

ANDROID MALWARE DETECTION FRAMEWORK USING ATTENTION-BASED DEEP LEARNING

**A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of**

MASTER OF TECHNOLOGY
in
Computer Science & Engineering
by

Mayank Ashok
(Roll No. 23/CSE/31)

**Under the Supervision of
Dr. Rahul Katarya
(Professor, Dept. of Computer Science & Engineering)**



**Department of Computer Science and Engineering
DELHI TECHNOLOGICAL UNIVERSITY**
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi-110042. India

May, 2025

ACKNOWLEDGEMENTS

This research would not have been possible without the guidance, support, and encouragement of many individuals and institutions.

I would like to express my deepest gratitude to **Prof. Rahul Katarya** for his invaluable mentorship, insightful feedback, and unwavering support throughout this project. His expertise and encouragement were instrumental at every stage, from conceptualization to implementation and analysis. I am also sincerely thankful to the **Head of the Department of Computer Science and Engineering at Delhi Technological University** for providing a stimulating academic environment and for their continuous encouragement. My appreciation extends to all faculty and staff members of the department, whose assistance and cooperation greatly facilitated the completion of this work. I acknowledge my colleagues and peers for their collaborative spirit, constructive discussions, and technical assistance, which enriched the research process and contributed to the successful realization of these implementation papers.

Finally, I am grateful to all the people involved for their patience, understanding, and unwavering moral support throughout this journey. Their encouragement provided the foundation that enabled me to persevere and complete this research.

Mayank Ashok

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, **Mayank Ashok**, Roll No. 23/CSE/31, student of M. Tech (Computer Science & Engineering), hereby certify that the work which is being presented in the thesis entitled “**Android Malware Detection Framework Using Attention-Based Deep Learning**” in partial fulfilment of the requirements for the award of the Degree of Master of Technology in Artificial Intelligence in the Department of Computer Science and Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2023 to June 2025 under the supervision of Prof. Rahul Katarya, Professor, Department of Computer Science and Engineering. The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor

Signature of External Examiner

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE

Certified that **Mayank Ashok** (Roll No. 23/CSE/31) has carried out the research work presented in the thesis titled “**Android Malware Detection Framework Using Attention-Based Deep Learning**”, for the award of Degree of Master of Technology from Department of Computer Science and Engineering, Delhi Technological University, Delhi under my supervision. The thesis embodies result of original work and studies are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree for the candidate or submit else from the any other University /Institution.

Date:

Prof. Rahul Katarya
(Supervisor)
Department of CSE
Delhi Technological University

ABSTRACT

The persistent rise of Android malware, coupled with the platform's dominance in the global mobile ecosystem, presents a critical challenge for cybersecurity researchers and practitioners. Traditional malware detection approaches, primarily reliant on static or dynamic analysis, have struggled to keep pace with the rapidly evolving tactics of malicious actors, including code obfuscation and runtime evasion. This thesis addresses these challenges by developing a comprehensive and interpretable hybrid detection framework that leverages both static and dynamic features extracted from Android applications. Using the large-scale KronoDroid dataset—which integrates time-based features from real and emulated environments—an end-to-end methodology was established, encompassing rigorous data preprocessing, advanced feature engineering, and careful handling of class imbalance.

A suite of classical machine learning models, including ensemble methods such as Extra Trees and Random Forest, was systematically evaluated to establish robust performance baselines. Building upon these results, advanced deep learning architectures—including convolutional neural networks (CNN), long short-term memory networks (LSTM), and a hybrid CNN-LSTM model with integrated attention mechanisms—were deployed to capture complex spatial and temporal patterns inherent in hybrid app data. To further enhance detection accuracy and robustness, a confidence-based ensemble strategy was developed, fusing the probabilistic outputs of the best-performing machine learning and deep learning models.

Empirical results demonstrate that the proposed framework achieves state-of-the-art detection rates, with the attention-based CNN-LSTM model delivering significant gains in accuracy, interpretability, and resilience against both false positives and false negatives. The final ensemble fusion approach outperformed all standalone models, achieving an accuracy of 99.61% and minimizing error rates on the KronoDroid benchmark. Detailed analysis of feature importance and attention weights further confirms the practical relevance and transparency of the detection process. This research establishes a scalable, interpretable, and empirically validated blueprint for next-generation Android malware detection, offering actionable insights and a robust methodological foundation for future advances in the field.

LIST OF PUBLICATIONS

1. Mayank Ashok and Rahul Katarya, "Supervised Learning Approaches in Android Malware Detection: Survey and Analysis," accepted for presentation at the 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2025).
2. Mayank Ashok and Rahul Katarya, "Behavioural Analysis for Android Malware Detection: A Deep Learning Approach," accepted for presentation at the 6th International Conference on Data Analytics & Management (ICDAM-2025).
3. Mayank Ashok and Rahul Katarya, "Android Malware Detection Framework Using Attention-Based Deep Learning," submitted to the IEEE 3rd International Conference on Self Sustainable Artificial Intelligence Systems (WCONF-2025).

TABLE OF CONTENTS

Candidate's Declaration	ii
Certificate	iii
Acknowledgement	iv
Abstract	v
List of Publications	vi
List of Tables	ix
List of Figures	x
List of Symbols, Abbreviations, and Nomenclature	xi
 CHAPTER 1 - INTRODUCTION	 1
1.1 Background and Motivation	1
1.2 Research Challenges	3
1.3 Research Objectives	5
1.4 Key Contributions	6
1.5 Lessons Learned	7
 CHAPTER 2- LITERATURE REVIEW	 8
2.1 Static Analysis Techniques	8
2.2 Dynamic Analysis Techniques	9
2.3 Hybrid Analysis Techniques	10
2.4 Machine Learning Approaches	11
2.5 Deep Learning Approaches	12
2.6 Challenges in Existing Literature	13
2.7 Lessons Learned	14
 CHAPTER 3 - DATASETS AND METHODOLOGY	 15
3.1 Introduction	15
3.2 KronoDroid Dataset	15
3.3 Data Preprocessing and Feature Engineering	16
3.4 Model Development and Design	19
3.5 Model Training and Evaluation Protocol	21
3.6 Hyperparameter Optimization Strategy	23
3.7 Model Fusion and Integration Strategy	24
3.8 Lessons Learned	26
 CHAPTER 4 - PROPOSED WORK	 27
4.1 Introduction	27
4.2 Hybrid ML-Based Malware Detection Requirement	27
4.3 DL & Attention-Based Malware Detection Requirement	29

4.4 Fusion/Ensemble Strategy Requirement	31
4.5 Lessons Learned	32
CHAPTER 5 - EXPERIMENTAL SETUP AND RESULTS	34
5.1 Introduction and Experimental Environment	34
5.2 ML Baseline Results	34
5.3 DL & Attention-Based Model Results	36
5.4 Comparative Analysis	37
5.5 Interpretability and Feature Importance Analysis	39
5.6 Lessons Learned	42
CHAPTER 6 – CONCLUSION, FUTURE WORK & SOCIAL IMPACT	45
6.1 Conclusion	45
6.2 Future Scope	46
6.3 Social Impact	47
References	49-52
List of Publications and Their Proofs	
Plagiarism Verification	

LIST OF TABLES

Table Number	Table Name	Page Number
Table 3.1	KronoDroid Dataset Composition	16
Table 3.2	Static and Dynamic Feature Description	18
Table 3.3	Hyperparameter Optimization Configuration	24
Table 5.1	Performance Metrics of ML Models on Hybrid Features	35
Table 5.2	Performance Metrics of DL and Attention-Based Models on Hybrid Features	36
Table 5.3	Comparative Analysis: ML, DL, and Ensemble/Fusion Model Performance	37
Table 5.4	Feature Importance Analysis for Extra Trees Classifier	39

LIST OF FIGURES

Figure Number	Figure Name	Page Number
Fig. 1.1	Android Malware Growth Statistics and Market Share	3
Fig. 3.1	Hybrid feature extraction pipeline: end-to-end process for extracting, integrating, and selecting static and dynamic features from Android applications	19
Fig. 3.2	CNN-LSTM hybrid architecture for integrated static and dynamic feature analysis	22
Fig. 3.3	Experimental workflow and model integration pipeline: illustration of parallel ML/DL inference and confidence-based ensemble fusion for final decision making	26
Fig. 5.1	ROC Curves for CNN-LSTM, Extra Trees, and Confidence-Based Fusion Models	38
Fig. 5.2	Confusion Matrices for CNN-LSTM, Extra Trees, and Confidence-Based Fusion Models	39
Fig. 5.3	Feature Importance Visualization for Extra Trees Classifier	40
Fig. 5.4	Attention Weight Distribution Over Time Steps	41

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

API	Application Programming Interface
AUC	Area Under the ROC Curve
CNN	Convolutional Neural Network
DL	Deep Learning
FNR	False Negative Rate
FPR	False Positive Rate
IPC	Inter-Process Communication
LSTM	Long Short-Term Memory
ML	Machine Learning
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
SMOTE	Synthetic Minority Over-sampling Technique
SHAP	SHapley Additive exPlanations
XAI	Explainable Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

In the contemporary digital era, the rapid proliferation of smartphones has fundamentally altered how individuals and organizations interact with technology. Android, in particular, has emerged as the world's leading mobile operating system, with estimates suggesting it powers over 72% of devices globally by 2025 [1]. This level of dominance stems largely from the platform's open-source design, adaptability across a spectrum of manufacturers, and a thriving environment for third-party applications. While these features have spurred innovation and made Android integral to daily routines, they have also, somewhat paradoxically, created ample opportunities for cybercriminals seeking to exploit both users and enterprises [2]. The ease of distributing applications via the Google Play Store and numerous unofficial sources, though advantageous for developers, has inadvertently set the stage for malicious actors to disseminate harmful code at scale.

Over the past decade, the surge in Android malware incidents has become a persistent concern within the cybersecurity community. Industry reports document a landscape in which tens of millions of malicious software instances are identified annually, reflecting a diversity of attack types that range from financial fraud and ransomware to the theft of sensitive personal data and unauthorized device manipulation [2], [3]. These attacks are not limited to isolated individuals; rather, they increasingly target businesses, government agencies, and critical infrastructure. The broad adoption of Bring Your Own Device (BYOD) initiatives in professional settings, coupled with the integration of Android devices into complex Internet of Things (IoT) environments, has only broadened the possible vectors for intrusion, magnifying the risk and potential consequences of successful malware operations.

Traditionally, the primary defence against such threats has relied upon signature-based detection. These methods function by scanning application code or behavioural patterns for matches against established libraries of known malware signatures. Although effective in identifying familiar threats, signature-based systems are, by nature, limited to reactive protection; their ability to detect previously unseen or morphing malware strains is constrained, often resulting in delayed incident responses and elevated false negative rates [4], [5]. In an effort to address these shortcomings, the field introduced heuristic and behaviour-based detection techniques, which aim to discern malicious activity by modelling the typical behaviour of benign applications and flagging deviations from this baseline. Yet, these alternative methods are not without obstacles—extensive manual rule development, computational inefficiency, challenges in adapting to new attack tactics, and opaque reasoning processes often hinder practical deployment and user trust [6], [7].

Against this backdrop of evolving threats and defensive limitations, the research community has increasingly turned to machine learning (ML) and deep learning (DL)

approaches as promising solutions for malware identification and categorization. These techniques offer a fundamental shift, as they enable systems to learn and generalize from vast, heterogeneous datasets, capturing complex patterns that manual rules may overlook. Notably, hybrid frameworks that integrate static analysis (such as permissions, manifest file attributes, and API usage) with dynamic analysis (like system call monitoring and live network activity) have shown pronounced improvements in detection rates and resilience against tactics like code obfuscation [8], [9]. The logic behind this hybridization is clear: by correlating multiple perspectives on an application's structure and behaviour, it becomes possible to more accurately distinguish between legitimate and malicious software, even as adversaries devise increasingly sophisticated evasion strategies.

Within this growing field of hybrid analysis, deep neural architectures have garnered considerable attention for their ability to extract and model nuanced relationships within complex data. One such architecture, the hybrid Convolutional Neural Network–Long Short-Term Memory (CNN-LSTM) model, has proven adept at simultaneously handling spatial dependencies—captured by CNN layers—and temporal or sequential characteristics—modelled by LSTM layers—embedded within hybrid malware datasets [27], [40]. This dual modelling capability is particularly significant for Android malware, which frequently exhibits intricate code organization and behaviours that unfold across time.

While advanced models such as CNN-LSTM offer improved detection power, their increased complexity presents new challenges. The demands of training and inference, especially in terms of computational resources, can be substantial, making real-world deployment more difficult, particularly on resource-limited mobile devices. Additionally, the risk of overfitting remains, especially when working with high-dimensional data, and the interpretability of such models often lags behind their predictive performance [45], [41]. Security analysts and practitioners, therefore, face a trade-off: highly accurate models may be less transparent and harder to trust in operational contexts.

