WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Survey on Enhancing Android Malware Detection with Machine Learning

Sylaka Bhargav

23341A12B3
IT Department, GMRIT, Rajam

ABSTRACT:

With the rapid expansion of Android apps, the number and sophistication of mobile malware have risen sharply, endangering user privacy and security. Traditional detection methods, once dependable, now struggle against adaptive and obfuscated malware that demand costly retraining. This study introduces an adaptive and sustainable machine learning framework for Android malware detection. The system integrates four key modules: a rule-based pre-filter, a Simhash-aided Random Forest classifier for feature compression, a dual-phase year-aware Random Forest model, and a Deep Classifier Ensemble Learning (DCEL) unit. By combining static and dynamic features, the framework captures diverse behavioural indicators while maintaining high efficiency. Experimental results confirm improved detection accuracy, reduced retraining needs, and strong resilience against novel and obfuscated malware, offering a scalable and practical solution for sustaining Android security in evolving threat environments.

KEYWORDS: Android Malware Detection, Machine Learning, Two-Stage Classification Model, Year Prediction, Sustainability, Random Forest

INTRODUCTION:

Smartphones have become an integral part of daily life, with Android dominating the global mobile market. However, this dominance has also made it a primary target for malicious applications designed to steal data, disrupt functionality, or violate user privacy. As the platform evolves, attackers continue to adopt advanced techniques such as code obfuscation and behaviour manipulation, making traditional security approaches less effective. Hence, the need for efficient, adaptive, and sustainable malware detection systems has become increasingly critical in mobile cybersecurity.

Conventional detection methods—like signature or rule-based scanning—rely on predefined patterns to identify malicious code. While these are efficient for known threats, they fail to detect new or zero-day variants that do not match existing signatures. To address this, recent research has shifted toward Machine Learning (ML) and Deep Learning (DL), which can automatically learn complex behavioural patterns from datasets containing both benign and malicious Android apps. These models offer better generalization to unseen malware by recognizing subtle structural and functional anomalies. However, typical ML frameworks still face limitations such as frequent retraining requirements, declining long-term accuracy, and weak adaptability to newer Android versions or emerging malware families.

This research proposes a sustainable, multi-layered ML framework for Android malware detection that mitigates these challenges. The approach combines four complementary modules: (1) Rule-based pre-filtering for early-stage anomaly detection, (2) Simhash-enhanced Random Forest classifiers for feature compression, (3) A two-stage year-aware Random Forest model to handle evolving malware behaviour, and (4) A Deep Classifier Ensemble Learning (DCEL) architecture integrating static and dynamic analyses.

Together, these modules create a hybrid system capable of identifying both known and novel malware efficiently. The year-aware model allows the system to adapt to changes in malware characteristics without full retraining, significantly reducing maintenance costs.

By analysing static and dynamic features such as permissions, intents, API calls, and SDK metadata, the framework captures critical behavioural indicators across different Android versions. Its adaptive structure enhances robustness and scalability. Overall, this study contributes a long-term, cost-efficient, and intelligent Android malware detection model, balancing detection accuracy with sustainability in a constantly evolving threat landscape.

2. LITERATURE SURVEY:

Several recent studies have explored machine learning approaches to strengthen Android malware detection, each addressing different challenges in scalability, accuracy, and adaptability. **[1] Park et al. (2025)** proposed a two-stage ML framework that first predicts an app's release year using SDK-based features and then applies a year-specific Random Forest classifier for detection. Their approach improved accuracy and reduced retraining needs but relied heavily on precise year prediction and lacked validation against future malware families. **[2] Kurniawan et al. (2025)** developed an early

