International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

Machine Learning based Prediction Model of Android Malware: A Review

Raj Sagar Kumar[1], Dr. Ritesh Kumar Yadav[2]
[1]M.Tech Scholar, Department of CSE, Sarvepalli Radhakrishnan University, Bhopal, India
[2]Professor, Department of CSE, Sarvepalli Radhakrishnan University, Bhopal, India

Abstract— Android malware has become a major cybersecurity threat due to the widespread use of Android devices and the increasing number of malicious applications. Traditional signature-based detection techniques often fail to detect new, unknown, and obfuscated malware variants. This review paper presents an overview of Machine Learning (ML)-based prediction models for Android malware detection and classification. It discusses various machine learning techniques, including supervised, unsupervised, and deep learning approaches, along with commonly used datasets, feature extraction methods, and evaluation metrics. The paper compares existing ML-based models in terms of detection accuracy, efficiency, robustness, and their ability to identify previously unseen malware. It also highlights key challenges such as feature selection, data imbalance, malware obfuscation, adversarial attacks, and dataset limitations. Finally, the review identifies research gaps and discusses future directions for developing accurate, efficient, and reliable ML-based Android malware prediction models.

Keywords—Malware, Android, Machine Learning, Attack, Cyber.

I. INTRODUCTION

Android malware has become a major cybersecurity concern due to the rapid growth of Android smartphones and mobile applications. Android devices are widely used for communication, online banking, digital payments, shopping, social media, and storing personal information. As a result, attackers increasingly target Android platforms to steal sensitive data, monitor users, perform unauthorized transactions, and gain control over devices [1]. Android malware refers to malicious applications or software designed to perform unauthorized and harmful activities without the user's knowledge or permission.

There are several types of Android malware, including trojans, spyware, ransomware, adware, banking malware, and botnets. Trojans commonly disguise themselves as legitimate applications, while spyware secretly collects user information such as messages, contacts, location, and credentials. Ransomware can restrict access to files or devices, whereas banking malware targets financial information and banking applications [2]. The increasing complexity and diversity of these malware families make their detection a challenging task for security researchers.

Traditional Android malware detection systems mainly depend on signature-based techniques. These techniques compare an application with previously identified malware signatures. Although signature-based detection can effectively identify known malware, it has limitations when dealing with new, unknown, modified, or obfuscated malware [3]. Malware developers frequently modify malicious applications to avoid detection by changing their code, encrypting components, or using dynamic code loading. Therefore, more intelligent and adaptive detection techniques are required.

Static and dynamic analysis are two commonly used approaches for Android malware analysis. Static analysis examines an application without executing it and extracts features such as permissions, API calls, intents, and opcode sequences. Research has shown that application permissions and API calls can provide useful information for distinguishing malicious applications from benign applications [4]. The Drebin system is another important contribution that demonstrated the effectiveness of lightweight static features for detecting Android malware [5].

Dynamic analysis, on the other hand, executes an application in a controlled environment and observes its actual

behavior. It can monitor activities such as system calls, network communication, file access, and API usage. Dynamic analysis can identify malicious behaviors that may not be visible through static examination alone. However, it generally requires more computational resources and may fail to detect malware whose malicious behavior is triggered only under specific conditions [6].

Machine Learning (ML) has emerged as a promising solution for addressing the limitations of traditional Android malware detection methods. ML techniques can automatically learn patterns from large collections of benign and malicious applications and use these patterns to classify unknown applications. Commonly used algorithms include Decision Tree, Random Forest, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors, and Logistic Regression. These algorithms can use features such as permissions, API calls, system calls, application components, and network behavior for malware prediction [7].

The performance of machine learning models strongly depends on the quality of the dataset used for training and testing. The Drebin dataset is one of the most widely used datasets in Android malware research and contains thousands of malicious applications from different malware families [8]. Other datasets have also been developed to support research on Android malware detection. However, existing datasets may suffer from limitations such as outdated samples, class imbalance, duplicate applications, and insufficient representation of newly emerging malware. These limitations can affect the generalization ability of trained models.