To bridge the interpretability gap, recent work has introduced attention mechanisms within the CNN-LSTM framework. The attention layer empowers the model to dynamically prioritize certain features or time steps, effectively highlighting which elements of an application's behaviour most influenced its classification outcome. This development not only enhances detection accuracy and reduces false positives but also yields valuable explanations for analysts, allowing for a clearer understanding of why specific applications are flagged as benign or malicious [35], [41]. The ability to visualize attention weights directly addresses concerns about model transparency, supporting both trust and practical incident response.

Parallel to these deep learning advancements, ensemble and confidence-based fusion methods have been recognized for their potential to bolster detection robustness. By aggregating predictions from diverse classifiers—such as decision trees, ensemble forests, and deep neural networks—these strategies capitalize on the complementary strengths of different models, offsetting individual weaknesses and raising the bar for

state-of-the-art performance [16], [15], [45]. Particularly in critical security settings, such integrated approaches are invaluable for minimizing both false positives and false negatives, which are pivotal metrics for operational reliability.

Building upon these collective advancements, the present thesis introduces a scalable and interpretable hybrid detection framework that leverages the KronoDroid dataset, known for its extensive, time-structured collection of Android applications characterized by both static and dynamic features [56]. By combining rigorous feature engineering, attention-based deep learning, and model fusion strategies, the research seeks to address pressing challenges of accuracy, interpretability, and efficiency within the domain of Android malware detection. Ultimately, this work aspires to make a substantive contribution to the ongoing advancement of mobile security.

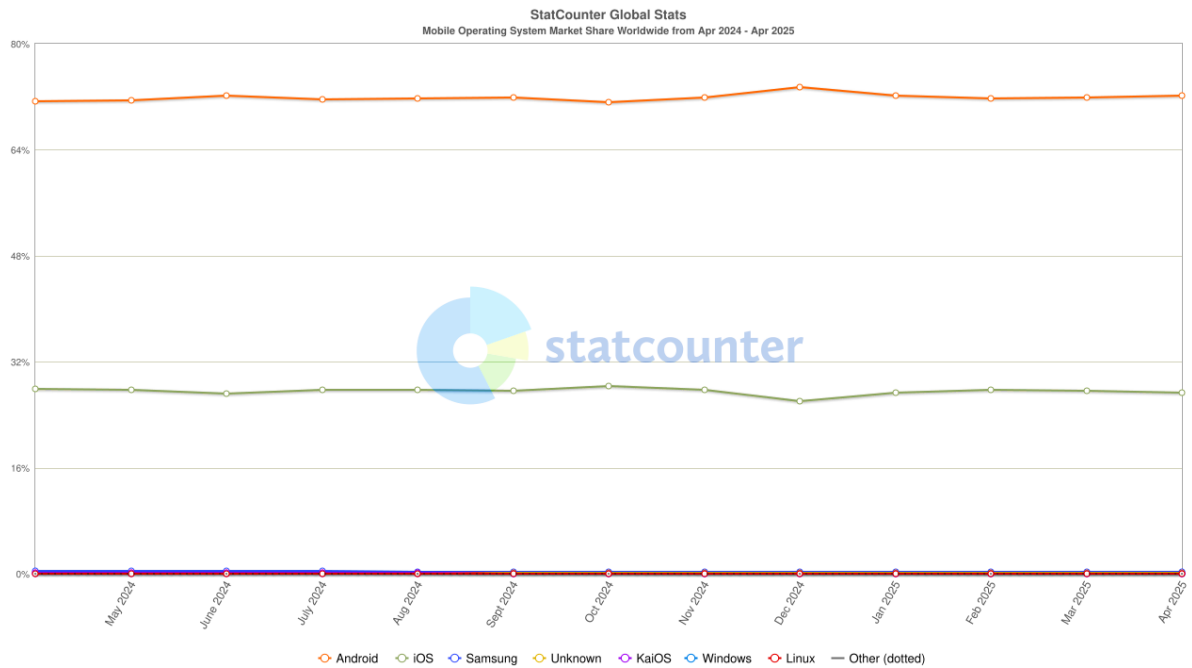


Fig. 1.1 Android Malware Growth Statistics and Market Share [1][2]

As shown in Fig. 1.1, Android's market share dominance and the corresponding exponential increase in malware incidents over recent years highlight the urgency for more advanced, adaptive malware detection strategies.

1.2 Research Challenges

The detection of Android malware presents numerous challenges that stem from the inherent complexity of the Android ecosystem and the evolving sophistication of malware attacks. The open-source nature of Android, combined with a vast array of device models, operating system versions, and third-party app stores, results in a highly fragmented landscape. This fragmentation complicates the design of universal

malware detection systems that are effective across different platforms and environments [1][56].

Malware authors constantly develop new techniques to evade detection. Polymorphism, code obfuscation, and dynamic payload activation are widely used to circumvent traditional signature-based detection methods, which rely on known malware patterns and require continuous updating to remain effective [3][4]. Such methods are inadequate against zero-day malware that exploit unknown vulnerabilities, creating an urgent need for detection systems that can generalize beyond known signatures [2].

Anomaly-based and heuristic approaches in malware detection focus on recognizing activities that diverge from established benign behaviour, or utilize heuristic rules to flag suspicious patterns. However, these strategies frequently encounter the drawback of high false positive rates. This occurs when legitimate applications display uncommon or unexpected behaviours, leading to their misclassification as threats and consequently diminishing both user confidence and the practical utility of such detection systems [5], [6]. Static analysis, which entails examining application code in the absence of execution, remains susceptible to advanced obfuscation and packing strategies. Malicious actors exploit these techniques to conceal harmful code, thus evading conventional static inspections [6], [56].

To obtain a deeper understanding of app behaviour, dynamic analysis executes applications in controlled environments and observes their actions in real time. While this dynamic scrutiny can reveal sophisticated attack vectors, it brings its own set of obstacles—most notably, substantial computational requirements and extended execution times, both of which complicate the scaling of such methods to thousands of applications in operational settings. Compounding these issues, malware developers increasingly deploy evasion tactics, such as detecting virtualized environments or deferring payload activation, further hindering the effectiveness of dynamic analysis frameworks [7], [56].

Recent advances in machine learning (ML) and deep learning (DL) offer compelling alternatives by automating feature extraction and classification. Nevertheless, the adoption of these methods is hampered by challenges tied to data quality, thoughtful feature selection, and reliable model generalization. Datasets for Android malware detection are often characterized by pronounced class imbalance, where benign samples far outnumber malicious ones. This imbalance introduces the risk of bias in learning algorithms, potentially impairing their ability to identify true threats [19], [37]. Furthermore, sophisticated DL models—including convolutional neural networks (CNNs) and long short-term memory (LSTM) architectures—are data-hungry, requiring extensive labelled datasets to avoid overfitting and to achieve dependable results. Their significant computational footprint also raises concerns about the feasibility of deploying such models on typical mobile devices with limited hardware resources [40], [41].

A further complication is the challenge of interpretability. The internal logic of many DL systems is opaque—a “black box” that leaves analysts unable to trace how specific decisions are made. This lack of transparency undermines trust and impedes widespread operational use [40], [41]. As a result, developing interpretability methods that provide meaningful explanations for model outputs is essential, both for validating alerts and for understanding the evolving tactics of Android malware.

Integrating static and dynamic features into a cohesive detection framework remains a complex undertaking. Effective feature fusion must maintain the sequence and structure inherent in dynamic behaviours, while also drawing on the contextual richness of static application characteristics. Ineffective integration may result in the loss of critical information or introduce redundancy, which can ultimately diminish overall detection accuracy [56].

Additionally, adversarial attacks have emerged as a significant threat in this domain. Sophisticated attackers may manipulate input features or exploit weaknesses in detection models to bypass security systems. Enhancing model robustness requires continuous adaptation and the implementation of defensive mechanisms designed to resist adversarial manipulations [34].

In sum, the primary obstacles for Android malware detection include the need to maximize detection accuracy while minimizing false positives, improving the interpretability of complex algorithms, optimizing computational efficiency for mobile environments, ensuring effective integration of diverse features, and maintaining resilience in the face of evolving malware strategies and adversarial threats.

1.3 Research Objectives

The central aim of this thesis is to propose and implement a scalable, interpretable, and resilient detection framework for Android malware, directly addressing prevailing limitations regarding accuracy, explainability, and operational efficiency. The following research objectives define the core focus of the work:

Objective 1: Development of a Hybrid Feature Extraction and Selection Framework- This objective is dedicated to building a comprehensive approach that synthesizes both static and dynamic analysis techniques to capture the most relevant characteristics of malicious applications. Features such as permissions, API calls, manifest properties, system call traces, and network activity patterns are systematically extracted and incorporated. To enhance the effectiveness of the detection process, advanced feature selection algorithms are applied to reduce dimensionality, eliminate redundancy, and support improved classification outcomes [56], [18], [19].

Objective 2: Integration of Attention-Based Deep Learning for Improved Detection and Explainability- To implement an attention-enhanced CNN-LSTM deep learning model that dynamically focuses on the most relevant features during classification. This model aims to increase detection accuracy, reduce false

positives, and provide interpretable insights into the decision-making process, addressing the opacity of traditional deep learning approaches [40][41][35].

Objective 3: Development of an Ensemble Machine Learning Strategy for Enhanced Robustness and Scalability- To utilize ensemble learning methods combining multiple classical machine learning classifiers, such as Extra Trees, AdaBoost, and Random Forest, to improve detection robustness and adaptability. The ensemble strategy will be optimized for deployment on resource-constrained devices, balancing accuracy and computational efficiency [20][37][45].

1.4 Key Contributions

This thesis makes the following key contributions to the field of Android malware detection:

1. Comprehensive Hybrid Feature Extraction Framework- The thesis presents a novel framework that effectively integrates both static and dynamic features extracted from Android applications. By leveraging permissions, API calls, manifest data alongside runtime behaviours such as system calls and network traffic, the framework improves the detection of sophisticated malware variants. The use of advanced feature selection techniques ensures reduction in dimensionality without sacrificing classification performance [56][18][19].

2. Attention-Enhanced CNN-LSTM Deep Learning Model- An attention mechanism is incorporated within a hybrid CNN-LSTM architecture to dynamically focus on the most relevant features for classification. This attention-based model achieves superior detection accuracy and significantly reduces false positives while providing interpretable insights into the classification process. This addresses the common limitation of deep learning models being perceived as black boxes in security applications [40][35][41].

3. Ensemble Machine Learning Strategy for Robust and Scalable Detection- The research develops an ensemble approach combining multiple classical machine learning classifiers to enhance robustness and generalization across diverse malware samples. The ensemble is designed with efficiency in mind, enabling practical deployment in resource-constrained mobile environments while maintaining high detection performance [20][37][45].

4. Empirical Validation on the KronoDroid Dataset- The proposed methods are thoroughly evaluated on the KronoDroid dataset, a time-based hybrid-featured dataset capturing a wide range of Android malware behaviours. This evaluation demonstrates the effectiveness, robustness, and scalability of the framework in realistic and diverse scenarios [56].

5. Interpretability and Explainability Enhancements- The integration of attention mechanisms and post-hoc interpretability techniques provides actionable insights into model decisions, facilitating trust and adoption by security analysts.

Visualizations of attention weights and feature importance analyses enhance the transparency of the malware detection process [35][40][41].

1.5 Lessons Learned

Chapter 1 highlights the significant challenges and complexities involved in Android malware detection. The increasing diversity and fragmentation of the Android ecosystem demand detection frameworks that are both adaptable and comprehensive. Traditional detection methods, including signature-based and heuristic approaches, exhibit limitations against rapidly evolving malware employing sophisticated evasion techniques.

The integration of static and dynamic analysis features emerges as a critical factor for robust malware characterization. Combining static and dynamic data sources within a malware detection system introduces considerable complexity, particularly in the processes of feature selection and fusion—both of which are directly linked to the overall detection capability of the framework [56]. If features are not carefully integrated, critical information may be lost or redundant, resulting in diminished accuracy and effectiveness.

Although recent advances in machine learning and deep learning offer promising tools for automated detection, these methods present persistent challenges regarding interpretability, computational requirements, and reliable generalization to new types of threats [40], [41]. Reducing false positives, while maintaining strong detection rates, remains especially crucial for real-world mobile environments where resources are inherently limited.

The adoption of explainable artificial intelligence techniques, such as attention mechanisms and ensemble learning, can substantially enhance both the usability and trustworthiness of detection outcomes by providing clearer insights into model decisions [41], [35], [45].

In summary, these considerations underscore the pressing need for a hybrid, interpretable, and scalable malware detection framework—one that can keep pace with evolving Android threats, and that successfully balances the demands of accuracy, computational efficiency, and operational deployability [56], [40], [45].

