INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Comprehensive Review of Feature Selection Strategies for Machine Learning Based Android Malware Detection

[1]Sonal Pandey, [2]Sandeep Kumar Jain, [3]Anil Kumar Gupta

[1]Research Scholar, [2]Professor, [3]Professor
[1]Department of Computer Science, [2]Department of Computer Science, [3]University Computer Centre
[1]Dr. Bhimrao Ambedkar University, Agra, India,
[2]Dr. Bhimrao Ambedkar University, Agra, India,
[3]Dr. Bhimrao Ambedkar University, Agra, India

Abstract: The emergence of even more advanced attacks of mobile malware that have been trying to steal user credentials and other important data has also been accompanied by the concept of mobile smartphone use that is projected to be in the billions across the globe. With high dimensionality of features in modern malware datasets, effective feature selection is important to perfor-mance as well as efficiency of machine learning (ML) models, which often involve selecting most relevant features from a potentially large set of features. This process not individual reduce computational overhead but also increases the accuracy and generalization of models, which can overcome the challenges associated with more complex and numerous malicious software instances. Feature selection is an important step in designing accurate and efficient Android malware detection systems because of high dimensionality of data obtained from static and dynamic application analysis. This problem is compounded by the fact that cyber-criminals are constantly inventing new ways to obscure their attacks, and signature-based detection is no longer sufficient, which has led to need for more dynamic ML approaches.

Index Terms - Malware detection, Machine learning, Comparative analysis.

I. INTRODUCTION

As the Internet and Android operating systems have grown in popularity, so too have the number of active users and daily activities of each user on Android smartphones. As a result, malware writers now target Android smartphones. Mobile malware has also significantly increased because of open nature of Android operating system. It is quite important to safeguard the systems against the fast-developing malware using Android malware detection. Traditional methods are often not capable of identifying malicious applications due to the growing number, diversity, and sophistication of malware. The analysis of the security detection according to Android showed that there is an alarming spread of threats in three major categories. The most frequent threat is malware, which consists of 42 percent of all the detections, and it is possible to conclude that malicious software is very prevalent, and the Android devices are the victims. Potentially unwanted programmers (PUPs) were the second most common threat with 32% indicating that there is a substantial number of questionable applications that can endanger the privacy of user or their devices. Adware accounts for 26% of detections, demonstrating the ongoing presence of intrusive advertising software that can harm the user experience and potentially serve as entry points for other threats [3].

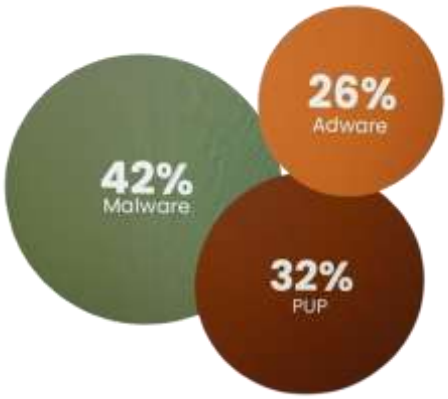

Fig. 1. Android Threat Detections 2024[3]

Additionally, Android permits the installation and downloading of Android applications from unreliable sources. Consequently, bundling and disseminating programs with malware has become simple for malware producers. Consequently, it is now difficult to identify Android malware quickly and accurately. A more dependable, accurate, generalized, and practical approach for Android malware detection is required in light of these problems.

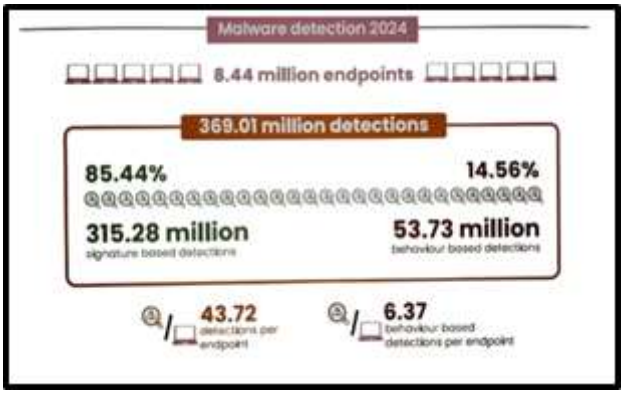

Fig. 2. Malware Detection 2024 [3]

Signature-based methods work fine compared to recognized malware, but they don't work well against new or unknown malware. Machine learning (ML) also deep learning (DL) practices were presented by academia towards sense sophisticated malware in the current environment. Feature vectors and sizable datasets are crucial for ML. We discovered throughout the literature review that many other writers are utilizing ML approaches to identify malware early on. Simultaneously, they have a sufficient dataset, but their lack of functional characteristics reduces the detection accuracy [3].

