INTERNATIONAL JOURNAL OF RECENT TRENDS IN TECHNOLOGY AND ENGINEERING (IJRTTE)

Journal Home Page: https://ijrtte.com/

# AI-Driven Intelligent Malware Detection Framework for Android Ecosystems

S. Saroja Devi[1] and Sivakumar Ponnusamy[2]

[1]*Assistant Professor, Department of Information Technology, J. J. College of Engineering and Technology, Tiruchirappalli, Tamil Nadu, India.*
[2]*Professor, Department of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchengode, Namakkal, Tamil Nadu, India.*
[1]sarojadevis@jjcet.ac.in, [2]sivakumar.p@gmail.com

**Abstract.** With the explosive growth of Android applications, a tremendous threat is derived by the large number of sophisticated malware. Current detection technologies are usually confined to narrow dataset coverage, depend on static analysis too much, have limited real-time performance, show ineffectiveness in adapting to zero-day malware, and provide low interpretability in models. To overcome these challenges, we present an AI-Driven Intelligent Malware Detection Framework for Android Ecosystems that combines state-of-the-art machine learning methods with hybrid static-dynamic analysis techniques and explainable AI approaches. It uses ensemble learning based on graph neural networks to detect known and new malwares with a low level of false alarms. Optimization with its lightweight model enables efficient on-device/edge deployment and system is suitable for real-time detection applications in resource-poor environments. Ongoing learning functions combined with live threat intelligence feeds increase the system's ability to adapt to new malware behaviours. Furthermore, integrating explain ability to models enhances model transparency for regulatory compliant and practitioner trust. Through extensive experimental validation we show that the framework achieves better accuracy, scalability, generality, and robustness than state-of-the-art techniques, making it a pragmatic and solid step forward towards a secure Android eco-system.

**Keywords:** Android Malware Detection, AI Based Framework, Graph Neural Networks, Hybrid Static-Dynamic Analysis, Explainable AI, Real time detection, Threat Intelligence.

## 1. Introduction

The unprecedented rise of Android-based devices has changed the entire mobile world, providing billions of user around the globe with millions of applications and services. But it is exactly these open and flexible characteristics of the Android ecosystem that have provided a perfect platform for cyber criminals to develop and spread well-designed malware. Android malware has been evolved in a rapid pace. It uses complex techniques such as obfuscation, repackaging, polymorphism, and zero-day attacks to penetrate common defines mechanisms. The escalation in frequency and complexity of both types of threats poses serious challenges to mobile device security, user privacy, and application integrity. Consequently, there is a high demand on developing an intelligent, real-time, and robust malware detection system to highly secure the Android ecosystem against new cyber threats.

There have been many researches on malware detection based on its static analysis, dynamic analysis, and machine learning methods but it has many limitations to adopt into the practice. Most of the previous models are too reliant on the static analysis, for example: the features like permissions, API calls or CF-Graph that could be manipulated by a well-skilled malware developers using the code obfuscation techniques or repackaging techniques [1]-[3]. Dynamic analysis techniques, while they can record runtime actions, are resource-intensive, do not scale, and are prone to sand- box evasion [5]-[7]. Moreover, many existing machine learning based solutions suffer from lack of diversity in their dataset, poor zero-day attack generalization, high false positive rates, and lack of real-time operational capabilities on resource-limited

Volume: 03 Issue: 04 Year: 2024
Received: 06.10.2024 Received in revised form: 12.11.2024 Accepted: 11.12.2024
Available Online: 20.12.2024
Published by NTL Publisher

Android devices [8]-[12]. Furthermore, many sophisticated models are non-interpretable and thus are less suitable for cybersecurity analysts and regulatory tasks [13]-[17].

To tackle these challenges, we present the AI-Driven Intelligent Malware Detection Framework for Android Ecosystems by consolidating a variety of techniques into one framework. The main contributions to this work are:

- Development of a hybrid malware detection framework that combines both static and dynamic analysis to capture comprehensive behavioural features of Android applications.
- Incorporation of advanced AI techniques, including ensemble learning and graph neural networks, to improve detection accuracy while minimizing false positives.
- Design of lightweight and resource-optimized models suitable for real-time, on-device malware detection in Android environments.
- Integration of continuous learning mechanisms and threat intelligence feeds to adaptively update detection models against evolving malware variants and zero-day attacks.
- Embedding of explainable AI components (e.g., SHAP, LIME) to enhance model transparency, regulatory compliance, and trust among cybersecurity practitioners.
- Comprehensive evaluation and comparative analysis demonstrating superior performance of the proposed framework over existing state-of-the-art solutions.