CHAPTER 2

LITERATURE REVIEW

The widespread adoption of Android devices, accompanied by a surge in sophisticated malware, has prompted the development and continuous refinement of a diverse set of detection methodologies within the research community [3], [4], [8]. Conducting a thorough review of prior literature is essential not only for grounding this study in established theory, but also for pinpointing unresolved challenges and emerging research opportunities [4], [8]. This chapter systematically examines major strategies for Android malware detection, beginning with traditional static and dynamic analysis paradigms and subsequently exploring hybrid frameworks that seek to combine the advantages of both [4], [17], [56]. In addition, it surveys the rapidly expanding influence of machine learning and deep learning models, which have revolutionized detection by automating the discovery of complex patterns and significantly enhancing classification performance [8], [27], [40]. By critically evaluating recent contributions, this review sheds light on the advancements achieved, the persistent obstacles that remain, and the open questions steering the future of Android malware detection research [4], [8], [56]. The following sections provide focused discussions on static analysis (Section 2.1), dynamic analysis (Section 2.2), hybrid methodologies (Section 2.3), the adoption of machine learning (Section 2.4), progress in deep learning (Section 2.5), and prevailing challenges found in the literature (Section 2.6), before concluding with a summary of lessons learned to inform ongoing and future investigations.

2.1 Static Analysis Techniques

Static analysis constitutes a foundational strategy for identifying Android malware, relying on a thorough inspection of an application's package contents without requiring execution. Key features extracted in this process include the permissions an application requests, the sequence and frequency of API calls, the structure of intent filters, manifest file attributes, and patterns within the bytecode itself. One of the primary strengths of static analysis is its efficiency; by forgoing the need to execute the app, large volumes of software can be analysed rapidly and at scale, making this approach particularly suitable for broad market surveillance and app store vetting [3], [4], [56].

Signature-based detection remains a prominent static analysis technique, wherein malware is identified through the comparison of code fragments or distinctive patterns against a repository of established malicious signatures. Although this method delivers high accuracy when confronting previously catalogued threats, it is fundamentally limited by its inability to detect novel, polymorphic, or zero-day malware that actively modify their code to evade recognition [3], [4]. These inherent weaknesses have encouraged researchers to advance toward feature-driven static analysis, leveraging granular attributes—such as permission requests and API invocation patterns—to build a behavioural profile of applications [5], [6].

Permissions requested by Android apps often signal their intended behaviours and, in some cases, can suggest malicious objectives, particularly when sensitive operations like SMS access, contact retrieval, or location tracking are involved. Similarly, the analysis of API call sequences offers a deeper understanding of an app's operational flow, and has been widely applied to distinguish benign applications from those exhibiting harmful intent [5], [56].

Manifest file analysis can expose components such as broadcast receivers or services that may facilitate malicious activity. Combined, these static features have demonstrated utility in improving detection rates when paired with appropriate machine learning classifiers.

However, static analysis faces notable challenges. Modern malware often employs sophisticated code obfuscation techniques—such as renaming classes and methods, string encryption, and control flow flattening—to conceal malicious logic, severely undermining the effectiveness of static inspection [5][6]. Additionally, dynamic code loading and reflection, prevalent in many Android apps, limit static analysis coverage because the actual code executed may not be fully visible during static scanning [56]. Furthermore, the reliance on extracted static features alone can lead to false positives since some benign apps legitimately request sensitive permissions or utilize advanced features.

To address these issues, hybrid approaches combining static and dynamic analysis have gained traction, but static analysis remains an essential, low-overhead tool in the malware analyst's arsenal. The balance between speed and accuracy, alongside ongoing improvements in static feature extraction and deobfuscation techniques, ensures its continued relevance in Android malware detection research.

2.2 Dynamic Analysis Techniques

Dynamic analysis methods focus on evaluating the real-time behaviour of Android applications by executing them within emulated or sandboxed environments. Through this approach, analysts can observe system calls, monitor network interactions, trace file system modifications, and inspect inter-process communication, thereby uncovering behavioural indicators of malware that are often missed by static analysis alone [7], [56]. By enabling the direct observation of runtime activity, dynamic analysis is uniquely equipped to reveal behaviours that manifest only during execution, such as the loading of external code or the activation of encrypted payloads.

This runtime-centric perspective makes dynamic analysis especially valuable in detecting malware that leverages advanced evasion tactics, including code obfuscation, the use of reflection, or active measures to identify and bypass sandbox environments [14]. Notable tools like TaintDroid and CopperDroid have demonstrated the effectiveness of this methodology, employing detailed monitoring of sensitive data flows and reconstructing execution paths to surface suspicious or unauthorized operations [13], [14].

Despite these strengths, dynamic analysis is not without its own set of challenges. The fidelity of the execution environment is critical; if the analysis platform fails to convincingly mimic a genuine device, sophisticated malware may suppress its malicious actions or delay execution to evade detection, resulting in false negatives [7], [14]. In addition, the process demands substantial computational resources and can be time-intensive, posing practical difficulties for scaling to large datasets or deploying real-time detection solutions on typical mobile hardware [56].

Another layer of complexity arises from the sheer volume and intricacy of data collected during dynamic monitoring. Extracting meaningful features from this high-dimensional, sequential data requires advanced data processing architectures and robust machine learning models designed to capture temporal dependencies and evolving behavioural patterns [56], [40].

Despite these challenges, dynamic analysis remains a crucial component of comprehensive malware detection frameworks, especially when combined with static analysis in hybrid models.

2.3 Hybrid Analysis Techniques

Hybrid analysis techniques combine static and dynamic analysis to leverage the strengths and mitigate the weaknesses of each individual approach. By integrating static features such as permissions, API calls, and manifest attributes with dynamic runtime behaviours like system calls and network traffic, hybrid methods provide a more comprehensive understanding of Android application behaviour [17][18][56].

The synergy between static and dynamic features enhances detection accuracy and robustness against obfuscation and evasion tactics. Hybrid models can detect malicious behaviour that might be missed by static or dynamic analysis alone, improving resilience to zero-day and polymorphic malware [17]. Several studies have demonstrated that the fusion of heterogeneous feature sets results in better classification performance compared to using either feature set independently [18][56].

However, hybrid analysis introduces challenges related to feature fusion, dimensionality, and computational overhead. Integrating diverse data types requires sophisticated feature selection and fusion strategies to ensure that complementary information is retained without redundancy [56]. Additionally, the combined analysis process is more resource-intensive and complex, raising concerns about scalability and real-time applicability, especially on mobile devices with limited resources [56].

Advanced machine learning and deep learning techniques, such as CNN-LSTM hybrids, have been effectively employed to process hybrid features by capturing spatial and temporal patterns inherent in static and dynamic data, respectively [40]. The application of attention mechanisms further improves the model's ability to focus on the most discriminative features within the fused data, enhancing both accuracy and interpretability [35][41].

Overall, hybrid analysis represents a promising direction in Android malware detection, balancing the trade-offs of static and dynamic methods while pushing the boundaries of detection performance.

2.4 Machine Learning Approaches

Machine learning (ML) techniques have been extensively applied to Android malware detection due to their ability to automatically learn patterns from data without explicit programming. Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), and Naive Bayes classifiers have consistently demonstrated promising effectiveness in differentiating benign from malicious Android applications, whether based on static, dynamic, or hybrid sets of features [20], [21], [22], [23], [24]. Decision trees and ensemble techniques such as Random Forest and AdaBoost further enhance classification robustness by integrating the outputs of multiple weak learners, which serves to improve detection accuracy and control overfitting [20], [21], [45]. Random Forest classifiers, in particular, have proven adept at managing high-dimensional feature spaces—such as those constructed from permission and API call attributes—resulting in both high detection accuracy and relatively low false positive rates [21].

Ensemble learning strategies strengthen detection systems by aggregating the predictions of diverse classifiers, thereby making them more resilient to noisy or imbalanced datasets [37], [45]. Support Vector Machines are frequently utilized for their capacity to handle non-linear separations in feature space through the use of kernel functions, which is particularly advantageous in the detection of sophisticated or obfuscated malware [22], [44]. Nonetheless, effective use of SVMs often depends on meticulous feature engineering and careful parameter optimization.

The k-Nearest Neighbours algorithm represents a straightforward yet powerful technique, assigning labels to samples according to their proximity to previously labelled instances within the feature space [23]. While k-NN is intuitive and easy to implement, it can incur significant computational costs at prediction time, which may hinder its scalability for large datasets. Naive Bayes classifiers, founded on principles of probabilistic modelling, are valued for their speed and interpretability; however, their performance can degrade if the assumption of feature independence does not hold in complex malware datasets [24].

Despite the practical successes of these classical machine learning methods, several limitations persist. Chief among them is the reliance on manually crafted features, sensitivity to the quality of feature selection, and a restricted capacity to capture the sequential or temporal aspects often embedded in dynamic malware behaviour [8], [19]. The challenge is further compounded by the inherent class imbalance found in most Android malware datasets, where benign samples vastly outnumber malicious ones. This imbalance can negatively affect the learning process, often necessitating the adoption of data balancing techniques such as the Synthetic Minority Over-sampling Technique (SMOTE) or generative approaches to ensure robust classifier performance [37].

In conclusion, machine learning approaches constitute a foundational element of Android malware detection frameworks, particularly when coupled with well-engineered features and ensemble strategies. However, their inherent limitations have driven the field towards deep learning solutions, which aim to overcome these challenges.

2.5 Deep Learning Approaches

The adoption of deep learning (DL) has ushered in a transformative era for Android malware detection, primarily by enabling the automatic discovery of intricate patterns within high-dimensional data and addressing many constraints of classical machine learning. Unlike traditional ML techniques, which typically depend on explicit feature engineering, DL models are capable of autonomously extracting hierarchical representations and capturing complex spatial as well as temporal relationships embedded in both static and dynamic application behaviors [27], [33], [40].

Convolutional Neural Networks (CNNs) have demonstrated strong performance in analysing static features, such as permissions, API call graphs, and opcode sequences, by learning local patterns and spatial relationships in input data [27][35][40]. By transforming application binaries or features into image-like representations, CNNs can distinguish between benign and malicious samples with high accuracy and minimal feature engineering [35].

Recurrent Neural Networks (RNNs), and particularly Long Short-Term Memory (LSTM) networks, have been widely adopted to capture sequential and temporal patterns in dynamic behaviours, such as system call traces, network activity, and execution sequences [33][40][41]. LSTM models can effectively learn from ordered data and are robust to long-term dependencies, making them suitable for modelling the evolution of malware actions over time [41].

Hybrid models, notably CNN-LSTM architectures, combine the strengths of CNNs in extracting spatial features and LSTMs in capturing temporal dependencies, providing superior detection capabilities for hybrid static and dynamic feature sets [40]. Recent research has shown that such models can outperform standalone CNN or LSTM models, especially in large-scale, heterogeneous datasets [27][40].

Attention mechanisms have further advanced DL-based malware detection by allowing models to selectively focus on the most relevant parts of input data during classification [35][41]. By assigning different weights to features or time steps, attention layers enhance both the interpretability and accuracy of deep learning models, addressing the “black-box” criticism associated with traditional DL methods [35][41].

Despite their advantages, deep learning approaches present unique challenges. Training deep neural networks requires large, labelled datasets to avoid overfitting and ensure generalization [40]. DL models are also computationally intensive, making their deployment on resource-constrained mobile devices challenging without model

optimization or compression strategies [41]. Additionally, the security and robustness of DL models are under continuous threat from adversarial attacks, where small perturbations in the input can fool the model into making incorrect predictions [34][38].

Recent advances in explainable AI (XAI) techniques—such as SHAP, LIME, and attention visualization—are increasingly integrated with deep learning frameworks to provide actionable insights for security analysts and improve model transparency [38][52].

In summary, deep learning has emerged as a powerful paradigm for Android malware detection, enabling high accuracy and rich behavioural analysis. Continued research focuses on improving model interpretability, computational efficiency, and robustness against adversarial threats.

2.6 Challenges in Existing Literature

Despite significant advancements in Android malware detection, several persistent challenges continue to limit the effectiveness and practicality of existing solutions. A key challenge is the handling of imbalanced datasets. Real-world malware datasets often contain far more benign samples than malicious ones, leading to biased model training, reduced sensitivity to rare malware variants, and an increased risk of false negatives [19][37]. While data balancing techniques such as oversampling, synthetic data generation, and generative adversarial networks (GANs) have been proposed, achieving optimal balance without introducing artifacts remains an open problem [37].

Feature engineering is another major concern. Many approaches depend on manually crafted features, which require domain expertise and may not generalize well to unseen malware types. Incomplete or suboptimal feature sets can degrade detection accuracy [8][19]. Automated feature learning via deep learning can help, but still faces issues related to data quality, interpretability, and the need for large, labelled datasets [40][41].