1.1 Malware

Malicious software, or malware, is any application, program, or code that is intentionally created to circumvent the authorization process, access a system, and reach a state where it can perform unauthorized operations. As malware developers are constantly developing new malicious code every other day, it is very challenging to detect them on zero days.

1.1.1 Malware Types

In addition to many traditional methods, there are several related phrases that may be employed for sending malicious and fraudulent packets, which include:

- Virus (Vital Information Resources Under Seize)
- Trojan Horse
- Worms
- Rootkit
- Spyware
- Bots
- Adware (Advertising-supported software)
- Ransomware.

1.2 Malware Detection Techniques

Detection methods are employed to examine the behavior of the software and determine whether it is harmful or safe. The capacity to effectively handle obfuscated malware is essential for robust malware detection [22]. When creating second-generation malware, two popular obfuscation approaches are polymorphism and metamorphism. Anti-malware software is designed to counteract malware threats and attacks, and its main presumption is that malware structures are relatively constant. Therefore, collaboration between academic institutions and anti-malware developers is necessary to stop the harm caused by malware. The methods for malware detection are as follows.

- Heuristic-Based Detection
- Signature-Based Detection
- Malware Normalization

1.3 Malware Analysis Approaches

Following detection phase, machine learning, heuristic, signature-based, and normalization approaches were implemented to complete process further, employing static, dynamic, and hybrid analysis. Term malware study approach refers to the way in which detection methods collect data that is then utilized to identify dangerous software.

1.3.1 Static Analysis. By deconstructing the programs using a disassembler tool, we are able to extract elements from the code that are utilized to discriminate among malicious and benign programs.

1.3.2 Dynamic Analysis. Dynamic examination, often referred towards by means of communication study, entails performing intrusive software as well as monitoring its activities, system interactions, as well as host computer impacts.
1.3.3 Hybrid Analysis. This technique syndicates static and dynamic malware analysis techniques. It practices a dynamic analysis strategy to improve the overall analysis after first analyzing malware code using a static analysis technique [19].

1.4 Feature Extraction in Android Malware Detection

Feature selection is inextricably linked to feature extraction, process of defining and computing candidate features from raw application data. In Android malware recognition, feature extraction spans static, dynamic, and hybrid methodologies.

1.4.1 Static Feature Extraction. Static study inspects the application package (APK) without executing it. Common static features include:

- **Permissions**: Set of permissions requested by app (e.g., SEND_SMS, READ_CONTACTS) is a strong indicator of potential malicious intent
- **Opcode Sequences:** The obfuscated or suspicious control flows can be identified by the sequence of operation codes in the app in the form of its bytecode.
- **API Calls:** The existence and the frequency of certain API calls, especially those involving sensitive operations (e.g., access to user data, SMS), are the frequently used features.
- **Metadata:** Developer signature, package names, and certificate information are some of the features that give the context of the app's provenance and the possibility of repackaging.

1.4.2 Dynamic Feature Extraction. Dynamic analysis monitors the behaviour of applications during execution, normally in a simulated or instrumented platform. Dynamic features include:

- **Runtime API Call Sequences:** Behavioural profiling can be used with the sequence and frequency of API calls in use, which can be captured during execution.
- **System Calls:** The attempts to use the vulnerabilities or escalate privileges can be detected by low-level communication with the operating system.
- **Network Traffic Patterns:** A review of the outbound connections, the attempts of data exfiltration, and the contact with the command-and-control servers can indicate the malicious intent.
- **Resource Usage:** Unusual CPU, memory, or battery usage can be an indication of execution of malware.

1.4.3 Hybrid Feature Extraction. Hybrid approaches combine static also dynamic features to influence the strengths of both. E.g., static permissions and API calls may be complemented by dynamic behavioral traces, yielding richer representations and improved detection accuracy.

1.5 Machine Learning

Machine learning stands a part of Artificial Intelligence (AI) concerned with building systems or models that are for learning from data and improving on a given task without being explicitly given out a guide or process to follow [33]. Supervised machine learning, unsupervised machine learning, and reinforcement machine learning are the three classes of machine learning methodologies.

1.5.1 Supervised Machine Learning. It refers to the subtype of machine learning in which prototypical is taught via categorized dataset, import that separate training example has both an input, which is the features and a corresponding output which is the label [33]. Supervised learning divided into 2 categories of algorithms.