Behavior-based machine learning techniques have also received considerable attention. MaMaDroid, for example, uses sequences of API calls to model application behavior and classify Android applications. Such behavioral approaches can reduce dependence on specific malware signatures and provide better protection against certain code modifications [9]. However, malware developers continuously introduce new techniques, making it necessary to regularly update and evaluate machine learning models.

Another major challenge is the use of obfuscation and adversarial techniques by malware developers. Obfuscation can hide malicious code or change application features without changing the actual malicious functionality. Similarly, adversarial modifications can manipulate the features used by machine learning models and cause malicious applications to be classified as benign [10]. Therefore, accuracy alone is not sufficient to evaluate an Android malware prediction model. Robustness, adaptability, computational efficiency, and the ability to detect previously unseen malware are also important factors.

Deep learning has recently become an important research direction in Android malware detection. Deep learning models can automatically learn complex relationships between application features and malware behavior. Techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and hybrid deep learning models have been investigated for improving detection performance [11]. Hybrid approaches that combine static and dynamic features can provide a more comprehensive representation of Android applications and may improve the detection of sophisticated malware [12].

II. LITERATURE SURVEY

Kauser et al. [1] proposed a hybrid deep learning model, DBN-GRU, for Android malware detection. The model combines Deep Belief Network (DBN) and Gated Recurrent Unit (GRU) to learn complex malware patterns. It uses both static and dynamic features for improved detection. The study reported high detection performance with better efficiency than conventional approaches. The work demonstrates the potential of hybrid deep learning for Android malware detection.

2. Kim et al. [2] developed a lightweight hybrid machine learning approach for real-time Android malware detection. Their method focuses on reducing computational requirements while maintaining reliable detection accuracy. The proposed approach is suitable for resource-constrained mobile devices. The study combines multiple machine learning techniques to improve classification performance. The results indicate that lightweight ML models can support real-time malware protection.

3. Alamro et al. [3] presented an optimal ensemble learning approach for automated Android malware detection. The proposed method combines multiple machine learning classifiers to improve prediction reliability. Feature optimization was used to identify relevant characteristics of malicious applications. The ensemble strategy achieved

effective malware classification performance. The study highlights the importance of combining classifiers for cybersecurity applications.

4. Zhu et al. [4] proposed SEDMDroid, a stacking ensemble framework for Android malware detection. The framework combines different machine learning models using a stacking strategy to improve classification accuracy. It utilizes multiple Android application features for identifying malicious behavior. The experimental results demonstrated the effectiveness of the ensemble approach. The study shows that stacking can improve the robustness of malware detection systems.

5. Gong et al. [5] investigated overlay-based Android malware detection at large market scales. The study focuses on detecting malicious applications that exploit overlay technologies to deceive users. The authors developed a systematic approach capable of adapting to changes in the Android technological environment. Their work addresses the scalability challenges of market-level malware detection. The study demonstrates the importance of adaptive detection methods for evolving Android threats.

6. Almomani et al. [6] proposed a hybrid evolutionary approach for detecting Android ransomware. The study specifically addressed the problem of highly imbalanced malware datasets. Evolutionary optimization techniques were combined with machine learning to improve ransomware classification. The proposed approach achieved effective detection despite the imbalance between benign and malicious samples. The research highlights the importance of handling imbalanced data in malware detection.

7. Mercaldo et al. [7] introduced a formal equivalence checking approach for mobile malware detection and family classification. The method uses formal techniques to compare application behaviors and identify similarities between malware samples. It can also support the classification of malware into different families. The approach provides a more systematic alternative to conventional signature-based methods. The study demonstrates the usefulness of formal methods in mobile malware analysis.

8. Vu and Jung [8] proposed AdMat, a CNN-based approach for Android malware detection and classification. The method represents Android application features in matrix form and applies a Convolutional Neural Network to learn important patterns. This reduces the need for extensive manual feature engineering. The approach was evaluated for both malware detection and family classification. The results demonstrate the potential of CNN-based models for Android malware analysis.