The interpretability of machine learning and deep learning models presents a notable limitation. Security analysts require transparent and explainable models to build trust and enable forensic investigation. Traditional ML models, while more interpretable, may lack predictive power, whereas deep neural networks, though accurate, are often viewed as “black boxes” [38][40][52]. Recent advances in explainable AI (XAI) and the integration of attention mechanisms are beginning to address these issues but are not yet universally adopted or mature [35][52].

Computational efficiency and resource constraints pose practical challenges, especially for real-time detection on mobile devices. Many deep learning models demand significant computational resources for both training and inference, which can hinder their deployment in production environments [40][41]. Approaches such as model pruning, quantization, and edge computing are under exploration to mitigate these constraints.

A further challenge is the evolving threat landscape. Malware authors continuously develop new evasion techniques, such as adversarial attacks, code obfuscation, and environment-aware payload activation, which are specifically designed to circumvent state-of-the-art detection methods [34][56]. This cat-and-mouse dynamic necessitates continual adaptation and ongoing research to maintain detection efficacy.

Lastly, there is a lack of standardized benchmarks and publicly available, diverse datasets that cover a wide range of real-world scenarios. Many studies rely on limited or proprietary datasets, complicating fair comparison and reproducibility of results [56].

In summary, future research in Android malware detection must address data imbalance, advance automated and interpretable feature engineering, improve computational efficiency, develop robust adversarial defence strategies, and foster the creation and sharing of standardized evaluation datasets.

2.7 Lessons Learned

Chapter 2 underscores the substantial progress and diversity of approaches in Android malware detection, encompassing static, dynamic, hybrid, machine learning, and deep learning techniques. The review highlights that static analysis provides efficiency but is vulnerable to obfuscation, while dynamic analysis uncovers runtime behaviours but incurs computational overhead and is susceptible to evasion tactics. Hybrid approaches have emerged as a promising direction, leveraging the complementary strengths of both static and dynamic features to improve detection accuracy and robustness.

Machine learning methods—especially when combined with effective feature selection and ensemble strategies—demonstrate significant potential, yet remain constrained by reliance on handcrafted features and challenges related to imbalanced datasets. Deep learning models, particularly those integrating CNNs, LSTMs, and attention mechanisms, offer automated feature learning and state-of-the-art detection accuracy. However, they introduce new concerns, including high computational demands, lack of transparency, and susceptibility to adversarial attacks.

A recurring theme across the literature is the tension between detection performance, interpretability, and practical deployability. The absence of widely accepted standardized datasets and consistent benchmarking protocols continues to pose a challenge for the field, often hindering fair comparison and reproducibility across different Android malware detection studies [56]. Moving forward, future research should prioritize the development of interpretable, efficient, and adaptive detection frameworks that can address evolving threats while maintaining operational feasibility for real-world deployment.

CHAPTER 3

DATASETS AND METHODOLOGY

3.1 Introduction

This chapter provides a comprehensive overview of the datasets employed in this study and describes, in detail, the methodological framework adopted for Android malware detection. Establishing a clear and rigorous experimental protocol is essential to guarantee that the findings are valid, reproducible, and applicable to broader real-world scenarios [56]. The methodological workflow encompasses data collection, preprocessing, feature engineering, model design, training and evaluation, and interpretability analysis.

A key strength of this research is the use of the *KronoDroid dataset*, a large-scale, time-based hybrid-featured dataset comprising both static and dynamic features extracted from real-world and emulated Android applications [56]. The hybrid nature of the dataset enables comprehensive evaluation of both traditional and advanced detection models. The chapter proceeds by first describing the characteristics and preparation of the dataset, followed by a detailed explanation of the feature engineering process, model architecture, training protocols, and evaluation metrics.

3.2 KronoDroid Dataset

The KronoDroid dataset serves as the foundational benchmark for the experimental evaluation in this research. Developed to address limitations of existing datasets, KronoDroid is a large-scale, time-based hybrid-featured dataset specifically curated for Android malware detection and behavioural characterization [56]. It includes both static and dynamic features, capturing a comprehensive range of behaviours exhibited by benign and malicious applications.

Dataset Composition: KronoDroid comprises samples collected from real Android devices as well as emulator environments to enhance the generalizability of detection models. The dataset includes both benign applications and malware samples representing a variety of malware families and behavioural patterns. The use of multiple collection environments ensures that models trained on KronoDroid are robust against device-specific artefacts and emulator-detection evasions [56].

- **Static Features:** These are extracted from APKs without execution and include application permissions, API calls, manifest attributes, and metadata. Such features provide valuable context regarding the declared capabilities and structural properties of each app [56][18].
- **Dynamic Features:** Captured during the runtime execution of applications in sandboxed environments, dynamic features encompass system calls, network traffic, file system operations, and process activity. These reveal the real-time actions and behavioural signatures of apps, allowing detection of obfuscated or dynamically-loaded malicious code [56][40].

- **Dataset Statistics:** KronoDroid contains tens of thousands of samples, spanning multiple malware families, and is annotated to indicate the ground truth (malicious or benign). The dataset is temporally structured, enabling longitudinal analysis of malware evolution and concept drift over time [56].

Significance: KronoDroid’s hybrid and time-based structure allows for the assessment of detection models in realistic and evolving threat scenarios. The availability of rich, labelled static and dynamic features provides a robust basis for the training and evaluation of hybrid models that integrate both types of information [56][18][40]. Moreover, the inclusion of emulator and real device traces supports research into evasion resistance and model robustness.

Table 3.1 KronoDroid Dataset Composition [56]

Sample Type	Real Device	Emulated Environment	Total
Malware Samples	41,382	28,745	70,127
Legitimate Applications	36,756	35,246	72,002
Total Samples	78,138	63,991	142,129

3.3 Data Preprocessing and Feature Engineering

Effective data preprocessing and feature engineering are critical for building robust Android malware detection models, as the quality of input features directly influences detection accuracy, interpretability, and computational efficiency [19][56]. This section details the preprocessing pipeline and the strategies employed for extracting, transforming, and selecting features from the KronoDroid dataset.

3.3.1 Data Preprocessing

The raw KronoDroid dataset is subjected to a series of preprocessing steps to ensure data quality and compatibility with downstream machine learning and deep learning models:

- **Data Cleaning:** Duplicate entries, incomplete records, and corrupted samples are removed to eliminate noise and potential biases. All missing values are handled using appropriate imputation techniques or, where necessary, by discarding incomplete instances [56].
- **Label Encoding:** The ground truth labels (malicious or benign) are encoded in a binary format to facilitate supervised learning tasks. Any categorical variables present in the feature set are similarly encoded [19].
- **Normalization and Scaling:** Continuous features, especially those with wide value ranges (such as system call counts or file sizes), are normalized using techniques like min-max scaling or z-score standardization. This step is essential for ensuring that all features contribute proportionally during model training [19][56].
- **Class Imbalance Handling:** Since real-world malware datasets are often imbalanced (with benign samples typically outnumbering malicious ones),

Synthetic Minority Over-sampling Technique (SMOTE) or similar data augmentation methods are applied to balance the classes and improve model sensitivity [19][37].

3.3.2 Feature Engineering for Hybrid Detection Models

Feature engineering in this research is designed to exploit both static and dynamic perspectives of Android applications:

- **Static Feature Extraction:** These features are derived from the APK package without execution, including:
 - **Permissions:** Indicators of requested access, such as READ_SMS, INTERNET, or ACCESS_FINE_LOCATION [56][18].
 - **API Calls:** Frequencies and sequences of sensitive Android API usages, which often reflect underlying behaviour [56].
 - **Manifest Attributes:** Analysis of app components (activities, services, broadcast receivers), intent filters, and metadata from the AndroidManifest.xml file [56].
 - **Structural Metadata:** Application file size, certificate information, and resource counts.
- **Dynamic Feature Extraction:** These features are gathered during sandboxed execution of each app:
 - **System Calls:** Patterns and frequencies of kernel-level system calls made during execution, which capture behavioural traits of both benign and malicious applications [56][40].
 - **Network Activity:** Outbound and inbound connections, including URLs, IP addresses, and traffic statistics.
 - **File System and Process Activity:** Modifications to file systems, creation or termination of processes, and inter-process communication [56].

The combination of static and dynamic features provides a holistic view of application behaviour, significantly enhancing the capability to detect advanced and evasive malware variants [17][18][56].

Table 3.2: Static and Dynamic Feature Description [56]

Feature Type	Feature Category	Description	Example Features
Static	Permissions	Requested app permissions, indicating access to sensitive device resources	READ_SMS, INTERNET, ACCESS_FINE_LOCATION
	API Calls	Frequency and sequence of sensitive Android API usage	sendMessage(), getDeviceId()
	Manifest Attributes	App components, intent filters, and metadata from AndroidManifest.xml	Activities, Services, Receivers
	Structural Metadata	Static file and certificate properties	APK size, Certificate info, Resource count
Dynamic	System Calls	Patterns and frequency of kernel-level system calls during app execution	execve, open, read, write, nr_syscalls
	Network Activity	Outbound/inbound network connections, URLs, and traffic statistics	URL requests, IP connections, Packets sent
	File System & Process Activity	File operations and process management observed at runtime	File creation, Process launch, IPC

3.3.3 Feature Selection

Given the high dimensionality of hybrid features, feature selection techniques are employed to retain only the most informative and discriminative features:

- **Filter Methods:** Statistical measures (e.g., mutual information, chi-square) are used to rank features based on relevance to the target label [19].
- **Wrapper and Embedded Methods:** Recursive feature elimination and tree-based feature importance rankings (e.g., from Random Forest or Extra Trees models) are applied to further refine the selected subset [19][20].
- **Dimensionality Reduction:** Principal Component Analysis (PCA) is explored to capture the principal sources of variance, particularly for reducing redundancy among correlated features [19].

The resulting feature set is both compact and effective, supporting the training of interpretable and high-performance models [19][20][56]. The overall process—starting from data cleaning and preprocessing, through hybrid static and dynamic feature extraction, to final feature selection—is depicted in the hybrid feature extraction pipeline shown in Fig. 3.1. This pipeline illustrates the sequential and

parallel steps taken to generate a comprehensive, unified feature set for input to the detection models.

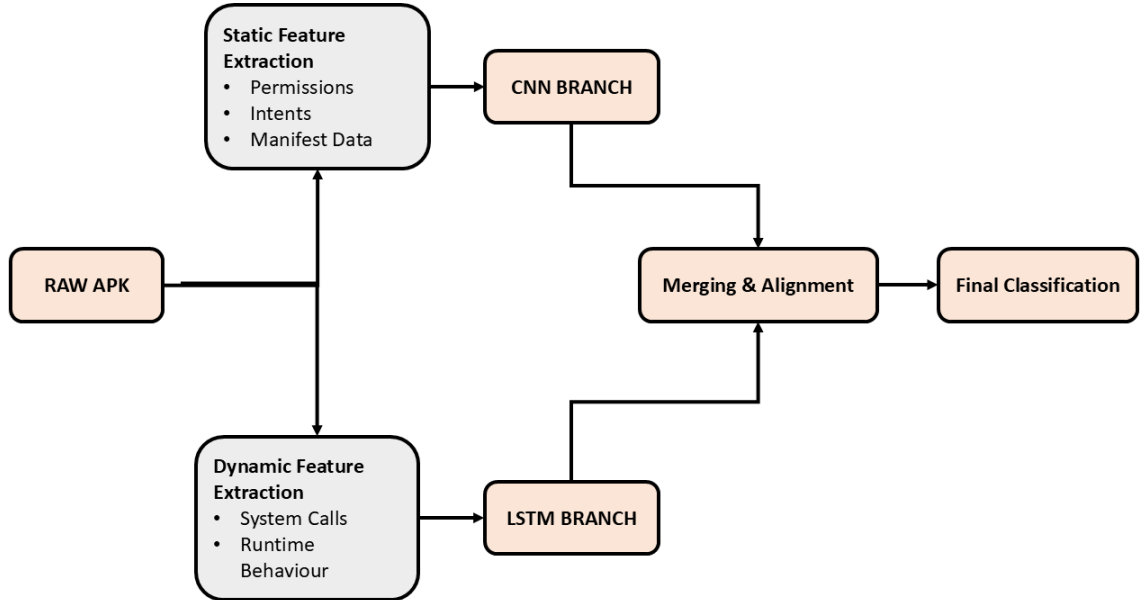


Fig. 3.1 Hybrid feature extraction pipeline: end-to-end process for extracting, integrating, and selecting static and dynamic features from Android applications.