Classification: A classification algorithm is type of supervised ML algorithm utilised to predict which category or class a given input belongs to, based on patterns learned from labelled training data. Some examples of classification algorithms are:

- Decision “Trees
- Random Forest
- k-NN
- XGBoost

Regression: Regression in ML is a type of supervised knowledge utilized to forecast a continuous numerical value relied on input data [33]. Some examples of Regression Algorithms are:

- Random Forest Regression
- Support Vector Regression (SVR)
- Linear Regression

1.5.2 Unsupervised Learning. This is form of MLF which does not need any form of supervision. Rather, the model is supposed to work autonomously to discover some information. It primarily focuses on unlabeled datasets (i.e., data without predefined categories or outputs).

1.5.3 Reinforcement Learning. Reinforcement Learning (RL) is sort of machine learning wherever agent acquires how to make results by relating through an atmosphere. Looking at term's agent, action environment, state, and reward, the concept of reinforcement learning can be explained.

II REVIEW OF LITERATURE

The goal of **Kurniawan et al. (2025)** is to provide a ML agenda used for early Android malware recognition that is both reliable as well as effective. Malicious activity in APK files is detected using six different ML algorithms. Reverse engineering was employed to extract the 2,084 Android applications in the collection, which includes 770 benign apps and 1,314 malware variants. Outperforming other algorithms, Random Forest method performs best, with 97% accuracy and 95% precision, according to trial data [6].

Panja et al. (2025) developed fourteen ML models based on the research dataset using the five-fold cross-validation strategy. Each model was evaluated for execution time as well as memory usage under two scenarios: utilizing all features as well as utilizing reduced set of features obtained after data pre-processing stage. Top ten most important features were selected using the Extra Trees Classifier (ETC) based on Gini impurity scoring, that improved efficiency and accuracy. With a ROC-AUC score of 0.99, Random Forest (RF) model trained with condensed feature set had an exceptionally high prediction accuracy of 99.39%, according to the results [7].

Park et al. (2025) introduces a two-stage machine learning system in the field of Android malware classification that would be able to enhance detection accuracy and promote sustainability. The information of the SDK version is first used to predict the year of release of the app. In the second step, programmes are classified as malicious or harmless using a weighted voting mechanism that relies on the malware detection algorithms that are specific to years. This technique ensures that robust malware detection is scalable, and the computational cost is minimised, but the precision is equal to that achieved by retraining. Two-stage classification algorithm is successful in getting the performance of retraining-based models with consistent training times and relatively small models, and this guarantees a viable way of sustainable malware detection in the actual situation. [8].

The text of **Das et al. (2024)** presents an original entropy-based encoding strategy of the category attributes of the data, which were based on malicious inputs. The test of the effectiveness of the proposed encoding strategy has been carried out on CIC-Evasive-PDFMal2022, UNSW-NB15, and KDDCUP99 datasets. Using dimension-reduced entropy-encoded KDD Cup99 dataset, the ensemble classifier trained with the recommended approach achieved a 99.99% F1 score in data classification. The ensemble approaches produced a peak F1 score of 99.27% on the CIC-Evasive-PDFMal2022 dataset, indicating a slight improvement in classification parameters for entropy encoding [9].

Pathak et al. (2024) used static analysis in a methodical manner to create an Android permission-based dataset. Author employed a systematic process to create Android permission samples (APK files), with an emphasis on permission-based feature extraction and the reverse engineering technique. Study's experimental outcomes show that the RF classifier-based Android malware recognition prototypical achieves highest correctness of 97.5% using only 48 features [10].

Kahla et al. (2024) get APK API information from dynamic analysis files along with permissions also intent filters from static analysis files. Tried several hash algorithms and tokenizing techniques in the Simhash algorithm, and carried out many tests to assess the efficacy of our methodology on a dataset of 38,355 tagged Android applications. Great accuracy was achieved with minimal memory and training time using RF as classifier, SHA-512 as hash method, as well as 2-WORD tokenization [11].

Cui et al. (2023) To describe attacking intent and create malware, the authors employed the CBSeq approach, which creates a reliable traffic representation and behavior sequence. MSFormer is potent transformer-grounded multi-sequence fusion classifier that was designed by the author. It separates malicious from benign traffic by capturing the intrinsic similarity of the activity sequence [12].

Costa et al. (2023) recommended an Android malware recognition method that practices a collection of specialized detectors, each of which does a multi-stage examination using instructions and ML approaches at various stages of the application sequence (both earlier and later installation). [13].