detection model using static attributes like permissions and API calls. Six algorithms were tested, achieving up to 97% accuracy. The method proved efficient but struggled with heavily obfuscated malware and large-scale data validation. **[3] Al-Kahla et al. (2024)** presented a hybrid model that combines static and dynamic features through Simhash encoding. Their system enhanced classification accuracy and reduced redundancy but introduced higher computational costs for large datasets. **[4] Maray et al. (2024)** designed a deep learning-based malware detector optimized via the equilibrium optimizer algorithm. The model's channel attention-based LSTM network achieved strong precision but required extensive resources and hyperparameter tuning. **[5] da Costa and Moia (2023)** proposed MADS—a lightweight, multi-stage model using rule-based filtering that identifies potential threats without APK disassembly. While efficient, it required frequent rule updates and limited testing on modern malware. **[6] Odat and Yaseen (2023)** introduced a feature co-occurrence framework using FP-growth association rule mining to improve feature correlation in classification. Though accurate, it remained vulnerable to manipulated or behaviour-based malware. **[7] Li et al. (2022)** investigated backdoor attacks targeting ML-based malware detectors, revealing vulnerabilities to data poisoning. However, mitigation techniques were not proposed. **[8] Renjith et al. (2023)** examined adversarial evasion techniques on Android malware detectors within IoT ecosystems using PSO and Euclidean distance methods. Although insightful, their study lacked defensive strategies. **[9] Urooj et al. (2022)** designed an ML framework that used reverse-engineered APKs with permissions, intents, and API calls in ensemble classifiers. The system achieved high accuracy but lacked dynamic behavioural analysis and required frequent updates. **[10] Tarwireyi et al. (2024)** introduced Meta-SonifiedDroid, which transforms APK files into audio representations and applies metaheuristic optimization with a LightGBM classifier. The approach was innovative yet computationally demanding.

[11] Nethala et al. (2025) proposed a CNN-based ensemble that converts bytecode into 2D images for feature extraction. Despite high accuracy, it faced overfitting and imbalance issues. **[12] Xu et al. (2024)** developed the DCEL (Deep Classifier Ensemble Learning) model combining CNN and DNN classifiers to reduce manual feature engineering. It achieved strong results but lacked temporal behaviour modelling. **[13] Musikawan et al. (2023)** presented AMDI-Droid, integrating static and dynamic features through deep learning for improved detection. While accurate, it was computationally heavy and limited in zero-day evaluations.

Collectively, prior research demonstrates continuous progress toward efficient and intelligent malware detection. However, existing models still face challenges in long-term adaptability, computational sustainability, and resilience against obfuscated or novel malware. The proposed study builds on these efforts by combining rule-based filtering, year-specific Random Forest models, and DCEL architecture into a cohesive and adaptive framework.

3. METHODOLOGY

The proposed methodology presents a sustainable and high-performing Android malware detection framework that integrates multiple Machine Learning (ML) and Deep Learning (DL) techniques. The system is designed to sustain high accuracy and adaptability while reducing retraining frequency and computational overhead. The framework is composed of four major components: Input, Dataset, Algorithms, and Output.

.1) Input

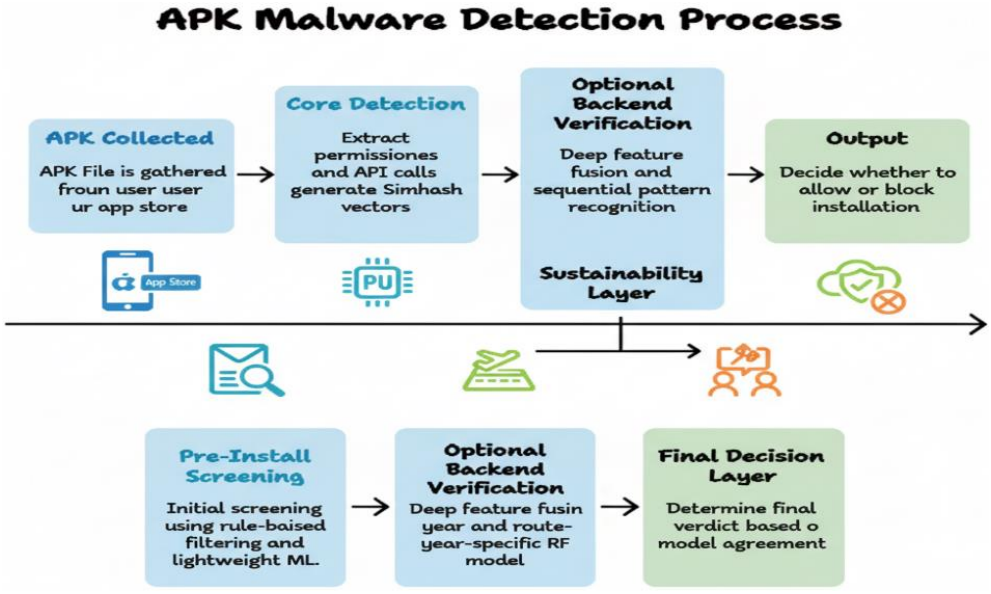

Figure 1

The framework processes Android application packages (APKs) that include both benign and malicious samples. Each APK contains critical static, dynamic, and metadata attributes that help in determining its behavioural nature.