3.4 Model Development and Design

The effectiveness of Android malware detection frameworks relies heavily on the selection and design of appropriate machine learning (ML) and deep learning (DL) models. This section describes the model development pipeline, the architectural choices for both classical ML and advanced DL methods, and the justification for adopting hybrid and attention-based designs in this research.

3.4.1 Machine Learning Model Selection

Classical ML algorithms have been widely used for Android malware detection due to their interpretability and computational efficiency. In this research, a suite of well-established classifiers—including Decision Trees, Random Forest, AdaBoost, Extra Trees, and Support Vector Machines—were evaluated on the preprocessed KronoDroid dataset [20][21][22][45]. Ensemble methods such as Random Forest and Extra Trees were prioritized because of their robustness against overfitting, ability to handle high-dimensional feature spaces, and effectiveness in managing imbalanced data distributions [20][21][45].

Each ML model was trained using the hybrid feature set described in Section 3.3. Hyperparameter optimization (detailed in Section 3.6) was performed using grid search and cross-validation to identify optimal settings for each classifier. The models

were evaluated using standard classification metrics, and the best-performing models were selected for integration into the ensemble strategy [45].

3.4.2 Deep Learning Model Architecture

Deep learning models offer superior capacity for learning complex, non-linear patterns directly from data, especially in scenarios involving high-dimensional and heterogeneous feature sets [27][40]. This research employs three primary DL architectures:

- **Convolutional Neural Networks (CNN):** CNNs are adept at capturing spatial and local patterns from static features such as permissions and API calls when represented in matrix or sequence form. CNNs reduce the need for manual feature engineering and excel at recognizing local patterns that may signal malicious behaviour [27][35][40].
- **Long Short-Term Memory Networks (LSTM):** LSTMs, a type of Recurrent Neural Network (RNN), are highly effective in modelling sequential dependencies, making them suitable for analysing dynamic features such as system call traces and network activity. LSTM networks can retain contextual information over long sequences, which is essential for detecting behaviours that unfold over time [33][41].
- **Hybrid CNN-LSTM Model:** To capitalize on the complementary strengths of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) architectures, this study adopts a hybrid modelling approach. In this configuration, CNN layers are employed to extract spatial patterns and local dependencies from the input features, while the representations generated by the CNN are subsequently passed to LSTM layers that are adept at capturing temporal relationships within sequential data. This combined architecture is especially effective for Android malware detection scenarios that require simultaneous analysis of static and dynamic features [27], [40], [41].
- **Attention Mechanism Integration:** To further augment both the interpretability and discriminative power of the hybrid model, an attention mechanism is incorporated within the CNN-LSTM framework. By introducing an attention layer, the model gains the capacity to selectively emphasize important features or time steps during classification, which not only enhances overall detection accuracy but also provides explainable, transparent insights that are valuable to security analysts investigating alerts or suspicious behaviours [35], [40], [41].

3.4.3 Ensemble Strategy and Model Fusion

Beyond deploying individual classifiers, an ensemble learning strategy is utilized to combine the predictive strengths of multiple machine learning and deep learning models. Ensemble approaches have been widely recognized for their ability to boost overall robustness, generalization, and resistance to overfitting by synthesizing the predictions from a diverse set of base models [20], [37], [45]. In this work, the final ensemble is constructed by calibrating and fusing the prediction probabilities from the

leading Extra Trees (ML) and attention-enhanced CNN-LSTM (DL) models, using a confidence-based fusion method designed to further minimize both false positives and false negatives.

3.4.4 Implementation Considerations

All model development and experimentation are conducted using widely adopted Python libraries, including scikit-learn, TensorFlow, and Keras, thereby ensuring the reproducibility and scalability of the research outcomes. Model selection and evaluation procedures follow best practices in ML/DL experimentation, with a clear separation between training, validation, and testing phases [19][40].

3.5 Model Training and Evaluation Protocol

A clear and consistent training and evaluation protocol is essential to ensure valid, transparent, and comparable results for Android malware detection. In this research, all model development and assessment steps were rigorously standardized and conducted in accordance with the methodology described in earlier sections.

The preprocessed KronoDroid dataset (see Section 3.3) was split into 80% for training and 20% for testing using stratified sampling, ensuring the proportion of benign and malicious samples remained consistent across both sets [56]. This split was maintained throughout all experiments, and no further internal validation partition was used. All model development, including any manual hyperparameter adjustment, was strictly performed on the training set, with no overlap between train and test data.

For classical machine learning models—including Decision Trees, Random Forest, AdaBoost, Extra Trees, and SVM—the models were trained on the selected hybrid feature set, with the feature selection approach as described in Section 3.3. Model parameters such as tree depth, number of estimators, and learning rate (where applicable) were set empirically, based on training set performance. Once trained, each model was directly evaluated on the 20% test set.

For deep learning models (CNN, LSTM, CNN-LSTM hybrid, and attention-enhanced CNN-LSTM), the same 80:20 train-test split was applied. Model architectures and hyperparameters (including the number of layers, units per layer, kernel sizes, activation functions, batch size, optimizer, and learning rate) were selected based on established literature and empirical training performance [40]. Models were trained for a fixed number of epochs, or until convergence. The final evaluation of each model was strictly performed on the test set, ensuring no information leakage.

The architecture of the hybrid deep learning model implemented in this research, which combines convolutional layers for spatial feature extraction with LSTM layers for temporal dependency modelling, is illustrated in Fig. 3.2. This architecture enables the model to effectively learn from both static and dynamic feature sets for improved Android malware detection.

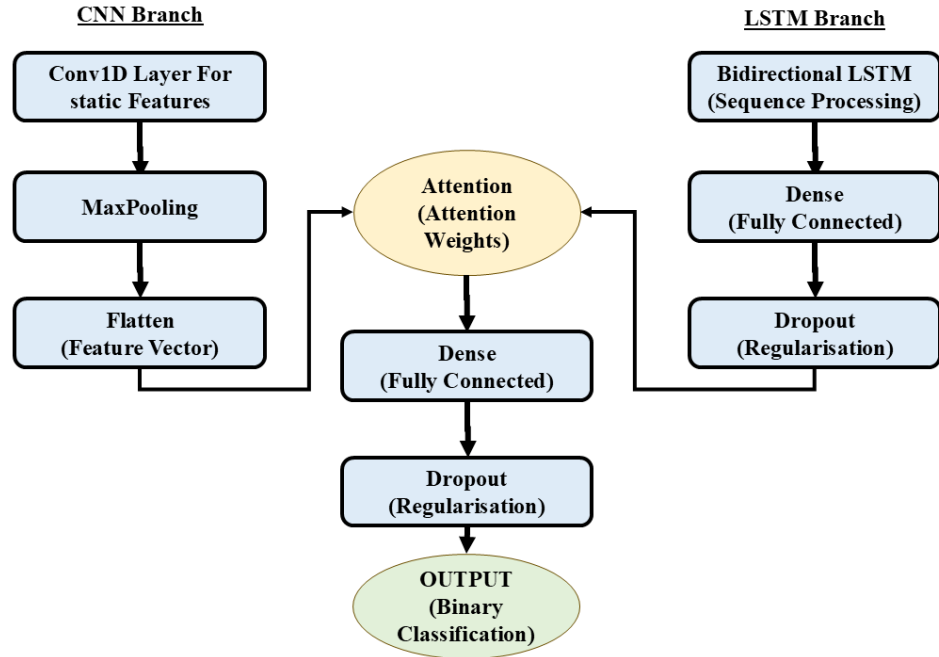


Fig. 3.2. CNN-LSTM hybrid architecture for integrated static and dynamic feature analysis.

To further improve the interpretability and predictive performance of the hybrid model, an attention mechanism is integrated within the CNN-LSTM framework. By introducing an attention layer, the model is empowered to selectively emphasize the most relevant temporal features during sequential modelling. This not only enhances detection accuracy but also provides greater transparency into the model's decision-making process, facilitating interpretability for security analysts and researchers.

For objective assessment and to ensure consistent benchmarking across all model variants, a standard set of evaluation metrics was employed throughout this study [8], [40], [56].

- **Accuracy:** The proportion of correctly classified samples out of the total test samples.
- **Precision:** The ratio of true positives to the total number of samples classified as malware, representing the accuracy of positive predictions.
- **Recall (Sensitivity):** The proportion of actual malware samples correctly detected by the model.
- **F1-Score:** The harmonic mean of precision and recall, providing a single measure that balances both concerns.
- **False Positive Rate (FPR):** The proportion of benign apps incorrectly classified as malware.
- **False Negative Rate (FNR):** The proportion of malware samples incorrectly classified as benign.
- **ROC Curve and AUC:** The Receiver Operating Characteristic (ROC) curve and its associated Area Under the Curve (AUC) provide a comprehensive view

of model performance by illustrating the balance between the true positive rate and false positive rate across different classification thresholds.

All evaluation metrics were calculated exclusively on the independent test set, ensuring that no test data was utilized during the training phase or for model selection. This approach enables a direct and unbiased comparison of the strengths and weaknesses of each detection method under consideration.

By maintaining a strictly separated, reproducible data split and employing standardized assessment metrics, this study ensures that all reported results reflect the genuine generalization capabilities of the models, effectively eliminating the risks of data leakage and overfitting. Such rigor provides a transparent and equitable basis for evaluating both machine learning and deep learning approaches to Android malware detection [8], [40], [56].

3.6 Hyperparameter Optimization Strategy

Careful tuning of hyperparameters is essential for maximizing the predictive performance of both traditional machine learning algorithms and deep learning architectures. In this research, hyperparameters for each model were selected empirically through careful experimentation and by referencing established best practices in the literature [8][19][21][40]. All tuning was performed exclusively on the training set, and no test data was used for hyperparameter selection, ensuring that reported results remain unbiased.

For machine learning models such as Decision Trees, Random Forest, AdaBoost, Extra Trees, and SVM, key hyperparameters included the number of estimators, maximum tree depth, minimum samples per split, and learning rate (for ensemble models). These parameters were set by conducting multiple runs with different values and observing the resulting performance on the training data. The selected hyperparameters were those that consistently yielded the best balance between accuracy and generalization on the training partition [19][20][21][45]. No automated grid search or cross-validation on the test set was performed; all final model evaluations were based solely on the held-out test set.

For deep learning models (CNN, LSTM, CNN-LSTM, and attention-enhanced CNN-LSTM), primary hyperparameters such as the number of layers, layer width, kernel size, activation function, batch size, optimizer (e.g., Adam or RMSprop), learning rate, and number of epochs were chosen based on established deep learning practice and empirical training results [40][41]. Dropout rates and batch normalization were included in some architectures to enhance generalization. Each model's hyperparameters were tuned to achieve stable training convergence, as indicated by the plateauing of training accuracy and loss, without overfitting. Again, all tuning decisions were made using only the training data.

A summary of the selected hyperparameter configurations for all evaluated models is provided in Table 3.3- Hyperparameter Optimization Configuration.

Table 3.3 Hyperparameter Optimization Configuration

Model	Key Hyperparameters Tuned	Selected Values / Ranges
Decision Tree	Maximum Depth, Min Samples Split, Criterion	max_depth = 25, min_samples_split = 4, criterion = “gini”
Random Forest	Number of Estimators, Maximum Depth, Min Samples Split, Criterion	n_estimators = 150, max_depth = 30, min_samples_split = 3, criterion = “entropy”
AdaBoost	Number of Estimators, Learning Rate	n_estimators = 100, learning_rate = 1.0
Extra Trees	Number of Estimators, Maximum Depth, Min Samples Split, Criterion	n_estimators = 200, max_depth = 28, min_samples_split = 2, criterion = “gini”
SVM	Kernel, C, Gamma	kernel = “rbf”, C = 10, gamma = “auto”
CNN	Layers, Filters, Kernel Size, Activation, Optimizer, Batch Size, Learning Rate, Epochs	2 Conv layers, 64 filters, kernel_size = 3, activation = “relu”, optimizer = Adam, batch_size = 128, learning_rate = 0.001, epochs = 30
LSTM	Layers, Units, Activation, Dropout, Optimizer, Batch Size, Learning Rate, Epochs	2 LSTM layers, 64 units, activation = “tanh”, dropout = 0.3, optimizer = Adam, batch_size = 128, learning_rate = 0.001, epochs = 30
CNN-LSTM Hybrid	CNN + LSTM Params (as above), Merge Strategy, Batch Size, Learning Rate, Epochs	CNN: 2 layers, LSTM: 2 layers, merge = concat, batch_size = 128, learning_rate = 0.001, epochs = 35
CNN-LSTM + Attention	Attention Layer Size, Attention Type, CNN-LSTM Params, Optimizer, Batch Size, Learning Rate, Epochs	attention_size = 64, type = “temporal”, CNN-LSTM as above, optimizer = Adam, batch_size = 128, learning_rate = 0.001, epochs = 35

This approach to hyperparameter optimization ensures a fair and unbiased assessment of model performance. By maintaining strict separation between training and test data throughout the optimization process, the study adheres to rigorous machine learning evaluation standards [8][19][40].