Huang et al. (2019) emphasized the characteristics of the application programming interface and suggested techniques for identifying fraudulent Android applications. Initially, the authors suggested a way to choose API attributes associated with the kind of infection. Second, the manually enhanced API-based feature graph allowed them to examine the structural relationships between these APIs in greater detail and move them to an interpreted matrix. Third, the API-based feature graph categorization is trained using a CNN-grounded classifier. The authors worked with 3,697 malicious and 3,312 benign applications; only the top 20 APIs were able to achieve 94.3% correctness using the RF algorithm [15].

Neha et al. (2018) centered on several ML-based analytical techniques for Android malware application categorization. Using variety of ML algorithms (comprising Random Tree, Naïve Bayes, AdaBoost, Bagging, and SMO), Opcode-based Android malware analysis technique presented in this study achieved 99.5% accuracy with 0.995 TPR [19].

Tao et al. (2018). This paper's authors examined malware's covert tendencies in actual Android applications. To find unknown Android malware, the authors developed an automated malware detection technique called MalPat. They also recovered important APIs that the virus uses. Extensive trials are carried out on a dataset that includes 15 336 malware samples and 31 185 benign apps. According to experimental findings, MalPat can identify malware with high F1 score (98.24%) [20].

Zhu et al. (2018) presented a very efficient and affordably priced method for extracting permissions as well as sensitive APIs as features. They used an ensemble Rotation Forest to train the model to discriminate between benign and damaging applications. The scientists obtained 88.26% detection correctness, 88.40% sensitivity after training the model on 2130 samples [21].
Sun et al. (2018) presented SigPID, a permission-analysis-based tool for identifying malicious applications. Only 34 rights, or 25% of the total permissions, were employed by the authors to differentiate among malicious and benign programs out of 135 permissions that they collected from the dataset. They claim an correctness of 91.4% for unknown malware and 93.62% aimed at malware in dataset, having trained their model using the SVM [22].

Ref.	Name	Detection Approach	Feature Extration/Selection Method	About the Data Set	ML algo model	Selected Model	Model Accuracy	Strength	Limitation
[5]	Hasan et al. [2025]	Assess "the influence of feature selection, feature scaling, and machine learning models on malware identification" [5].	Linear Discriminant Analysis (LDA), and Principal Component Analysis (PCA))	MalDroiod binary tabular dataset with 11,598 instances	NB, DT, SVM, LR, Extra Tree, KNN, RF, LGBM	LGBM	97.16%	Improving model selection and preprocessing techniques to create dependable and effective malware detection systems.	In publicly accessible dataset, wide range of malware data is not fully represented
[6]	Kurniawan et al. [2025]	Binary classification of lightweight code obtained from static characteristics in the Android manifest file [6].	Permissions, activities, services, content providers, and Broadcast receivers.	AMD dataset consists of 2,084 Android applications.	SVM, Random Forest, KNN, Naive Bayes, AdaBoost, and Gradient Boosting	Random Forest algorithm	97%	Early detection of malware on the Android platform	Need for deeper feature extraction techniques and detection methods
[7]	Panja et al. [2025]	Method for identifying malware in any type related to a resource-limited device within a short timeframe [7].	Variance cutoff and Variance inflation factor was employed to remove minor variations.	Kaggle with 138047 instances.	LGBM, AB, HGB, RF, DT, BNB, ETC, and KNN	Random Forest	99.39%	When it comes to devices with limited resources, the model that is being offered is responsive, stable, and robust.	The model will undergo testing and integration with various datasets, incorporating the flavor of artificial neural networks.
[10]	Pathak et al. [2024]	Employ permission-based features to generate models for identifying malware on devices running Android [10].	Permission (Feature extraction, feature importance score for Feature reduction	Malware Bazaar database and Virus Share with 585 Android APK files.	K-NN, NB, decision tree (DT), and random forest (RF).	Random forest	97.5%.	Highly effective Android malware recognition system that progresses precision and reduces model execution interval	Deep learning methods need to be used to enhance categorization accuracy.
[11]	Kahla et al. (2024)	Utilize Simhash "for encoding specific sections of the analysis files to produce feature vectors [11].	Permissions, Intent filters, API Calls	Google Play Store with 38,355 malware and benign samples.	Logistic Regression, NB, SVM, Random Forest	Logistic Regression	99.88%	Impact Analysis of Hash Functions and Tokenizing Types	Incorporate adaptive learning techniques to handle evolving malware threats.
[13]	Costa et al. [2023]	Malware APK Detection Solution (MADS) employs a multi-phase strategy to detect malware,lacking the need for an internet connection [13].	Permissions, intents, hardware component and app component	MADS dataset was created (Google Play Store, AppBrain) and total 11500 malware and benign APKs.	Random Forest algorithm	Random Forest algorithm	97%	Carries out a multi-phase analysis at various stages of the application cycle (before and after its installation) using rules and ML approaches.	CL (Confidence Level) values affect classification results.
[20]	Tao et al. [2019]	Automated malware detection system, MalPat, to combat	Permission and API calls	28,787 Gentle apps from Google -Play and 24,317	Random Forest	Random Forest	98.24%	able to collect extremely sensitive APIs that are frequently	It can be challenging to increase the mining patterns of harmful apps