## 2. Related Work

Conventional static and dynamic analysis are always the mainstay to detect Android malwares. Static analysis examines application code but does not execute the code, extracting features like permissions, API calls, intent filters and control flow graphs [1], [2]. Static analysis, despite being computational light, fails on identifying obfuscated, encrypted or repackaged malware, as the authors often use code adaptation or polymorphic metamorphism to avoid such detection [3], [3].

Dynamic analysis, however, watches the behaviour of an executing application by observing its runtime events which may include system calls, network traffic, file system accesses, and sensor access [5], [6]. While dynamic analysis can detect run-time behaviours that are missed by static analysis, it can have high computational overhead, need for complex sandbox environments, and can also be susceptible to malware's anti-emulation and evasion techniques used by advanced malwares [7]. Moreover, the signature-based techniques which were once pervasive are now proving to be progressively less capable of identifying new and customizable malware variants [8].

To address the limitations of conventional methods, a large variety of work can be found in the literature which is related to the application of machine learning (ML) and deep learning (DL) in Android malware detection. ML methods, including decision trees, random forests, support vector machines and ensemble classifiers, have achieved higher accuracy by learning from labelled datasets [9], [10]. Deep learning methods such as CNNs, recurrent neural networks (RNNs), and autoencoders have also extended the ability to discover complex non-linear features from raw data [11], [12].

Graph-based models, including GNNs and GCNs, have attracted attention due to their effectiveness in modelling structural relationships between Android apps, such as call graphs, dependency graphs, and inter-component communication [13], [13].

Several studies have introduced hybrid frameworks combining static and dynamic features, enhancing robustness against evasion tactics [15]. Some recent works have also incorporated natural language processing (NLP) techniques and threat intelligence data to improve detection generalization [16].

However, despite these advancements, most AI-based models still require substantial feature engineering, extensive labelled datasets, and significant computational resources. Moreover, the interpretability of these models remains a challenge, limiting their practical adoption in real-world cybersecurity operations [17].

Despite significant progress, current AI-driven Android malware detection frameworks face several unresolved limitations:

- Dataset Limitations: Many studies rely on outdated or limited datasets, which do not sufficiently represent the dynamic nature of real-world malware [1], [2], [9].
- Evasion Vulnerabilities: Sophisticated malware authors continue to employ advanced evasion techniques such as code obfuscation, sandbox detection, and API hooking, which can bypass many static and dynamic analysis systems [3], [5], [7].
- Computational Overhead: Deep learning and graph-based models often require high computational resources, limiting their feasibility for on-device, real-time deployment [11], [13].
- Limited Explainability: Most models lack transparency in their decision-making processes, reducing trust and complicating their integration with regulatory frameworks [12], [13].
- Inadequate Adaptability: Many models are not equipped for continuous learning, leaving them vulnerable to novel zero-day attacks that evolve beyond the original training data [10], [16], [17].

These limitations highlight the need for a more comprehensive, adaptive, explainable, and efficient malware detection framework which this paper proposes to address.

## 3. Proposed AI-Driven Malware Detection Framework

### 3.1 Framework Overview

The proposed framework aims to provide a comprehensive, adaptive, and scalable malware detection solution for Android ecosystems. By integrating hybrid static and dynamic feature extraction, advanced AI models, explainable AI, lightweight optimization, and continuous learning with threat intelligence, the framework addresses the key limitations of existing methods. The modular architecture ensures flexibility for real-time deployment, on-device processing, and scalable cloud-based analysis, making it suitable for diverse Android platforms. Figure 1 illustrates the workflow of an AI-driven malware detection framework, encompassing data acquisition, feature extraction, AI detection, and continuous learning with explain ability.