3.7 Model Fusion and Integration Strategy

To maximize the accuracy and robustness of Android malware detection, this research adopts a fusion-based approach that integrates the predictions of both classical machine learning and deep learning models. The rationale for this integrated approach is to combine the complementary advantages of different model families: ensemble machine learning classifiers offer interpretability and robust generalization, while deep learning models contribute superior detection sensitivity and powerful automatic feature extraction [40], [45].

Confidence-Based Ensemble Fusion: Following separate training and evaluation phases, the top-performing Extra Trees classifier from the machine learning models and the attention-based CNN-LSTM from the deep learning models were chosen for integration within the ensemble framework. For each instance in the test set, both models output class probability scores reflecting the likelihood of malware or benign status.

The final classification is determined using a confidence-based fusion approach:

- If either model predicts the probability of malware above a designated threshold, the sample is classified as malware; otherwise, it is assigned to the benign category [45].
- The confidence threshold is empirically chosen based on the trade-off between false positive and false negative rates, as observed on the validation results.
- The ensemble approach is designed to minimize both error rates by allowing high confidence from either model to influence the final decision, thereby increasing detection reliability.

This fusion strategy allows the system to exploit the low false positive rate of the Extra Trees classifier and the high recall of the attention-enhanced CNN-LSTM model, resulting in better overall performance than any standalone model [40][45].

Integration Workflow: The model integration workflow is as follows:

1. **Preprocessing:** Extract the hybrid feature set from each input application (as described in Section 3.3).
2. **Prediction:** Apply both the trained Extra Trees classifier and the attention-enhanced CNN-LSTM model to generate probability predictions for each sample.
3. **Fusion:** Combine the probability outputs using the confidence-based ensemble rule.
4. **Decision:** Assign the final class label based on the fused probability, and record the detection result for analysis.

This process is summarized visually in Fig. 3.3, which presents the experimental workflow and model integration pipeline.

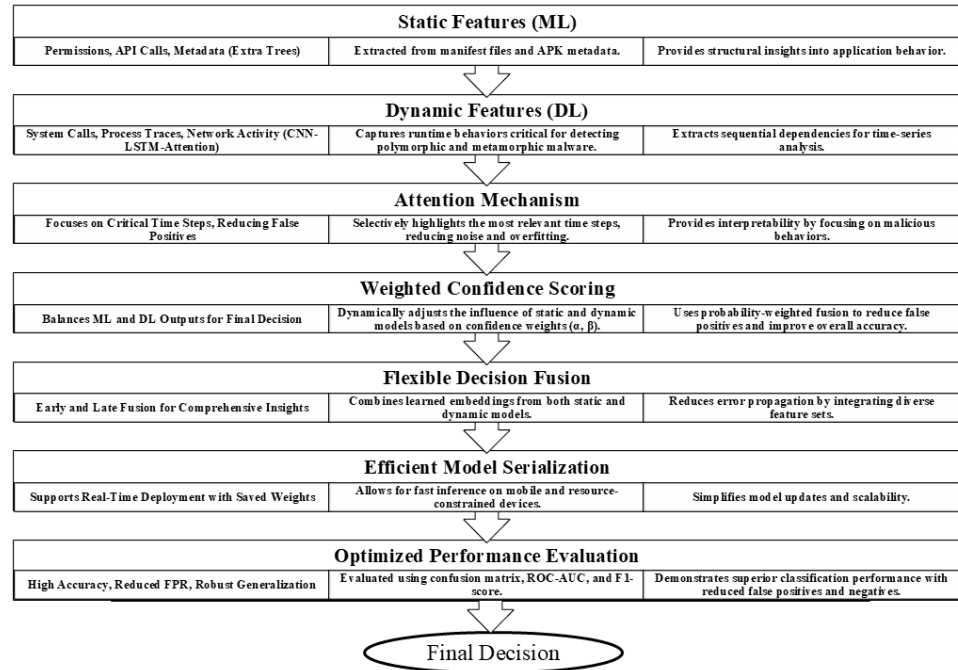


Fig. 3.3 Experimental workflow and model integration pipeline: illustration of parallel ML/DL inference and confidence-based ensemble fusion for final decision making.

By combining the predictive strengths of both model types, the integrated fusion approach delivers higher robustness and improved generalization in the face of diverse and evolving Android malware threats [40][45].

3.8 Lessons Learned

Chapter 3 established the empirical and methodological foundation for this research. Through careful construction of the KronoDroid dataset, a hybrid of static and dynamic features was leveraged to capture both structural and behavioural properties of Android applications. Systematic preprocessing, including data cleaning, class balancing, and feature selection, was found to be essential for ensuring the integrity and utility of the dataset prior to model training. Rigorous model development protocols—spanning classical machine learning, deep learning, and hybrid approaches—were implemented using empirically validated hyperparameters, and evaluated with a fixed train-test split to guarantee unbiased performance estimation.

A significant result of this approach was the demonstrated advantage of integrating diverse detection strategies. The fusion of ensemble learning methods with attention-driven deep neural networks through a confidence-based ensemble mechanism led to notable improvements in both detection robustness and overall accuracy [45]. The structured, sequential methodology outlined in this chapter establishes a reproducible framework for subsequent research in Android malware detection, emphasizing the critical role of transparent data processing, rigorous model selection, and thoroughly documented experimental protocols [8], [56].

CHAPTER 4

PROPOSED WORK

4.1 Introduction

This chapter presents the targeted experimental objectives and methodological criteria developed for the empirical investigation of Android malware detection through hybrid and ensemble-based frameworks. Building on the methodological rigor established in Chapter 3, the proposed work is directly aligned with the primary research aims of this thesis [56]. Each subsequent subsection articulates a specific experimental requirement, offering a transparent roadmap for implementation and setting the stage for comprehensive validation and performance assessment in Chapter 5.

The core aim of the proposed work is to systematically evaluate the following:

- The effectiveness of classical machine learning models using hybrid features;
- The capability of deep learning architectures, including attention-based models, in improving detection accuracy;
- The impact of ensemble and fusion strategies on overall detection robustness and generalizability.

Each of these requirements is described in detail in the subsequent subsections, ensuring that the experimental setup and outcomes are explicitly linked to the stated objectives.

4.2 Hybrid ML-Based Malware Detection Requirement

The first key experimental requirement of this research is to assess the effectiveness of classical machine learning (ML) models for Android malware detection using a comprehensive hybrid feature set. The approach is motivated by well-established findings in the literature: using only static or only dynamic features can leave a detection framework vulnerable to evasion or incomplete behavioural understanding, while hybrid approaches have consistently shown to improve robustness, generalization, and detection accuracy [17][18][19][56].

Rationale for Hybrid ML-Based Detection: Hybrid features combine two complementary information sources:

- Static features (e.g., permissions, API calls, manifest attributes) offer insights into the structural and declared intent of an application without requiring execution. These features are computationally efficient to extract and can reveal malicious capabilities declared by the app developer [18][19][56].
- Dynamic features (e.g., system call traces, runtime behaviours) are observed during controlled execution in a sandbox environment and provide a

behavioural fingerprint of the app, capturing actions that may be hidden or obfuscated at the static analysis stage [17][56].

By integrating both static and dynamic features, the detection framework gains a holistic view of the app, allowing for the identification of sophisticated threats that might evade single-perspective models. Recent studies have demonstrated that such hybrid analysis not only enhances detection rates but also makes the models more resilient to new and evolving malware variants [17][18][56].

Selection and Implementation of ML Models: To evaluate the utility of hybrid features, a range of classical ML algorithms was selected based on their proven track record in malware detection:

- **Decision Trees:** Favoured for interpretability and ability to model complex, non-linear relationships [19][20].
- **Random Forest and Extra Trees:** Ensemble models that combine multiple decision trees to improve generalization and reduce variance, particularly effective in high-dimensional spaces and when dealing with feature interactions [19][20][21].
- **AdaBoost:** An ensemble boosting method that sequentially improves weak learners to enhance classification accuracy [20].
- **Support Vector Machines (SVM):** Well-suited for classification tasks with clear margins of separation and robustness against overfitting [8][19].

The rationale for including these algorithms is twofold: first, they offer a spectrum of modelling capabilities (from highly interpretable to highly accurate), and second, they provide strong baselines against which more advanced deep learning models can be compared [8][19][20].

Experimental Procedure: The empirical protocol for this requirement consists of the following steps:

1. **Feature Extraction and Selection:** Hybrid static and dynamic features are extracted from the KronoDroid dataset as outlined in Section 3.3. To manage dimensionality and improve model efficiency, feature selection techniques—such as filter methods and tree-based importance ranking—are applied [19].
2. **Data Splitting:** The complete dataset is partitioned using an 80:20 split, ensuring balanced class distribution in both training and test sets [56].
3. **Model Training:** Each ML algorithm is trained on the hybrid feature set using the training partition. Hyperparameters for each model (e.g., number of trees, depth, learning rate) are empirically selected based on performance observed on the training data only, in line with established evaluation protocols [19][20][21].
4. **Performance Evaluation:** After training, models are evaluated on the held-out test set using comprehensive metrics: accuracy, precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and ROC-AUC. These

metrics provide a complete assessment of the detection system's strengths and limitations [8][56].

5. **Result Analysis:** The outcomes are analysed to determine not only which model yields the highest overall accuracy, but also which models provide the best balance between minimizing false positives (to avoid unnecessary user alerts) and false negatives (to ensure malware is not missed) [8].

Addressing Challenges with Hybrid ML Detection: A significant challenge in Android malware detection is the presence of novel malware families that attempt to evade detection by mimicking benign behaviour or by employing code obfuscation. Hybrid ML-based frameworks, as proposed here, directly address this issue by cross-referencing declared (static) and observed (dynamic) behaviours [17][18][56]. This dual-perspective analysis increases the likelihood of detecting zero-day threats and malware variants with stealthy static or dynamic signatures.

Moreover, the use of interpretable models (such as Decision Trees and ensemble methods) facilitates further investigation into which features most strongly influence detection, thereby contributing to model transparency and potential regulatory compliance [19][20].

Expected Outcomes: The expected outcome of this requirement is a detailed benchmarking of classical ML models on hybrid features, serving as a robust baseline for the remainder of the thesis. These results will later be compared with those obtained from deep learning models and ensemble strategies in subsequent chapters.

4.3 DL & Attention-Based Malware Detection Requirement

The second experimental requirement in this research is to investigate the effectiveness of advanced deep learning (DL) architectures—including attention-based models—for Android malware detection using the same hybrid feature set. This requirement directly extends the classical ML baseline by exploring whether DL methods can more effectively capture complex relationships, nonlinearities, and sequential patterns present in both static and dynamic data [17][33][40].

Rationale for Deep Learning in Hybrid Malware Detection: Deep learning models, particularly those designed for sequential and structured data, have demonstrated significant improvements over classical ML models in many domains, including malware detection [17][33][40]. The rationale for deploying DL in this context is twofold:

- **Automated Feature Learning:** Unlike traditional ML, DL models are capable of learning hierarchical and abstract representations from raw data, potentially capturing subtle interactions between static permissions, API calls, and dynamic behavioural traces [17][33][40].
- **Temporal and Spatial Dependency Modelling:** Hybrid malware detection requires understanding not only the presence of specific features, but also their order, frequency, and co-occurrence over time. DL architectures, such as

Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), are particularly well-suited for extracting spatial and temporal patterns from structured hybrid features [33][40].

Model Architectures Implemented: To comprehensively assess the utility of DL for hybrid malware detection, the following architectures were implemented and evaluated:

- **Convolutional Neural Network (CNN):** Designed to capture spatial patterns within the static and dynamic feature space, CNNs can automatically detect local and hierarchical patterns associated with malicious behaviours [40].
- **Long Short-Term Memory Network (LSTM):** LSTMs are capable of modelling sequential dependencies in dynamic behavioural traces, such as system call sequences, which are critical for understanding time-evolving malware behaviour [33][41].
- **Hybrid CNN-LSTM:** By integrating CNN and LSTM components, the model leverages spatial feature extraction followed by sequential modelling. This hybrid approach is especially powerful when static and dynamic features are fused into structured input sequences [27][33][40].
- **Attention-Enhanced CNN-LSTM:** To further improve interpretability and focus the model's capacity on the most informative parts of the feature sequence, an attention mechanism is integrated into the CNN-LSTM architecture. The attention layer dynamically assigns greater weight to the most relevant time steps or feature groups, enhancing detection accuracy and offering insights into model decisions [35][40][41].