		against malware and identify unfamiliar harmful applications [20].		hateful apps from VirusShare.				utilized by malware Android apps.	due to the limited quantity of malware currently gathered on the Internet.
[22]	Sun et al. [2018]	Utilize SigPID, a three-tier data purification technique through machine learning models [22].	Manifest Analysis for Permissions	5,494 malevolent apps and 310,926 gentle apps taken from Google Play.	Naïve Bayes, Decision Tree,SVM	Support Vector Machine	91.4%	High accuracy and effectiveness	Only examined the permission analysis, which might indicate omitting other important parts of the analysis.
[34]	McDonald et al. [2021]	Examining permissions and training the model using the Recognized ML algorithm [34].	Manifest Analysis for Permissions	GooglePlay, AndroZoo, AppChina	SVM, RF, Gaussian NB, K-Means	RF	81.5%	A great deal of different samples were used to train the model.	Didn't think about other parts of basic analysis like OpCode, API calls, and so on.
[35]	Sahin et al. [2021]	Dipping dimension vector generation based on that complete malware recognition by means of ML models [35].	Permissions extraction	AMD, APKPure	MLP, NB, Linear Regression	MLP	96%	It is guaranteed to be efficient, useful, and easy to understand.	It is not possible to choose hyperparameters for" use
[36]	Nawaz et al. [2021]	Choosing "feature by means of dimensionality reduction algorithms and Info Gain method [36].	Permissions and intents analysis	Drebin, Google Play	RF, NB, GB, AB	RF	98%	Analysed at the traits as separate parts instead of as a whole	Didn't think about other functions like Opcode, API calls, etc.
[37]	Cai et al. [2021]	Feature weighting via joint optimization of weight mapping utilizing the proposed JOWMDroid. context [37].	Permission, Intents, activities and Services analysis	Drebin, AMD, Google Play AKPure	SVM, RF, LR,KNN	LR	96%	Improved efficiency and accuracy	Relationships between features have not been taken into account.
[38]	Lou et al. [2019]	FlowDroid for static study and TFDroid background for malware detection. [38].	Manifest Examination for permissions and Code Review for information flow	Drebin, Google Play	SVM	SVM	93.7%	Examined the data flow by analyzing the applications' functions based on their descriptions.	Neglected to take into consideration the application of other ML models and the advancements in clustering approaches.
[39]	Kabakus et al. [2019]	Obtaining application characteristics from the manifest while decompiling classes.dex into a jar file and implementing ML models [39].	Manifest Examination for permissions, activities then Code Analysis for Opcode	Drebin, Play Store, Genome	BayesNet, SVM, NB, LR, J48, RT, RF, AB	RF	98.7%	High effectiveness, Lightweight study and completely computerized method	When analyzing the DEX, the API calls and other crucial characteristics were overlooked.
[40]	Koli et al. [2018]	Extract characteristics and transform them into binary vectors whereas training with machine learning concluded the	Manifest Analysis focused on Permissions along with Code Assessment for API calls, opcode, and native calls.	Drebin	DT, SVM, RF, NBs	Decision Tree	97.7%	Highly detailed examination of permissions, API, and opcode native invocations	Deep native code analysis, broadcast receivers, filtered intent, and Control Flow Graph analysis" were

		RanDroid Framework. [40].						for malware identification.	not taken into account.

Table 1. Static analysis-based approach

III COMPARATIVE ANALYSIS

These limitations have several implications for how the study's results should be interpreted:

- **Generalizability:** Due to the reliance on a specific public dataset and the limited scope of malware samples, the findings may not be broadly generalizable to all types of malwares or across different operational environments.
- **Completeness of Analysis:** The absence of a variety of characteristics of the static analysis (e.g., OpCode, API calls) and the absence of correlation analysis imply that the knowledge of the study may be partial. The crucial characteristics of the malware behaviour, which are not identified using permission as a one-dimensional tool, may be overlooked and result in a biassed view of the malicious activities.
- **Model Robustness and Performance**: Lack of enhanced and richer feature extraction, sophisticated generation of deep learning and adaptive learning mechanisms imply that the robustness of the model in the face of complex and evolving malware attacks may not be ideal. The unspecified hyperparameter choices may also indicate that the model is not optimally tuned, which has an impact on its stated performance.
- **Bias:** Giving the permission analysis exclusive attention and neglecting the other features may create bias in the model as it may excessively assign significance to the permission whilst ignoring other equally or more important signals of malicious behaviour.
- **Future Directions:** The recognised necessity to extract more features, combine them with neural networks, use DL, and adaptive learning signifies that the existing findings can be viewed as an initial phase of the development, and various ways of improvement and advancement are possible. Thus, the presented findings can be regarded as provisional but not final.