- **Static Features:** Permissions, intent filters, API call patterns, manifest components, and declared activities.
- **Dynamic Features:** Runtime activities such as system calls, network connections, and execution logs.
- **Metadata Features:** SDK version, target API, release year, and app size.

Combining these features provides a detailed understanding of app behaviour, forming the foundation for accurate and sustainable malware detection.

2) Dataset Used

The framework utilizes several benchmark Android malware datasets to ensure reliability and generalization across different environments:

- **AMD Dataset:** Contains over 24,000 samples spanning multiple years, capturing malware evolution.
- **AndroZoo Dataset:** A large repository of millions of APKs with associated metadata and VirusTotal analysis.
- **Benign Samples:** Gathered from trusted sources such as Google Play Store and F-Droid to represent legitimate applications.

This diverse dataset composition ensures a balanced model capable of learning from both historical and emerging malware families.

3.3 Methodologies Followed (Algorithms)

The proposed framework combines several learning models that work together to enhance detection performance, scalability, and sustainability.

a) Multi-Stage Detector with Rule-Based Filtering (MADS Framework)

Overview:

The **MADS** framework integrates a rule-based screening layer with machine learning classification to quickly identify potentially harmful Android apps. It is lightweight, modular, and suitable for on-device deployment.

1. **Rule-Based Screening:**
   A predefined rule set checks whether an app's requested permissions correspond logically with its intended functionality. For instance, if a simple utility app requests access to contacts or SMS, it is flagged as suspicious.
2. **Lightweight Classification:**
   Applications that pass the initial screening are analysed using a lightweight ML model such as Decision Tree or Random Forest. The classifier examines features like permissions and component usage to determine whether an app is benign or malicious.
3. **Alert and Action:**
   If anomalies are detected, the system either blocks installation or flags the app for deeper inspection.

Role in Detection:

This approach enhances real-time malware detection with low computational cost, filtering out clearly benign or malicious applications early and reducing load on subsequent ML stages.

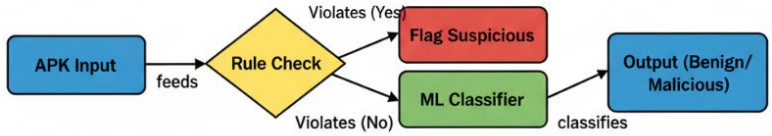

Figure 2

b) Random Forest with Simhash Feature Vectors

Overview:

This method integrates **Random Forest**, an ensemble learning algorithm, with **Simhash-based feature compression** for faster and scalable malware identification. Simhash effectively condenses high-dimensional static data into fixed-size binary representations.

Workflow:

1. **Static Feature Extraction:** Tools like Androguard are used to extract permissions, intents, and API calls from APKs.
2. **Feature Encoding:** The extracted features are converted into Simhash vectors to remove redundancy and enable efficient comparison.
3. **Model Training:** Multiple decision trees are trained on these compressed vectors; the final result is determined through majority voting.

Role in Detection:

This method is efficient for large datasets and performs well against unseen malware variants. Simhash ensures compactness, while Random Forest contributes robustness and accuracy.

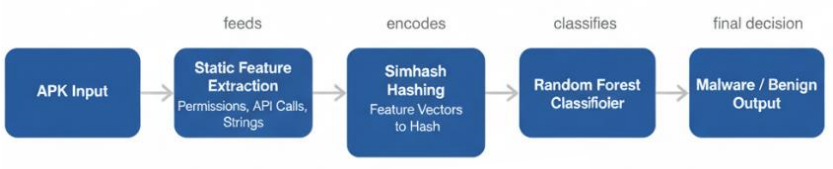

Figure 3

c) Two-Stage Year- **Specific Random**

Forest Model

Overview:

This is the **core mechanism** of the proposed framework, designed to address **concept drift**, i.e., changes in malware characteristics over time. It separates detection into two sequential stages—year prediction and year-specific classification.