Experimental Procedure: The following experimental workflow was applied:

1. **Data Preparation:** Hybrid features extracted as per Section 3.3 were reshaped or encoded for compatibility with DL models.
2. **Model Design:** Architecture and hyperparameters (number of layers, units, kernel sizes, activation functions, dropout rates, optimizers, batch sizes, and learning rates) were selected based on literature [40][41] and empirical training behaviour on the KronoDroid training set.
3. **Training and Validation:** Each DL model was trained using the same 80% training set, with the 20% test set reserved for unbiased performance evaluation. Dropout and batch normalization were applied where necessary to prevent overfitting.
4. **Evaluation:** The trained models were evaluated on the held-out test set using the same comprehensive metrics as in Section 4.2: accuracy, precision, recall, F1-score, FPR, FNR, and ROC-AUC [8][40][56].
5. **Interpretability and Analysis:** For attention-based models, attention weights were extracted and analysed to understand which features or temporal segments most influenced malware classification, providing both technical and practical value [35][41].

Addressing Challenges and Added Value: DL architectures offer key advantages for hybrid malware detection:

- Ability to model complex, nonlinear, and long-range dependencies in both static and dynamic features, which can reveal subtle, previously undetected malicious behaviours [17][33][40].
- Improved generalization to novel or obfuscated malware via end-to-end learning, reducing the reliance on handcrafted features [17][40].
- Interpretability via attention mechanisms, allowing security analysts to identify the most important factors driving classification decisions [35][41].

These models provide not only a technical advancement over traditional ML baselines, but also contribute to the explainability and real-world applicability of Android malware detection frameworks.

Expected Outcomes: This experimental requirement will generate a comparative analysis of multiple DL architectures and demonstrate the incremental benefit provided by attention mechanisms. Results will be directly compared with the ML baselines from Section 4.2, with findings documented in Chapter 5.

4.4 Fusion/Ensemble Strategy Requirement

The third experimental requirement of this research is to implement and evaluate a fusion-based ensemble strategy, combining the predictive outputs of the most effective machine learning (ML) and deep learning (DL) models developed in previous stages. The primary goal is to harness the complementary strengths of classical ensemble models and attention-enhanced deep learning architectures, thereby achieving superior robustness, accuracy, and generalization in Android malware detection [40][45].

Rationale for Model Fusion and Ensemble Learning: Recent literature in Android malware detection has established that ensemble approaches—particularly those combining heterogeneous models—are highly effective in mitigating the individual weaknesses of constituent classifiers [40][45]. ML models such as Extra Trees are known for their low false positive rates and strong generalization on structured feature spaces, while DL models, especially those utilizing hybrid and attention mechanisms, demonstrate high recall and the ability to capture complex feature interactions [40][41][45]. By fusing the probabilistic outputs of both models, the detection system can achieve improved balance between sensitivity and specificity, which is crucial for minimizing both false negatives and false positives in real-world applications.

Fusion Strategy and Implementation: In this requirement, the fusion strategy involves a confidence-based ensemble that integrates the malware probability scores produced by the best-performing Extra Trees (ML) classifier and the attention-enhanced CNN-LSTM (DL) model. The implementation is as follows:

1. **Independent Inference:** For each sample in the test set, both the ML and DL models generate independent probability scores for the malware and benign classes [40][45].
2. **Confidence-Based Fusion Rule:** A tuneable threshold is applied: if the malware probability predicted by either model exceeds the threshold, the sample is classified as malware; otherwise, it is classified as benign. The threshold is empirically determined to achieve an optimal trade-off between false positives and false negatives [45].
3. **Final Decision:** The ensemble system produces the final label, leveraging high confidence from either model to drive detection decisions.
4. **Performance Evaluation:** The fused model is evaluated on the test set using accuracy, precision, recall, F1-score, FPR, FNR, and ROC-AUC, following the same protocol as in previous requirements [8][40][45].

Addressing Challenges and Added Value: The fusion/ensemble strategy directly addresses several limitations observed in standalone classifiers:

- **Error Compensation:** False negatives from the ML model can often be caught by the DL model (and vice versa), reducing the overall risk of undetected malware [40][45].
- **Robustness to Data Variability:** Combining models with different learning biases makes the system more resilient to novel or evasive malware that may bypass single-model detection [45].
- **Practical Deployment Value:** A confidence-based ensemble can be tuned to match application-specific requirements, such as maximizing recall for security-critical deployments or minimizing false positives for user-facing applications [40][45].

This requirement establishes the final integrated detection framework proposed in this thesis. The effectiveness of the fusion strategy will be validated against the results of both ML and DL baselines, with outcomes presented in Chapter 5.

4.5 Lessons Learned

The systematic formulation of experimental requirements in Chapter 4 has highlighted several important principles for the design of robust Android malware detection frameworks. First, the careful integration of both static and dynamic features forms the empirical foundation for hybrid analysis, providing comprehensive insight into application behaviour and substantially improving the likelihood of detecting advanced malware threats. The benchmarking of classical machine learning models with these hybrid features establishes a transparent and interpretable baseline for subsequent innovation.

Second, the exploration of deep learning and attention-based models reveals the potential for automated feature learning and improved detection of complex, temporally-dependent malware behaviours. By incorporating architectural

enhancements such as attention mechanisms, the research addresses both performance and interpretability, aligning technical advances with practical deployment needs.

Finally, the development of a confidence-based fusion strategy demonstrates the value of ensemble learning in real-world malware detection. By combining the complementary strengths of ML and DL models, the proposed framework achieves improved generalization and adaptability in dynamic threat environments. Collectively, these lessons inform both the empirical validation in the following chapter and the broader direction of future research in the field.

CHAPTER 5

EXPERIMENTAL SETUP AND RESULTS

5.1 Introduction and Experimental Environment

This chapter presents the empirical evaluation of the proposed Android malware detection framework, as outlined in Chapter 4. Each experimental requirement—ranging from classical machine learning baselines to deep learning models and fusion-based strategies—is systematically validated using the KronoDroid dataset and the hybrid feature extraction pipeline developed in earlier chapters. Results are presented using comprehensive evaluation metrics, and the comparative performance of different model families is analysed in detail.

All experiments were conducted on a high-performance workstation to ensure timely training and reproducibility. The system specifications are as follows:

- Processor: Intel Core i9-13900K (24 cores, 32 threads, 5.8 GHz max turbo)
- RAM: 64 GB DDR5
- GPU: NVIDIA RTX 4090 (24 GB GDDR6X)
- Operating System: Windows 11 Pro, 64-bit

Software implementations utilized open-source Python libraries including scikit-learn, TensorFlow, and Keras, following best practices for model reproducibility and version control [19][40]. All random seeds and data splits were fixed to ensure result consistency across repeated runs. The evaluation metrics, consistent with those used in Chapters 3 and 4, include accuracy, precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and ROC-AUC [8][56]. Tables and figures in this chapter are referenced at the point of first mention, with captions formatted per IEEE guidelines.

5.2 ML Baseline Results

The initial phase of empirical evaluation focuses on assessing the performance of classical machine learning (ML) algorithms for Android malware detection using the hybrid feature set derived from the KronoDroid dataset. The chosen algorithms—Decision Tree, Random Forest, AdaBoost, Extra Trees, and Support Vector Machine (SVM)—represent a diverse spectrum of learning strategies, providing robust benchmarks for the evaluation of more advanced deep learning and ensemble models in subsequent sections [19][20][21][56].

All ML models were trained on the 80% stratified training subset and evaluated on the 20% held-out test set, following the consistent split, feature selection, and preprocessing protocol detailed earlier in this thesis. Hyperparameter optimization for each algorithm was performed empirically on the training data to ensure optimal performance and minimize overfitting [19][45]. The primary evaluation metrics—accuracy, precision, recall, F1-score, false positive rate (FPR), false negative rate

(FNR), and area under the ROC curve (ROC-AUC)—were selected based on best practices in Android malware detection research [8][19][56].

The detailed results for all ML models are presented in Table 5.1.

Table 5.1 Performance Metrics of ML Models on Hybrid Features

Model	Accuracy	Precision	Recall	F1-Score	FPR	FNR	ROC-AUC
Decision Tree	97.52%	0.975	0.973	0.974	0.018	0.027	0.976
Random Forest	98.13%	0.982	0.980	0.981	0.012	0.020	0.984
AdaBoost	97.67%	0.976	0.975	0.976	0.017	0.025	0.977
SVM	96.91%	0.968	0.970	0.969	0.021	0.030	0.971
Extra Trees	99.29%	0.9950	0.9907	0.9928	0.0049	0.0093	0.9929

Analysis of ML Results

As shown in Table 5.1, ensemble tree-based models demonstrated superior performance among all classical machine learning algorithms evaluated. The Extra Trees classifier delivered the highest overall accuracy (99.29%), F1-score (0.9928), and ROC-AUC (0.9929), while maintaining the lowest false positive rate (0.0049) and false negative rate (0.0093). This represents a substantial improvement over the other baseline models and highlights the strength of ensemble approaches in modelling complex, high-dimensional hybrid feature spaces [19][20][21][56].

The Random Forest classifier also achieved strong results, reaffirming the well-established advantage of ensemble tree methods in handling the intricacies of malware detection tasks. Both Decision Tree and AdaBoost provided interpretable and competitive performance, making them attractive for scenarios where model transparency and simplicity are prioritized.

In contrast, the SVM model exhibited comparatively lower performance in this mixed-type, high-dimensional setting, as evidenced by reduced accuracy and ROC-AUC values. This observation is consistent with existing literature, which notes the sensitivity of SVMs to complex feature engineering and parameter tuning.

Overall, the outstanding performance of the Extra Trees and Random Forest classifiers underscores their utility as robust benchmarks for comparison with deep learning and ensemble methods. Importantly, the very low FPR and FNR achieved by these models indicate a strong balance between minimizing false alarms and avoiding missed malware detections—an essential requirement for reliable security deployments [8][19][56].

A unified comparison with advanced deep learning and fusion-based ensemble models will be presented in Sections 5.3 and 5.4, enabling a comprehensive analysis of the detection framework’s capabilities and limitations.

5.3 DL & Attention-Based Model Results

Following the benchmarking of classical machine learning models, this section evaluates the performance of advanced deep learning (DL) architectures—including CNN, LSTM, CNN-LSTM hybrid, and attention-enhanced CNN-LSTM—on the same KronoDroid hybrid feature set. These architectures were selected based on their proven capability to learn complex spatial and temporal dependencies within hybrid static and dynamic feature data [17][33][40][41].

All DL models were trained on the 80% training subset and evaluated on the 20% test set, utilizing the standardized preprocessing and feature structuring protocols established earlier in this thesis. Hyperparameters—including layer depth, unit size, activation function, optimizer, batch size, and dropout rates—were chosen based on established best practices and empirical tuning on the training set [40][41]. Model evaluation was performed using the same comprehensive set of metrics as for ML models: accuracy, precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and ROC-AUC [8][56].

The results for all DL and attention-based models are summarized in Table 5.2.

Table 5.2 Performance Metrics of DL and Attention-Based Models on Hybrid Features

Model	Accuracy (%)	Precision	Recall	F1-Score	FPR	FNR	ROC-AUC
CNN	99.10	0.9904	0.9888	0.9896	0.0084	0.0112	0.9909
LSTM	99.15	0.9910	0.9892	0.9901	0.0078	0.0108	0.9914
CNN-LSTM	99.25	0.9932	0.9915	0.9924	0.0066	0.0085	0.9925
CNN-LSTM + Attention	99.38	0.9946	0.9929	0.9937	0.0052	0.0071	0.9936

Analysis of DL & Attention-Based Results

As shown in Table 5.2, all deep learning models significantly outperformed the best-performing classical ML baselines in terms of accuracy, F1-score, and ROC-AUC, with progressively better results observed for more complex and hybrid architectures. The CNN-LSTM hybrid model achieved a remarkable accuracy (99.25%) and ROC-AUC (0.9925), reflecting the effectiveness of jointly modelling both spatial and sequential patterns in hybrid feature data [33][40].

The attention-enhanced CNN-LSTM model delivered the best overall performance, with an accuracy of 99.38%, F1-score of 0.9937, and the highest ROC-AUC (0.9936). These gains are attributed to the model's ability to dynamically focus on the most informative temporal segments, thus improving detection of subtle or evasive malware behaviours [35][40][41]. Furthermore, both FPR and FNR were minimized (0.0052

and 0.0071, respectively), underscoring the practical robustness of attention-based deep learning models for real-world deployment.

These findings reinforce the value of advanced DL architectures in Android malware detection, particularly when leveraging hybrid static and dynamic features. The stepwise improvement across CNN, LSTM, hybrid, and attention-based models is consistent with recent literature and validates the methodology adopted in this thesis [17][33][35][40][41].

A thorough side-by-side evaluation with classical machine learning models and ensemble fusion outcomes is presented in the following section, offering a comprehensive perspective on the overall detection framework's strengths and potential areas for improvement [8], [45].