IV CHALLENGES AND LIMITATIONS

Irrespective of the advances, the creation of efficient malware detection software has a number of challenges:

- **Extraction Complexity and Feature Selection:** A significant difficulty is that it is hard to pick and remove features such as permissions, API calls, and control flow structures, as the optimization of malware obfuscation has been growing as much as the practice of repackaging is [6].
- **Data Representation:** Depending on publicly-accessible datasets might not be a complete reflection of a wide range of malware that exist in the real world, which influences the generalizability of results [5].
- **Computational Efficiency:** Although, the predictive accuracy of ML models is high, multiple algorithms may be used to improve it, but this should be optimized [6].

V FUTURE DIRECTIONS

It is recommended that further studies be conducted to address some of the critical areas that could be used to improve the malware detection capabilities:

- Hybrid Methodologies: The incorporation of dynamic analysis issues into detection systems and the study of hybrid frameworks that integrate dynamic and static analysis are important. It involves the use of improved DL methods and models such as CNN and LSTM to improve the detection strength in relation to complex obfuscation [6].
- Advanced Modeling and Preprocessing: The future research should consider more sophisticated modelling techniques such as deep neural networks and more powerful preprocessing approaches such as feature scaling and feature selection to enhance its performance, robustness, and interpretability. [6].
- Adversarial Robustness: The future research should consider more sophisticated modelling techniques such as deep neural networks and more powerful preprocessing approaches such as feature scaling and feature selection to enhance its performance, robustness, and interpretability. [6].
- Real-time Implementation and Large-scale Datasets: Future studies need to focus on real-time tracing of application behaviours through known dynamic analysis tools and carrying out intensive reviews of parallel and distributed computing applications on large datasets. [6].
- Diverse Datasets and Feature Selection: In an attempt to make the detection even more efficient, researchers could use more datasets to test on and investigate more complex approaches to choosing features beyond PCA and LDA. [6].

VI CONCLUSION

The results indicate that a cohesive framework is required to evaluate and replicate the results of feature assortment methods in Android malware recognition, both to overcome the existing limitations of benchmarking and contribute to the development of the field The results suggest that a coherent framework is needed to compare methods and identify optimal sets of features that can be used to recognize malicious activities in addition to increasing detection rates.