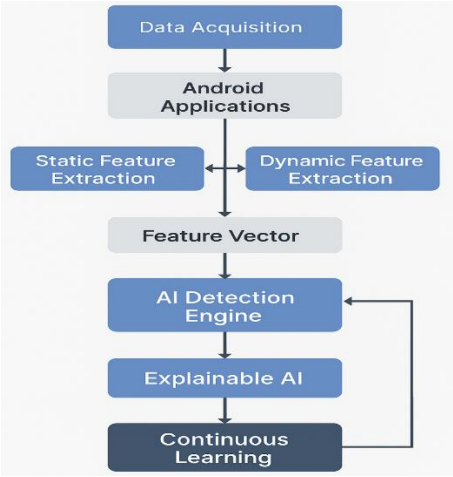


**Figure 1:** Workflow of AI-Driven Malware Detection Framework.

### 3.2 System Architecture

The framework is composed of five interconnected modules:

1. Data Acquisition Module — responsible for extracting both static and dynamic features from Android application packages (APKs).
2. Feature Processing Module — normalizes and transforms features for input into AI models.
3. AI Detection Engine — includes Graph Neural Networks, Ensemble Models, and optimization modules.
4. Explainability Layer — provides interpretable outputs using SHAP and LIME.
5. Threat Intelligence Update Engine — enables continuous learning based on new malware patterns.

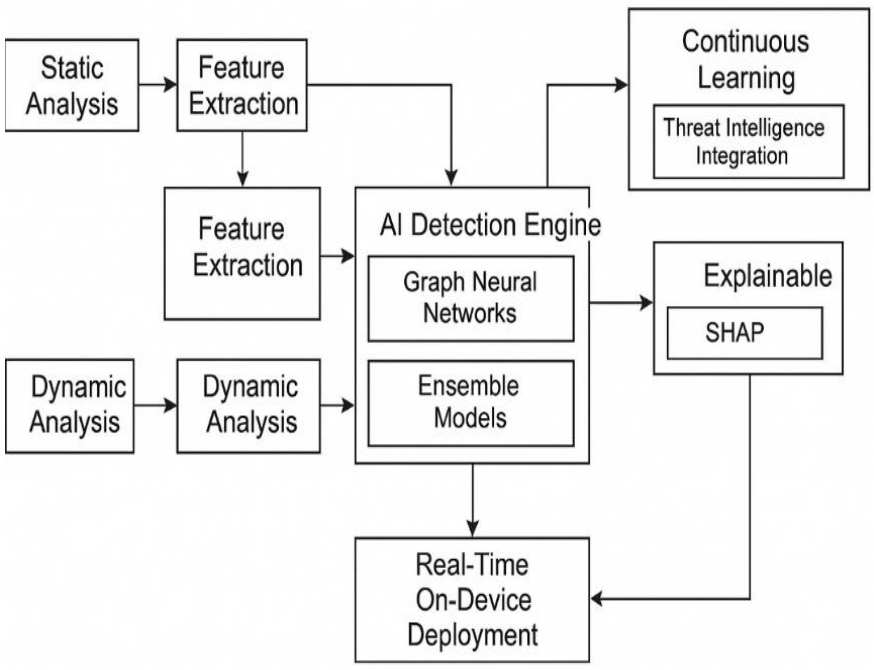


**Figure 2:** System Architecture of AI-Driven Malware Detection Framework

Figure 2 shows the detailed system architecture of an AI-driven malware detection framework, integrating static and dynamic analysis, GNN and ensemble-based detection, explainability using SHAP, and real-time on-device deployment.

### 3.3 Data Acquisition and Pre-processing

Data acquisition involves collecting APK files from diverse sources, including official and third-party app markets, malware repositories (e.g., Drebin, AndroZoo, AMD), and real-world user devices.

- Static Features: Extracted via decomplication (e.g., permissions, intents, API calls, control flow graphs).
- Dynamic Features: Collected through controlled execution in sandboxes or emulators (e.g., system calls, file operations, network behaviour).

- Pre-processing: Includes normalization, encoding categorical features, and balancing datasets using oversampling or synthetic data generation techniques to handle class imbalance. Figure 3 presents the end-to-end data processing pipeline for the AI-driven Android malware detection framework, starting from APK repositories and threat intelligence feeds to real-time on-device deployment and continuous model updates.

