştırılması,” Journal of Intelligent Systems: Theory and Applications, 4(1), pp.55-64, 2021.

[39] O. Sevli, “Farklı Sınıflandırıcılar ve Yeniden Örnekleme Teknikleri Kullanılarak Kalp Hastalığı Teşhisine Yönelik Karşılaştırmalı Bir Çalışma,” Journal of Intelligent Systems: Theory and Applications, 5(2), pp.92-105, 2022.

[40] V. Vapnik, S. Golowich, A. Smola, “Support vector method for function approximation, regression estimation and signal processing,” Advances in neural information processing systems, 9, pp.281-287, 1996.

[41] S. R. Gunn, “Support vector machines for classification and regression”, ISIS technical report, 14(1), pp.5-16, 1998.

[42] B. Deekshitha, C. Aswitha, C. S. Sundar, A. K. Deepthi, “URL Based Phishing Website Detection by Using Gradient and Catboost Algorithms.” International Journal Research Applied Science and Engineering Technology, 10(6), pp.3717-3722, 2022.

[43] S. Ramraj, N. Uzir, R. Sunil, S. Banerjee, “Experimenting XGBoost algorithm for prediction and classification of different datasets,” International Journal of Control Theory and Applications, 9(40), pp.651-662, 2016.

[44] N. Memon, S. B. Patel, D. P. Patel, “Comparative analysis of artificial neural network and XGBoost algorithm for PolSAR image classification,” in Conf. Pattern Recognition and Machine Intelligence, Cham: Springer International Publishing, 2019, pp.452-460.

[45] A. Korkmaz, S. Büyükgöze, “Sahte Web Sitelerinin Sınıflandırma Algoritmaları İle Tespit Edilmesi,” Avrupa Bilim ve Teknoloji Dergisi, (16), pp.826-833, 2019.

[46] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, “Drebin: Effective and explainable detection of android malware in your pocket,” In Conf Network and Distributed System Security Symposium (NDSS), Vol. 14, 2014, pp. 23-26.

[47] S. Mahdavifar, D. Alhadidi, A. A. Ghorbani, “Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder,” Journal of network and systems management, 30, pp.1-34, 2022.

[48] A. H. E. Fiky, A. E. Shenawy, M. A. Madkour, “Android malware category and family detection and identification using machine learning,” arXiv preprint arXiv:2107.01927, 2021.

[49] S. Lou, S. Cheng, J. Huang, F. Jiang, “TFDroid: Android malware detection by topics and sensitive data flows using machine learning techniques,” in Conf. information and computer technologies (ICICT) IEEE, 2019, pp.30-36.