9. Gong et al. [9] investigated the deployment of machine learning for market-scale mobile malware detection. The study focuses on systematically integrating ML techniques into large-scale mobile application security systems. It addresses practical challenges such as scalability, detection efficiency, and changing malware characteristics. The proposed framework demonstrates the feasibility of applying ML to large application markets. The research provides useful insights into real-world deployment of mobile malware detection.

10. Yuan et al. [10] proposed a lightweight on-device Android malware detection method. The main objective was to perform malware detection directly on mobile devices while minimizing computational and resource requirements. The approach uses carefully selected features and machine learning techniques for efficient classification. The study demonstrated that effective malware detection can be performed without relying completely on cloud-based systems. This approach is particularly useful for real-time mobile security.

11. Liu et al. [11] presented a comprehensive review of Android malware detection approaches based on machine learning. The study examined different machine learning algorithms, feature extraction techniques, datasets, and evaluation methods. It also discussed major challenges in Android malware detection, including feature selection, dataset limitations, and evolving malware. The review provides an overview of the development of ML-based malware detection research. It also identifies several directions for future research.

12. Li and Li [12] investigated adversarial attacks and defense mechanisms against malware detection models using deep ensemble learning. The study demonstrated that attackers can manipulate malware characteristics to evade machine learning-based detection systems. It also examined defense strategies for improving model robustness against such

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 15, Issue 7, July 2026)

attacks. The research highlights the security risks associated with relying solely on conventional deep learning models. The findings emphasize the need for robust and attack-resistant Android malware detection systems.

III. CHALLENGES

Machine learning-based Android malware detection has shown significant potential, but several challenges limit its accuracy, reliability, and practical deployment. The rapidly changing nature of malware makes it difficult for models trained on older samples to detect new variants. Other major issues include limited and imbalanced datasets, high-dimensional features, malware obfuscation, adversarial attacks, computational limitations of mobile devices, difficulty in real-time detection, and lack of standard evaluation methods. In addition, models that achieve high accuracy on benchmark datasets may not always perform equally well in real-world environments. Addressing these challenges is essential for developing robust, scalable, and efficient Android malware prediction systems.

1. Evolving and Unknown Malware

Android malware continuously changes its structure and behavior to avoid detection. New malware families and variants may have characteristics that were not present in the training dataset. As a result, a machine learning model trained on historical samples may fail to identify previously unseen malware. This problem is also known as **concept drift**. Regular model updating and continuous learning are required to maintain detection performance against emerging threats.

2. Dataset Quality and Imbalance

The performance of an ML model strongly depends on the quality of its training dataset. Many Android malware datasets contain outdated samples, duplicate applications, mislabeled data, or limited malware families. Another major problem is **class imbalance**, where the number of benign applications is much larger than malicious applications, or vice versa. An imbalanced dataset can cause the model to favor the majority class and reduce its ability to detect minority malware samples. Therefore, balanced and representative datasets are required for reliable model training.

3. Malware Obfuscation

Malware developers commonly use obfuscation techniques to hide malicious code and change the appearance of applications without changing their functionality. Techniques such as code encryption, renaming, packing, reflection, and dynamic code loading can make static analysis difficult. Machine learning models that depend heavily on static features may therefore fail to detect highly obfuscated malware. Developing features that are resistant to obfuscation is an important research challenge.

4. Feature Selection and Extraction

Android applications contain a large number of features, including permissions, API calls, intents, system calls, opcode sequences, and network behavior. Selecting the most relevant features is challenging because irrelevant or redundant features can increase computational cost and negatively affect model performance. At the same time, removing important features may reduce detection accuracy. Effective feature selection techniques are therefore necessary to obtain a balance between accuracy, complexity, and detection speed.

5. Adversarial Attacks

Machine learning models can be targeted by attackers who intentionally modify malware features to evade detection. Such modifications may cause a malicious application to be classified as benign while preserving its harmful functionality. These attacks are known as adversarial attacks. They represent a serious challenge because even highly accurate models may become vulnerable when attackers understand the features used by the detection system. Robust training and adversarial defense mechanisms are required to improve model security.