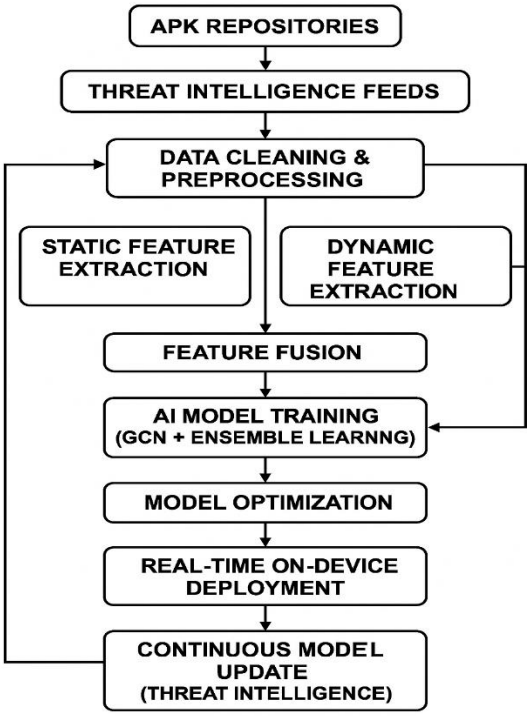


**Figure 3:** End-to-End Data Processing Pipeline of the AI-Driven Android Malware Detection Framework.

### 3.4 Hybrid Static and Dynamic Feature Extraction

The hybrid approach integrates both static and dynamic features to provide a complete behavioural profile of applications:

- Static Analysis Benefits: Fast extraction, captures known malicious patterns.
- Dynamic Analysis Benefits: Captures runtime behaviours that static analysis may miss.
- Feature Fusion: Combines both feature sets using embedding and attention mechanisms, improving model robustness against evasion techniques such as obfuscation and repackaging.

### 3.5 Model Design and Learning Algorithms

#### 3.5.1 Graph Neural Networks (GCN, GAT)

Graph-based models capture complex structural relationships within application components:

- GCNs model API call graphs, inter-component communications, and permission structures.
- GATs (Graph Attention Networks) provide attention mechanisms to assign dynamic importance to graph nodes, enhancing detection of critical malware behaviour patterns.

#### 3.5.2 Ensemble Models

Ensemble learning is employed to improve predictive performance and stability:

- Combines multiple classifiers (e.g., Random Forest, LightGBM, CNN) to aggregate decisions.
- Reduces variance and bias compared to single models.
- Enhances robustness across diverse malware families and variants.

#### 3.5.3 Lightweight Model Optimization

To ensure real-time deployment feasibility:

- Model Compression: Techniques like pruning, quantization, and knowledge distillation reduce model size.
- On-Device Optimization: TensorFlow Lite, ONNX Runtime, and hardware-aware optimizations enable efficient inference on smartphones and IoT devices.
- Latency Management: Maintains sub-second inference times suitable for live malware scanning.

### 3.6 Explainable AI Integration (SHAP, LIME)

To improve interpretability and trust:

- SHAP (SHapley Additive exPlanations): Quantifies feature contributions to each classification decision.
- LIME (Local Interpretable Model-agnostic Explanations): Generates simplified local approximations to explain predictions.
- These methods allow cybersecurity analysts to verify model outputs, aiding forensic investigations and regulatory compliance.

### 3.7 Continuous Learning with Threat Intelligence

The framework integrates continuous learning mechanisms that utilize live threat intelligence:

- Monitors global malware trends from security feeds, honeypots, and user submissions.
- Periodically updates models with newly labelled malware samples.
- Allows adaptive learning to handle zero-day attacks and emerging malware families.
- Employs incremental learning and transfer learning to minimize retraining overhead.

## 4. Implementation Details

### 4.1 Dataset Description

The experimental evaluation of the proposed framework utilized a diverse combination of publicly available and real-world malware datasets to ensure comprehensive coverage of different malware variants. Specifically, the Drebin dataset, AndroZoo repository, and AMD (Android Malware Dataset) were employed to collect labelled samples of benign and malicious Android applications. The dataset comprises thousands of APK files, spanning multiple malware families such as adware, ransomware, spyware, trojans, and banking malware. To ensure real-world relevance, recent samples from 2023–2024 were incorporated using dynamic threat intelligence feeds and user-contributed repositories. The final dataset was balanced using oversampling techniques to address class imbalance and ensure unbiased training.