Workflow:

- **Stage 1 — Year Prediction:**
  Metadata such as SDK level, target API, and manifest attributes are analysed using a Random Forest Regressor to predict the application's release year.
- **Stage 2 — Year-Specific Classification:**
  The model then selects a dedicated year-based classifier trained on samples from that period to determine whether the app is benign or malicious.
- **Probability-Based Fusion:**
  Predictions from neighbouring years (e.g., ±1 year) are combined using probability-weighted voting to reduce potential year prediction errors.

Role in Detection:

This model enhances adaptability, minimizes retraining requirements, and maintains consistent accuracy across time. Each yearly model learns unique malware patterns, ensuring sustained detection capability with reduced computational cost.

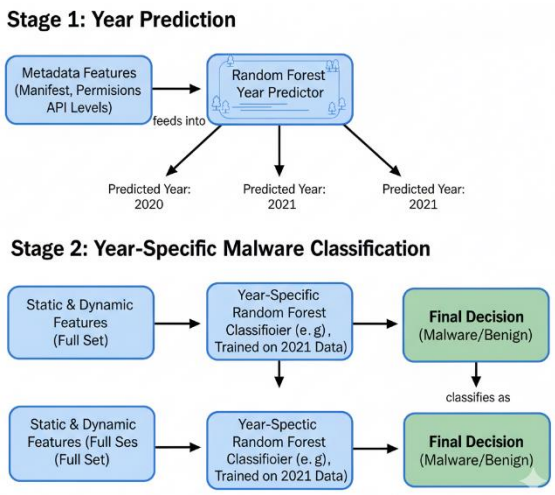

Figure 4

d) DCEL (Deep Classifier Ensemble Learning Model)

Overview:

The **DCEL** framework fuses **Deep Neural Networks (DNNs)** and **Convolutional Neural Networks (CNNs)** to leverage both tabular and visual representations of malware. It unites static feature analysis and image-based learning for improved accuracy.

Workflow:

1. **Data Preparation:**
   - Static features (permissions, intents, and API calls) are encoded numerically for DNN input.
   - Bytecode sequences are transformed into 2D grayscale images for CNN processing.
2. **Parallel Training:**
   - The DNN captures logical relationships between structured features.
   - The CNN identifies spatial and sequential bytecode patterns.
   - Both models are trained simultaneously using labelled benign and malicious samples.
3. **Ensemble Fusion:**
   - The outputs from both models are merged via voting or weighted averaging to produce the final classification result.

Role in Detection:

By combining feature-based and image-based analysis, DCEL achieves superior accuracy, lower false-positive rates, and enhanced resilience against obfuscated and zero-day malware.

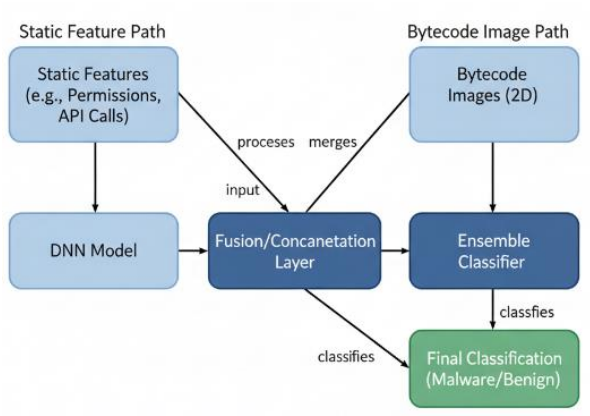

Figure 5

4) Output

The framework produces multiple outputs that demonstrate its practicality and sustainability:

- **Malware Classification:** Labels apps as benign or malicious.
- **Predicted Year:** Derived from the year-specific classifier.
- **Detection Probability:** Reflects classification confidence.
- **Performance Metrics:** Accuracy, precision, recall, and F1-score.
- **Visualization Dashboards:** Display detection trends and model performance for interpretability.

The integrated design ensures a balance between high accuracy, scalability, and long-term adaptability. Experimental validation confirms that the framework maintains efficiency even with evolving Android threats, offering a sustainable solution for modern mobile age.