6. Resource and Computational Limitations

Many Android devices have limited CPU, memory, battery, and storage resources compared with cloud or desktop systems. Complex deep learning models may require significant computational power and memory, making direct on-device deployment difficult. A model that provides high accuracy but consumes excessive resources may not be suitable for real-time mobile protection. Therefore, lightweight and optimized models are needed for practical deployment on smartphones.

7. Real-Time Detection and Scalability

Modern security systems need to detect malware quickly, preferably during application installation or execution. However, analyzing large numbers of applications and extracting multiple features can increase detection time. Large application markets also generate huge volumes of

applications that must be analyzed efficiently. Therefore, malware detection systems must be scalable enough to process large datasets while maintaining low detection latency and high accuracy.

8. Generalization, Explainability, and Evaluation

A model may achieve excellent results on a particular dataset but perform poorly when tested on applications from different sources or newer malware families. This indicates a lack of generalization. Additionally, complex deep learning models often operate as black boxes, making it difficult to understand why a particular application was classified as malicious. Standardized datasets, evaluation protocols, and explainable AI techniques are therefore needed to improve trust, reproducibility, and practical adoption of ML-based Android malware detection systems.

IV. CONCLUSION

Machine learning-based Android malware detection has emerged as an effective approach for identifying and predicting malicious applications by learning patterns from application features and behavior. This review shows that traditional machine learning, ensemble learning, and deep learning techniques can significantly improve malware detection compared with conventional signature-based methods. However, challenges such as evolving malware, limited and imbalanced datasets, code obfuscation, adversarial attacks, feature selection, computational limitations, and poor generalization still affect the reliability of existing models. Hybrid and lightweight approaches provide promising solutions by combining different features and reducing computational requirements. Future research should focus on developing adaptive, explainable, robust, and real-time models capable of detecting unknown malware while maintaining high accuracy and low resource consumption. Overall, machine learning provides a strong foundation for developing secure and intelligent Android malware prediction systems.

REFERENCES

1. Kauser.Sk H, Anu.V M (2025) Hybrid Deep Learning Model for Accurate and Efficient Android Malware Detection Using DBN-GRU. PLoS One 20(5): e0310230. https://doi.org/10.1371/journal.pone.0310230
2. Y. Kim, J. Seo, and H. Choi, “Lightweight Android malware detection using hybrid machine learning for real-time protection,” IEEE Access, vol. 12, pp. 145320–145332, 2024.
3. H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza and A. Y. Othman, "Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity," in IEEE Access, vol. 11, pp. 72509-72517, 2023, doi: 10.1109/ACCESS.2023.3294263.
4. H. Zhu, Y. Li, R. Li, J. Li, Z. You and H. Song, "SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection," in IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 984-994, 1 April-June 2022, doi: 10.1109/TNSE.2020.2996379.
5. L. Gong, Z. Li, H. Wang, H. Lin, X. Ma and Y. Liu, "Overlay-based Android Malware Detection at Market Scales: Systematically Adapting to the New Technological Landscape," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2021.3079433.
6. I. Almomani et al., "Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data," in IEEE Access, vol. 9, pp. 57674-57691, 2021, doi: 10.1109/ACCESS.2021.3071450.
7. F. Mercaldo and A. Santone, "Formal Equivalence Checking for Mobile Malware Detection and Family Classification," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2021.3067061.
8. L. N. Vu and S. Jung, "AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification," in IEEE Access, vol. 9, pp. 39680-39694, 2021, doi: 10.1109/ACCESS.2021.3063748.
9. L. Gong et al., "Systematically Landing Machine Learning onto Market-Scale Mobile Malware Detection," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 7, pp. 1615-1628, 1 July 2021, doi: 10.1109/TPDS.2020.3046092.

10. W. Yuan, Y. Jiang, H. Li and M. Cai, "A Lightweight On-Device Detection Method for Android Malware," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 9, pp. 5600-5611, Sept. 2021, doi: 10.1109/TSMC.2019.2958382.
11. K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," in IEEE Access, vol. 8, pp. 124579-124607, 2020, doi: 10.1109/ACCESS.2020.3006143.
12. D. Li and Q. Li, "Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3886-3900, 2020, doi: 10.1109/TIFS.2020.3003571.