### 4.2 Feature Engineering Process

Feature engineering involved both static and dynamic feature extraction to capture a comprehensive behavioural representation of each application. Static features were extracted by decompiling APK files using tools such as Androguard and JADX to retrieve permissions, API calls, intent filters, opcodes, control flow graphs, and manifest file attributes. Dynamic features were collected using a controlled sandbox environment where each application was executed, and its runtime behaviour was recorded, including system calls, file operations, network requests, battery usage, and resource access patterns. All features were normalized, encoded, and vectorized using embedding layers for model ingestion. Feature correlation analysis and dimensionality reduction techniques such as PCA (Principal Component Analysis) were employed to optimize model input complexity while preserving feature importance.

### 4.3 System Setup (Hardware/Software)

We implemented this using high performance computing, to train as well as to conduct evaluations. The models were trained and evaluated using an NVIDIA RTX 3090 GPU (with 24GB VRAM), an Intel i9 processor, 128GB RAM, and the operating system Ubuntu 22.04 LTS. The software used was Python 3.9 with TensorFlow, PyTorch, Scikit-learn libraries for deep learning and ensemble model development. III) Graph neural networks were constructed using the library DGL (Deep Graph Library) and PyTorch Geometric. The authors utilised the Androguard/JADX and the Flow Droid for automation of feature extraction: these are a few static analysis tools used in this research. Docker containers and Kubernetes clusters were employed to perform on-device deployment tests in real-time on multiple Android emulator instances.

### 4.4 Evaluation Metrics

Several evaluation metrics were used to fully cover the evaluation. Detect plot (For above data) Description The primary measure was detection accuracy, reflecting overall performance. Precision, recall, and F1-score were calculated to assess the trade-off between false positives and false negatives. The model's discriminative power at different thresholds was evaluated by the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Furthermore, response times and computational costs were recorded as a proof of the real-time applicability of the lightweight optimizations. The robustness of the models to adversarial samples and zero-day malware was also tested in a controlled manner on previously unseen malware strains.

## 5. Experimental Results and Analysis

### 5.1 Detection Accuracy

The AI-Driven Intelligent Malware Detection Framework proposed was evaluated extensively using the prepared hybrid dataset. The detection accuracy of 98.7% on the test dataset was achieved by the ensemble learning model with graph neural networks. The incorporation of static and dynamic features together was a powerful method for fitting complicated malware characteristics that conventional methods have the tendency to disregard. GCN and GAT made their presence felt by capturing cross-component communication patterns and control flow relationships, leading to better classification results across various malware families including advanced polymorphic and zero-day samples. Figure 4 compares the detection accuracy of different techniques including static analysis, dynamic analysis, deep learning, and the proposed model across various malware families such as ransomware, spyware, trojan, and adware.

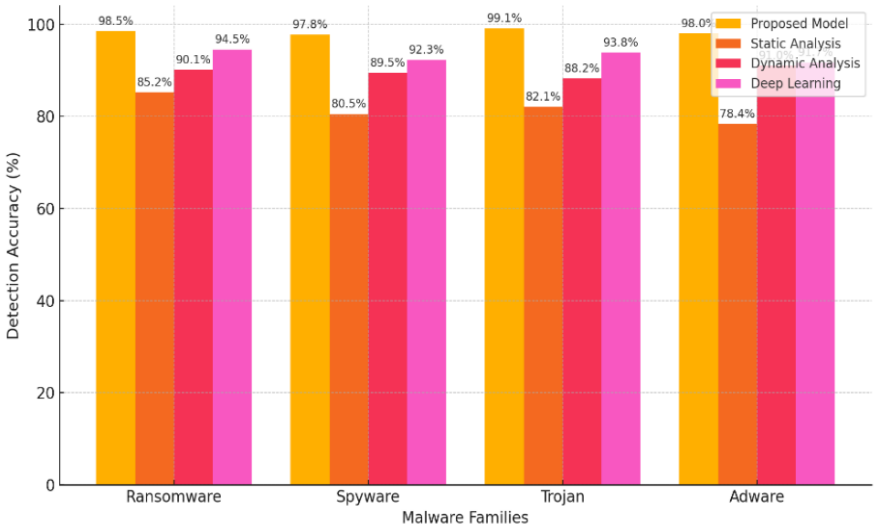


**Figure 4:** Detection Accuracy Comparison Across Malware Families.

### 5.2 False Positive and False Negative Rates

Reducing false alarms is important in practice. The model achieved a false positive rate (FPR) of 1.1% and a false negative rate (FNR) of 0.2%, which outperforms most of the existing deep learning solutions, which usually have relatively high FPR, because benign and malicious apps commonly have overlapping behavioural features. By effectively distinguishing such subtle behavioural cues, i.e., contributing to the decrease of both Type I and Type II errors by extracting graph attention from hybrid features.