REFERENCES

1. IDC Worldwide Quarterly Mobile Phone Tracker - 2025 Jul. (2025). https://my.idc.com/getdoc.jsp?containerId=IDC_P8397 [Accessed: 02-july-2025].
2. E. Protalinski, "Android passes 2.5 billion monthly active devices – Venturebeat"(2024). https://venturebeat.com/2019/05/07/%0Aandroid-passes-2-5-billion-monthly-active-devices/. [Accessed: 15-july-2025].
3. India Cyber Threat Report. (2025). https://www.quickheal.co.in/threat-reports?year=2025. [Accessed: 17-july-2025].
4. Android, "Platform Architecture Agenda," Architecture. (2025). https://developer.android.com/get-started/overview. [Accessed: 22-Sep-2023].
5. Hasan, R., Biswas, B., Samiun, M., Saleh, M. A., Prabha, M., Akter, J., Joya, F. H., & Abdullah, M. (2025). Enhancing malware detection with feature selection and scaling techniques using machine learning models. Scientific Reports, 15(1). https://doi.org/10.1038/s41598-025-93447
6. Kurniawan, F., Stiawan, D., Antoni, D., Yazid Idris, M., & Budiarto, R. (2025). A Robust and Efficient Machine Learning Framework for Enhancing Early Detection of Android Malware. IEEE Access, 13, 127183–127220. https://doi.org/10.1109/ACCESS.2025.3589656
7. Panja, S., Mondal, S., Nag, A., Prakash Singh, J., Jyoti Saikia, M., & Kumar Barman, A. (2025). An Efficient Malware Detection Approach Based on Machine Learning Feature Influence Techniques for Resource-Constrained Devices. IEEE Access, 13, 12647–12665. https://doi.org/10.1109/ACCESS.2025.3526878
8. Park, S., Lee, H., Kim, D., Jun Moon, H., Cho, S. J., Hwang, Y., Han, H., & Suh, K. (2025). Enhancing the Sustainability of Machine Learning-Based Malware Detection Techniques for Android Applications. IEEE Access, 13, 98876–98887. https://doi.org/10.1109/ACCESS.2025.3576733
9. Das, V., Nair, B. B., & Thiruvengadathan, R. (2024). A novel feature encoding scheme for machine learning based malware detection systems. IEEE Access, 12, 91187–91216. https://doi.org/10.1109/ACCESS.2024.3420080
10. Pathak, A., Kumar, T. S., & Barman, U. (2024). Static analysis framework for permission-based dataset generation and android malware detection using machine learning. Eurasip Journal on Information Security, 2024(1). https://doi.org/10.1186/s13635-024-00182-3
11. Al-Kahla, W., Taqieddin, E., Shatnawi, A. S., & Al-Ouran, R. (2024). Malware detection and classification in Android application using Simhash-based feature extraction and machine learning. IEEE Access, 12, 174255–174273. https://doi.org/10.1109/ACCESS.2024.3501277
12. Cui, S., Dong, C., Shen, M., Liu, Y., Jiang, B., & Lu, Z. (2023). CBSeq: A Channel-level Behavior Sequence for Encrypted Malware Traffic Detection. 18(9). http://arxiv.org/abs/2307.09002
13. Costa, L. da, & Moia, V. (2023). A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques. IEEE Access, 11, 73127–73144. https://doi.org/10.1109/ACCESS.2023.3296606
14. Zhang, Z., Chang, C., Han, P., & Zhang, H. (2020). Packed malware variants detection using deep belief networks. 309. https://doi.org/10.1051/matecconf/202030
15. Huang, N., Xu, M., Zheng, N., Qiao, T., & Choo, K. K. R. (2019). Deep android malware classification with API-based feature graph. Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigData SE 2019, 296–303. https://doi.org/10.1109/TrustCom/BigDataSE.2019.00047
16. Hernández Jiménez, J. M., & Goseva-Popstojanova, K. (2019). Malware Detection Using Power Consumption and Network Traffic Data. https://doi.org/10.1109/ICDIS.2019.00016
17. Laya Taheri, A. F. A. Kadir. A. H. Lashkari. (2019). 2019 International Carnahan Conference on Security Technology (ICCST). https://doi.org/10.1109/CCST.2019.8888430
18. Sharma, S., Rama Krishna, C., & Sahay, S. K. (2019). Detection of advanced malware by machine learning techniques. Advances in Intelligent Systems and Computing, 742, 333–342. https://doi.org/10.1007/978-981-13-0589-4_31
19. Tarar, N., Sharma, S., & Krishna, C. R. (2018). Analysis and Classification of Android Malware using Machine Learning Algorithms. Proceedings of the 3rd International Conference on Inventive Computation Technologies, ICICT 2018, 738–743. https://doi.org/10.1109/ICICT43934.2018.9034337
20. Tao, G., Zheng, Z., Guo, Z., & Lyu, M. R. (2018). MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs. IEEE Transactions on Reliability, 67(1), 355–369. https://doi.org/10.1109/TR.2017.2778147
21. Zhu, H. J., You, Z. H., Zhu, Z. X., Shi, W. L., Chen, X., & Cheng, L. (2018). DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing, 272, 638-646. https://doi.org/10.1016/j.neucom.2017.07.030
22. Sun, L. (2018). Significant Permission Identification for Android Malware Detection. “Entification for Android Malware Detection”. Computer Science and Engineering: Thesis, Dissertations, and Student Research. 104, 14(7), 3216–3225. http://digitalcommons.unl.edu/computerscidiss
23. Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). Machine learning aided Android malware classification. Computers and Electrical Engineering, 61, 266–274. https://doi.org/10.1016/j.compeleceng.2017.02.013