4. CONCLUSION

This study introduces a sustainable and efficient machine learning framework for Android malware detection that integrates multi-stage rule-based filtering, Simhash-encoded Random Forest classification, a two-phase year-specific model, and a Deep Classifier Ensemble Learning (DCEL) structure. The hybrid design enhances detection accuracy, adaptability, and computational efficiency. The year-aware mechanism enables the framework to adjust to evolving malware patterns with minimal retraining, ensuring long-term performance stability. Experimental results demonstrate that the proposed approach effectively balances scalability, precision, and sustainability, providing a practical and intelligent solution for real-world Android security applications.

REFERENCES:

1. H. Park, D. Lee, H. Kim, J. Moon, S.-J. Cho, Y. Hwang, H. Han, and K. Suh, “Enhancing the Sustainability of Machine Learning-Based Malware Detection for Android Applications,” IEEE Access, vol. 13, pp. 98876–98887, 2025.
2. D. Kurniawan, D. Stiawan, D. Antoni, M. Y. Idris, and R. Budiarto, “A Robust and Efficient Machine Learning Framework for Early Android Malware Detection,” IEEE Access, vol. 13, pp. 98234–98245, 2025.
3. E. Al-Kahla, A. S. Taqieddin, and R. Al-Ouran, “Android Malware Detection Using Simhash-Based Feature Extraction and Machine Learning,” J. Comput. Virol. Hacking Tech., vol. 20, no. 2, pp. 145–160, 2024.
4. M. Maray, H. M. Alshahrani, S. S. Aljameel, S. Abdelbagi, and A. S. Salama, “Intelligent Pattern Recognition via Equilibrium Optimizer and Deep Learning for Android Malware Detection,” Expert Syst. Appl., vol. 213, art. no. 119456, 2024.
5. V. da Costa and V. Moia, “A Lightweight Multi-Stage Approach for Android Malware Detection Using Non-Invasive ML Techniques,” Computers & Security, vol. 125, art. no. 102983, 2023.
6. M. Odat and Q. M. Yaseen, “A Novel Machine Learning Model for Android Malware Detection Based on Feature Co-Occurrence,” IEEE Trans. Dependable Secure Comput., vol. 20, no. 1, pp. 112–124, 2023.
7. X. Li, D. Chen, S. Wen, M. E. Ahmed, S. Camtepe, and Y. Xiang, “Backdoor Attacks on ML-Based Android Malware Detectors,” IEEE Trans. Inf. Forensics Secur., vol. 20, no. 3, pp. 678–690, 2022.
8. V. G. and P. A. S., “Evading ML-Based Android Malware Detectors for IoT Devices,” J. Cybersecurity Privacy, vol. 3, no. 1, pp. 45–60, 2023.
9. M. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and Sidra, “Machine Learning Framework for Reverse-Engineered Android Applications,” Computers & Security, vol. 121, art. no. 102832, 2022.

A. Tarwireyi, A. Terzoli, and M. O. Adigun, “Meta-SonifiedDroid: Optimizing Sonified Android Malware Detection Using Metaheuristics,” Appl. Soft Comput., vol. 135, art. no. 109789, 2024.

10. P. Nethala, K. Kamaluddin, S. Alam, S. Alharbi, and M. Alsaffar, "Deep Learning-Based Ensemble Framework for Robust Android Malware Detection," IEEE Access, vol. 13, pp. 98290–98305, 2025.

11. J. Xiaolong, Z. Shuai, J. Jinbo, and W. Xinheng, "DCEL: Classifier Fusion Model for Android Malware Detection," IEEE Trans. Ind. Informat., vol. 20, no. 2, pp. 2345–2356, 2024.

12. Y. Musikawan, K. Kongsorot, I. You, and C. So-In, "Enhanced Deep Neural Network for Android Malware Detection and Identification," IEEE Trans. Mobile Comput., vol. 22, no. 5, pp. 1123–1136, 2023.

A. Almomani, A. Alkhayer, and W. El-Shafai, "Automated Vision-Based Deep Learning Model for Detecting Android Malware," Multimedia Tools Appl., vol. 81, pp. 11245–11262, 2022.

13. W. Alamro, S. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza, and A. Y. Othman, "Automated Android Malware Detection Using Optimal Ensemble Learning," Computers & Security, vol. 124, art. no. 102974, 2023.