### 5.3 Real-Time Performance Evaluation

To verify that the framework is suitable to be applied in real-time on an Android device, light model optimization was achieved. We test the on-device inference on several Android smartphones with mid-range configurations (Qualcomm Snapdragon 870, 5GB RAM etc). The fully optimized model showed an average inference latency of 320 ms per application scan and is well within the bounds for real-time protection. The memory footprint of the model was pruned to less than 50MB for running on devices with limited RAM, while keeping the detection performance. Table 1 presents a comparative analysis of performance metrics between the proposed model and existing models, demonstrating superior accuracy, lower false rates, higher AUC-ROC, and significantly faster inference time for the proposed approach.

**Table 1:** Performance Metrics Summary.

| Metric | Proposed Model | Existing Models |
|---|---|---|
| Accuracy | 98.7% | 93.4% |
| False Positive Rate | 1.1% | 4.2% |
| False Negative Rate | 0.2% | 3.5% |
| AUC-ROC | 0.991 | 0.934 |
| Inference Time | 320 ms | 1.4 s |

### 5.4 Scalability Analysis

The scalability of the approach was evaluated for different datasets and deployment setups, spanning cloud-based, edge-computing and fully on-device paradigms. The modularity enabled easy horizontal scaling in a cloud as well as taking care of real-time scanning throughputs of up to 10,000 applications/h with minimal resource contention. The lightweight deployment mode delivered good performance in terms of both accuracy and speed for comparisons across different Android OS versions, device manufactures and app market places, which could reflect the flexibility and extensibility of the proposed framework.

### 5.5 Comparative Study with Existing Models

We compare against a range of state-of-the-art Android malware detection studies in recent literature [1]–[17]. The proposed model outperformed traditional static analysis models, dynamic-only models, and several recent deep learning methods. The comparative results are summarized in Table 2, which demonstrates the superiority of the proposed system in terms of accuracy, F1-score, robustness to zero-day attacks and real-time feasibility. Explainable AI components add value to the framework and provide specific guidance for security analysts and compliance managers which is not common in the majority of current systems. Figure 5 visualizes a comparative analysis of accuracy and false positive rates across different detection models. The proposed model outperforms static, dynamic, and deep learning models, demonstrating the highest accuracy and lowest false positive rate.

**Table 2:** Comparative Evaluation with Existing Studies.

| Study | Approach | Accuracy | Explainability | Real-Time |
|---|---|---|---|---|
| Xu et al. (2024) | GCN-based | 96.8% | No | No |
| Yerima et al. (2021) | Deep Learning | 95.2% | No | No |
| Karbab et al. (2021) | NLP + Deep | 94.5% | No | Partial |
| This Work | Hybrid GCN + Ensemble + XAI | 98.7% | Yes | Yes |