24. Rashidi, B., Fung, C., & Bertino, E. (2017). Android resource usage risk assessment using hidden Markov model and online learning. Computers and Security, 65, 90–107. https://doi.org/10.1016/j.cose.2016.11.006
25. Xu, K., Li, Y., & Deng, R. H. (2016). ICCDetector: ICC-Based Malware Detection on Android. IEEE Transactions on Information Forensics and Security, 11(6), 1252–1264. https://doi.org/10.1109/TIFS.2016.2523912
26. Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. Soft Computing, 20(1), 343–357. https://doi.org/10.1007/s00500-014-1511-6
27. Das, S., Liu, Y., Zhang, W., & Chandramohan, M. (2016). Semantics-based online malware detection: Towards efficient real-time protection against malware. IEEE Transactions on Information Forensics and Security, 11(2), 289–302. https://doi.org/10.1109/TIFS.2015.2491300
28. Zhang, Y., Huang, Q., Ma, X., Yang, Z., & Jiang, J. (2016). Using multi-features and ensemble learning method for imbalanced malware classification. 2016 IEEE Trustcom/ BigDataSE/ ISPA, 965–973. https://doi.org/10.1109/TrustCom/BigDataSE/ISPA.2016.161
29. Sharma, A., & Sahay, S. K. (2016). An effective approach for classification of advanced malware with high accuracy. International Journal of Security and Its Applications, 10(4), 249–266. https://doi.org/10.14257/ijsia.2016.10.4.24
30. Kruczkowski, M., & Niewiadomska-Szynkiewicz, E. (2014). Comparative study of supervised learning methods for malware analysis. Journal of Telecommunications and Information Technology, 2014(4),24–33. https://doi.org/10.26636/jtit.2014.4.1044
31. Kathy Wain Yee Au, Y. F. Z. Z. H. and D. L. (2013). Proceedings of the 2012 ACM conference on Computer and communications security. 217–228. https://doi.org/10.1145/2382196.2382222
32. I. Firdausi, C. Lim, A. Erwin, and A. S. Nugroho. (2010). "Analysis of machine learning techniques used in behavior-based malware detection," Proc. - 2010 2nd Int. Conf. Adv. Comput. Control Telecommun. Technol. ACT 2010, 201–203.
33. Alam, A. (n.d.). What is Machine Learning? https://doi.org/10.5281/zenodo.8231580
34. Mcdonald. J Todd, Herron, Nathan, Glisson. William Bradley, Benton. Ryan K (2021). "Machine Learning-Based Android Malware Detection Using Manifest Permissions," 6976- 6985. https://hdl.handle.net/10125/71460
35. Durmus¸O¨zkan Sahin, Og˘uz Emre Kural, Sedat Akleylek, Erdal Kılıç. (2023). "A novel permission-based Android malware detection system using feature selection based on linear regression," 35(1), 4903- 4903. https://doi.org/10.1007/s00521-021-05875-1
36. Nawaz Aqeel, Hussain Jawad, Jhanjhi NZ, Talib MN, Nadim Mohammad. (2021). "Feature engineering based on Hybrid Features for Malware Detection over Android Framework," 12(10), 2856-2864.
37. Lingru Cai, Yao Li, Zhi Xiong. (2021). "JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters," 100. https://doi.org/10.1016/j.cose.2020.102086
38. Songhao Lou*, Shaoyin Cheng†, Jingjing Huang*, Fan Jiang† (2019). "TFDroid: Android Malware Detection by Topics and Sensitive Data Flows Using Machine Learning Techniques," IEEE 2nd International Conference on Information and Computer Technologies, 30-36.
39. Abdullah Talha Kabakus. (2019). "What Static Analysis Can Utmost Offer for Android Malware Detection," Journal of Information Technology and Control, 48(2), 235-249. http://dx.doi.org/10.5755/j01.itc.48.2.21457
40. J. D. Koli. (2018). "RanDroid: Android Malware Detection Using Random Machine Learning Classifiers," IEEE International Conference on Technologies for Smart-City Energy Security and Power (ICSESP-2018).
41. BEŞTAŞ, M. Ş., & BATUR DİNLER, Ö. (2023). "Detection of Android Based Applications with Traditional Metaheuristic Algorithms". International Journal of Pure and Applied Sciences, 9(2), 381–392. https://doi.org/10.29132/ijpas.1382344
42. Sharma, S., P., Chhikara, R., & Khanna, K. (2023). "An efficient Android malware detection method using Borutashap algorithm".International Journal of Experimental Research and Review, 34, 86–96, https://doi.org/10.52756/ijerr.2023.v34spl.009
43. AlJarrah, M., Yaseen, Q., & Mustafa, A. (2022). "A Context-Aware Android Malware Detection Approach Using Machine Learning". In Information MDPI AG, 13(12), 563 https://doi.org/10.3390/info13120563
44. Yang, J., Zhang, Z., Zhang, H., & Fan, J. (2022). "Android malware detection method based on highly distinguishable static features and DenseNet". Public Library of Science (PLoS), 17(11), 0276332. https://doi.org/10.1371/journal.pone.0276332
45. Mahindru, Arvind, Sangal, A. L. (2022). "SOMDROID: android malware detection by artificial neural network trained using unsupervised learning". Evolutionary Intelligence, 15(1), 407-437 https://doi.org/10.1007/s12065-020-00518-1