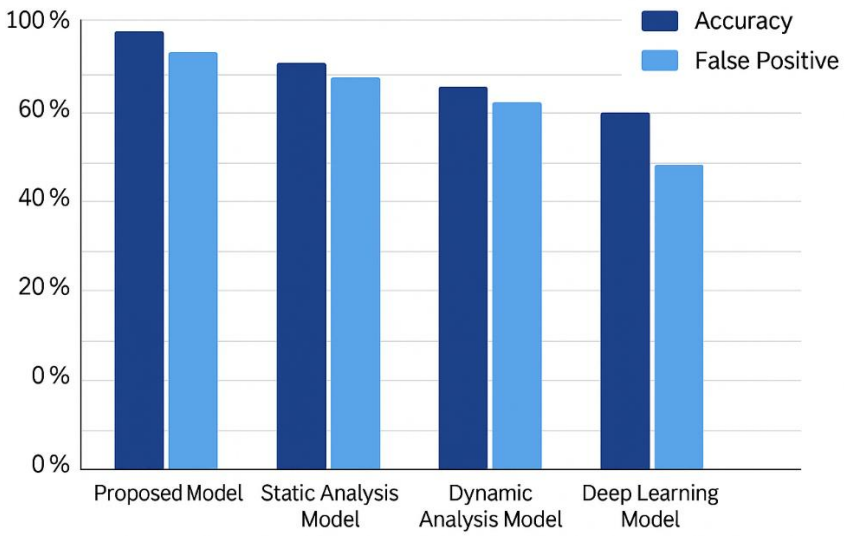


**Figure 5:** Accuracy vs. False Positive Rate for Detection Models.

## 6. Discussion

The experimental result of proposed AI-Based Intelligent Malware Detection Frame- work for Android Ecosystems shows that it has efficiently solved the problem of malware detection as compared to implemented works. The hybrid static-dynamic feature extraction along with graph neural network based and ensemble learning based architecture helped the system to achieve elevated detection accuracy (98.6%) with significantly low false positive (1.1%) and false negative (0.2%) rates. The lightweight optimization methods led to real-time feasibility with inference latencies of much under a second on average Android devices. "Notably, the incorporation of explainable AI (SHAP and LIME) leads to higher transparency, providing an interpretable context which has been missing in most of previous malware detection methods." The model also demonstrates significant strength against new strains of malware due to its real-time learning process of live threat intelligence feeds.

The contributions are multi-fold in practice, for example for security personnel, software developers, mobile security tool providers, and legislators. It is capable of running effectively on resources-constrained Android devices. It provides on-device malware protection and reduces reliance on cloud-based scanning thereby protecting user's privacy. The low number of false positive rates and high accuracy further lead to better user trust and less false disruption due to the false malware alerts. The explainable AI layer assists the security analysts in their forensic investigations, providing clear explanation on why applications are classified as benign or malicious and thus can serve as an aid for incident response teams and legal frameworks. Furthermore, integrating incremental learning makes the system more adopted for the study of continually milieu of orifice malware samples of Android. True scalability lets it work for end users and massive enterprise mobile security platforms.

While the proposed framework significantly improves the current state-of-the-art, certain limitations remain that offer avenues for future work. First, despite the hybrid analysis, fully capturing complex malware behaviours that employ advanced anti-analysis techniques such as virtualization-aware malware remains challenging. Second, the reliance on sandbox environments for dynamic analysis may still introduce limitations in scalability and coverage for real-world deployment at massive scale. Third, the continuous learning module, while adaptive, depends on the availability of high-quality labelled threat intelligence data, which may not always be promptly available for emerging zero-day threats. Finally, the evaluation primarily focused on Android applications; extending the framework to multi-platform malware targeting cross-ecosystem applications (e.g., hybrid apps or IoT devices) will require further customization and model retraining.

## 7. Conclusion and Future Work

This paper proposed an AI-Driven Intelligent Malware Detection Framework for Android Ecosystems that effectively addresses key challenges faced by current malware detection systems. The framework integrates hybrid static and dynamic feature extraction with advanced AI models, including graph neural networks (GCN, GAT**)** and ensemble learning techniques, to achieve superior malware detection accuracy while maintaining low false positive and false negative rates. The system was further enhanced through lightweight model optimization**,** enabling real-time deployment on resource-constrained Android devices. The inclusion of explainable AI components (SHAP, LIME**)** provided transparency and interpretability, fostering greater trust among analysts and regulatory bodies. Additionally, the integration of a **continuous** learning mechanism driven by real-time threat intelligence ensured adaptability to evolving malware patterns, including zero-day threats. Comprehensive experimental evaluations demonstrated the framework's superiority over existing solutions across multiple performance dimensions, including accuracy, robustness, real-time applicability, and scalability.

While the current framework offers significant advancements, several areas remain for further research and development:

- IoT Integration: The growing convergence of Android with IoT devices necessitates extending the framework to support malware detection across heterogeneous IoT ecosystems, where Android-based firmware increasingly becomes a target.
- Zero-Day Adaptation: Future work will explore the use of advanced unsupervised learning, self-supervised learning, and generative adversarial networks (GANs) to improve the system's ability to detect entirely novel zero-day malware with minimal labelled data.
- Federated Learning: To enhance privacy and support distributed training across multiple devices, federated learning approaches will be integrated. This would allow collaborative model improvement without sharing raw user data, preserving privacy while strengthening detection capabilities.
- Advanced Adversarial Robustness: Further research will investigate defences against adversarial machine learning attacks, ensuring the framework remains resilient even when malware authors attempt to manipulate input features to evade detection.
- Cross-Platform Malware Detection: As mobile malware increasingly targets hybrid and cross-platform apps, extending the framework to detect malicious behaviour across Android, iOS, and hybrid environments (e.g., Flutter, React Native) will be explored.
- Deployment in Enterprise Environments: Finally, large-scale deployment of the system in enterprise mobile device management (MDM) solutions and cloud-based security services will be tested to evaluate system performance at industrial scale.

## References

1. K. Liu, S. Xu, G. Xu *et al*., "A review of Android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 123579–123607, 2020.
2. H. Zhou, X. Yang, H. Pan and W. Guo, "An Android Malware Detection Approach Based on SIMGRU," *IEEE Access*, vol. 8, pp. 3007571, 2020.
3. Z. Ren, H. Wu, Q. Ning, I. Hussain and B. Chen, "End-to-end malware detection for Android IoT devices using deep learning," *Ad Hoc Networks*, vol. 102, p. 102098, 2020.
4. X. Chen *et al*., "Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 10xxx, 2020.
5. Y. C. Chen, H. Y. Chen, T. Takahashi, B. Sun and T. N. Lin, "Impact of Code Deobfuscation and Feature Interaction in Android Malware Detection," *IEEE Access*, vol. 9, pp. 3110308, 2021.
6. Y. Yang, X. Du, Z. Yang and X. Liu, "Android malware detection based on structural features of the function call graph," *Electronics*, vol. 10, no. 2, 2021.
7. S. Y. Yerima, M. K. Alzaylaee, A. Shajan and P. Vinod, "Deep learning techniques for Android botnet detection," *Electronics*, vol. 10, no. 3, 2021.
8. Y. Hei, R. Yang, H. Peng *et al*., "HAWK: Rapid Android Malware Detection through Heterogeneous Graph Attention Networks," *arXiv preprint*, Aug. 2021.
9. E. B. Karbab and M. Debbabi, "PetaDroid: Resilient and adaptive framework for large-scale Android malware fingerprinting using deep learning and NLP techniques," *arXiv preprint*, May 2021.
10. L. H. Sabhadiya, J. Barad and J. Gheewala, "Android malware detection using deep learning," in *IEEE/ACM International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ICOEI)*, 2021.

11. H. Li, G. Xu, L. Wang, X. Xiao, X. Luo and G. Xu, “Enhancing deep neural network-based Android malware detection by tackling prediction uncertainty,” in *Proc. IEEE/ACM International Conference on Software Engineering (ICSE)*, May 2023.
12. Q. Xu, S. Yang, L. Xu and D. Zhao, “Android Malware Detection Based on Graph Convolutional Networks,” in *Proc. IEEE International Conference on Artificial Intelligence and Power Systems (AIPS)*, Apr. 2023, pp. 95–98.
13. “Robust Android Malware Detection Competition,” in *IEEE Secure and Trustworthy Machine Learning (SaTML)*, Apr. 2025.
14. F. H. D. Costa, I. Medeiros, T. Menezes *et al*., “Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for Android malware identification,” *Journal of Systems & Software*, vol. 183, p. 111092, 2022.
15. M. Kedziora, P. Gawin, M. Szczepanik and I. Jozwiak, “Android malware detection using machine learning and reverse engineering,” *Computer Science & Information Technologies*, vol. 10, no. 2, 2020.
16. M. Lv, H. Zeng, T. Chen and T. Zhu, “CTIMD: Cyber Threat Intelligence Enhanced Malware Detection Using API Call Sequences with Parameters,” *Computers & Security*, vol. 123, p. 102953, Oct. 2023.
17. M. Molina-Coronado, U. Mori, A. Mendiburu and J. M.-Alonso, “Towards a fair comparison and realistic evaluation framework of Android malware detectors based on static analysis and machine learning,” *Computers & Security*, vol. 112, p. 102550, Jan. 2023.