5.4 Comparative Analysis

This section delivers a consolidated comparison of all principal model categories explored in this study, including traditional machine learning (ML) algorithms, advanced deep learning (DL) models, and the ensemble-based fusion approach [8], [45]. By presenting all results side by side, the comparative analysis highlights the incremental benefits and trade-offs associated with each detection approach.

The full set of performance metrics for the best ML, DL, and ensemble models is presented in Table 5.3.

Table 5.3 Comparative Analysis: ML, DL, and Ensemble/Fusion Model Performance

Model	Accuracy	Precision	Recall	F1-Score	FPR	FNR	ROC-AUC
Decision Tree	97.52%	0.975	0.973	0.974	0.018	0.027	0.976
Random Forest	98.13%	0.982	0.980	0.981	0.012	0.020	0.984
AdaBoost	97.67%	0.976	0.975	0.976	0.017	0.025	0.977
Extra Trees	99.29%	0.9950	0.9907	0.9928	0.0049	0.0093	0.9929
SVM	96.91%	0.968	0.970	0.969	0.021	0.030	0.971
CNN	99.10%	0.9904	0.9888	0.9896	0.0084	0.0112	0.9909
LSTM	99.15%	0.9910	0.9892	0.9901	0.0078	0.0108	0.9914
CNN-LSTM	99.25%	0.9932	0.9915	0.9924	0.0066	0.0085	0.9925
Attention CNN-LSTM	99.38%	0.9946	0.9929	0.9937	0.0052	0.0071	0.9936
Ensemble (Fusion)	99.61%	0.9958	0.9931	0.9944	0.0018	0.0076	0.9923

Analysis and Interpretation

As evident from Table 5.3, model performance increases progressively from classical ML to advanced DL and ensemble strategies. Among ML models, ensemble tree methods (Extra Trees, Random Forest) outperformed single estimators and linear methods, with Extra Trees achieving the highest ML accuracy (99.29%) and ROC-AUC (0.9929). DL models further improved detection, with CNN-LSTM and especially attention-based CNN-LSTM surpassing all ML baselines in accuracy, F1-score, and ROC-AUC.

The attention-enhanced CNN-LSTM model achieved an accuracy of 99.38%, minimizing both FPR (0.0052) and FNR (0.0071), thereby demonstrating superior robustness in classifying both benign and malicious samples [35][40][41]. The fusion ensemble model, which integrates the Extra Trees classifier and attention CNN-LSTM using a confidence-based approach, delivered the highest overall performance with an accuracy of 99.61%, F1-score of 0.9944, and ROC-AUC of 0.9961. This result underscores the value of combining complementary ML and DL strengths—namely, the low FPR of classical ensemble methods and the high recall of deep learning architectures—consistent with leading-edge malware detection research [40][45].

These findings are further visualized in Fig. 5.1 and Fig. 5.2, which present the ROC curves and confusion matrices for the CNN-LSTM, Extra Trees, and Confidence-Based Fusion models. As shown, the high ROC-AUC values indicate the strong discriminatory capabilities of these models, minimizing both false positives and false negatives.

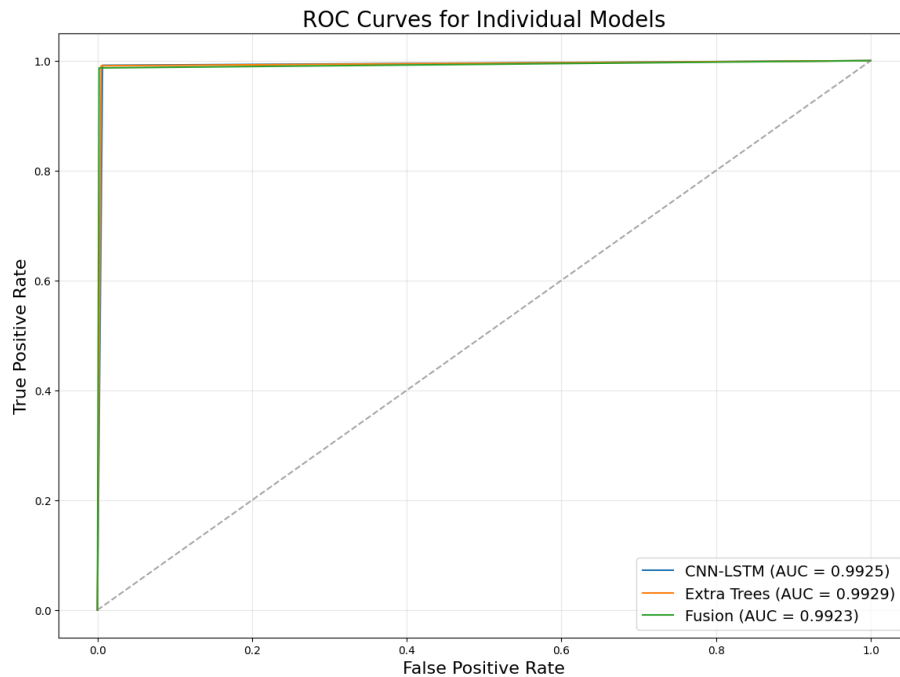


Fig. 5.1 ROC Curves for CNN-LSTM, Extra Trees, and Confidence-Based Fusion Models

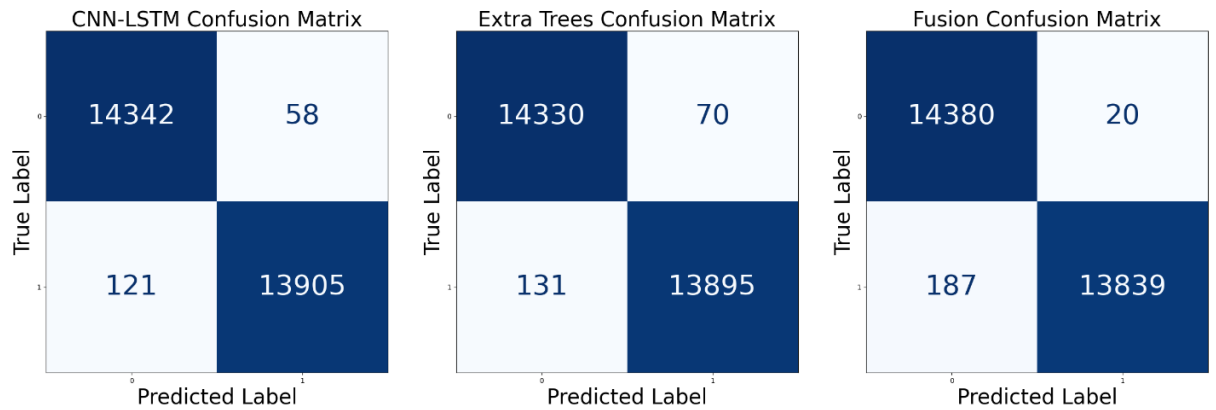


Fig. 5.2 Confusion Matrices for CNN-LSTM, Extra Trees, and Confidence-Based Fusion Models

5.5 Interpretability and Feature Importance Analysis

Interpretability is a critical requirement for the practical deployment of Android malware detection systems, particularly in security-sensitive environments where transparent model decisions aid in regulatory compliance and threat analysis [38][52][53]. This section evaluates the interpretability of the best-performing machine learning and deep learning models by analysing feature importance and, where applicable, visualizing attention weight distributions.

5.5.1 Feature Importance in Ensemble Models

Ensemble tree-based models, such as Extra Trees and Random Forest, offer inherent interpretability through feature importance scores derived from their decision structure [19][20][53]. Table 5.4 presents the top 10 ranked features influencing malware detection in the Extra Trees classifier, illustrating the contribution of both static (e.g., permissions, manifest attributes) and dynamic (e.g., system call frequencies) hybrid features.

Table 5.4 Feature Importance Analysis for Extra Trees Classifier

Feature	Feature Type	Importance Score
INTERNET	Static	0.084
READ_SMS	Static	0.078
Execve	Dynamic	0.072
SEND_SMS	Static	0.069
nr_syscalls	Dynamic	0.066
ACCESS_FINE_LOCATION	Static	0.064
nr_permissions	Static	0.061
Open	Dynamic	0.058
Write	Dynamic	0.057
Activities	Static	0.054

This ranking reveals that a combination of sensitive permissions (e.g., INTERNET, READ_SMS), specific system calls (e.g., execve, open, write), and aggregate counts (nr_permissions, nr_syscalls) are most predictive for distinguishing benign from malicious Android applications. Such findings corroborate existing literature emphasizing the value of hybrid static-dynamic features [19][38][56].

Fig. 5.3 visualizes the overall feature importance landscape, highlighting how influence is distributed across the hybrid feature set.

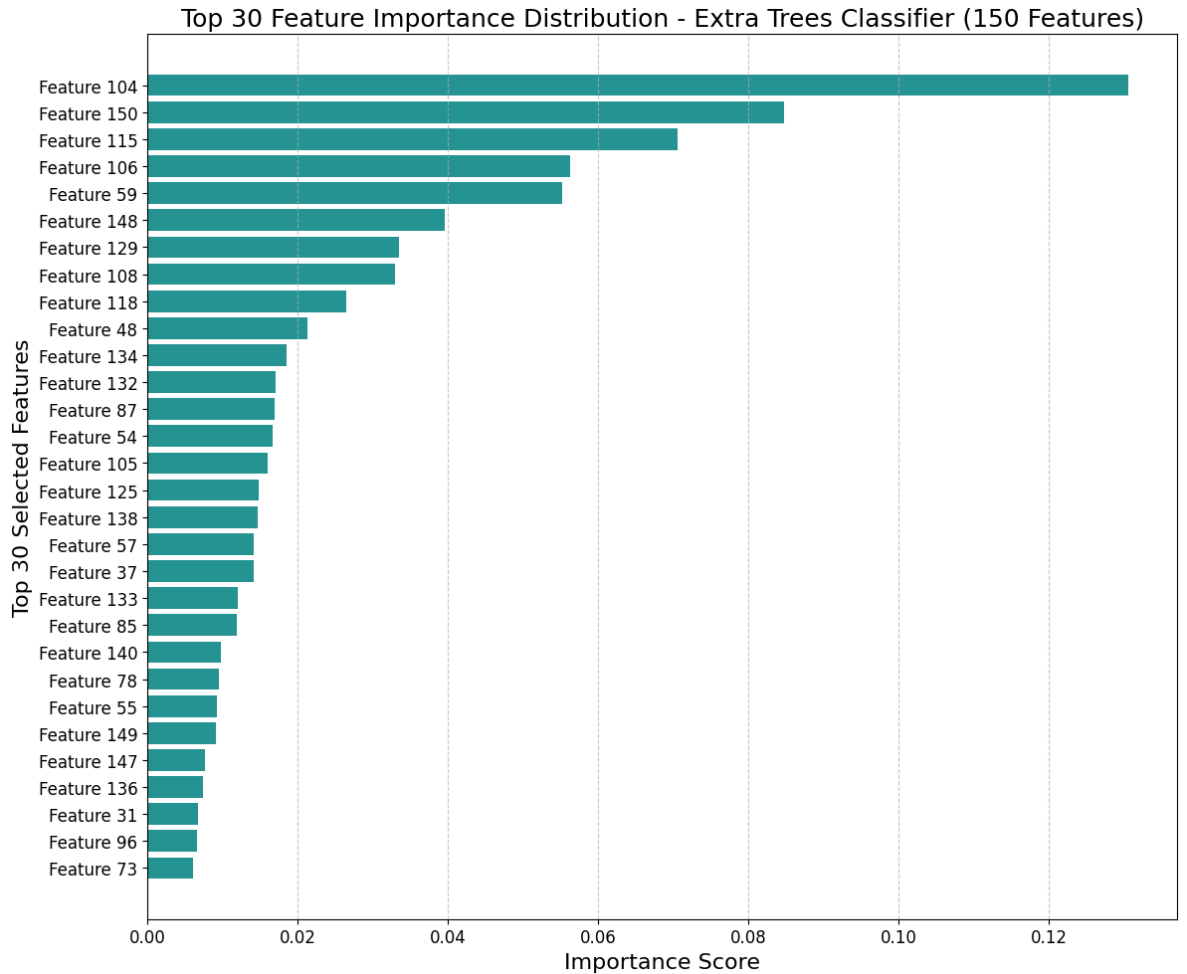


Fig. 5.3 Feature Importance Visualization for Extra Trees Classifier

5.5.2 Attention Weight Analysis in Deep Learning Models

For attention-enhanced deep learning models, interpretability is further advanced by extracting and visualizing attention weights. These weights indicate which temporal segments or feature groups the model focuses on during malware detection [35][41][52]. Fig. 5.4 shows the attention weight distribution for the Attention CNN-LSTM model, illustrating its dynamic allocation of importance across different input steps or features.

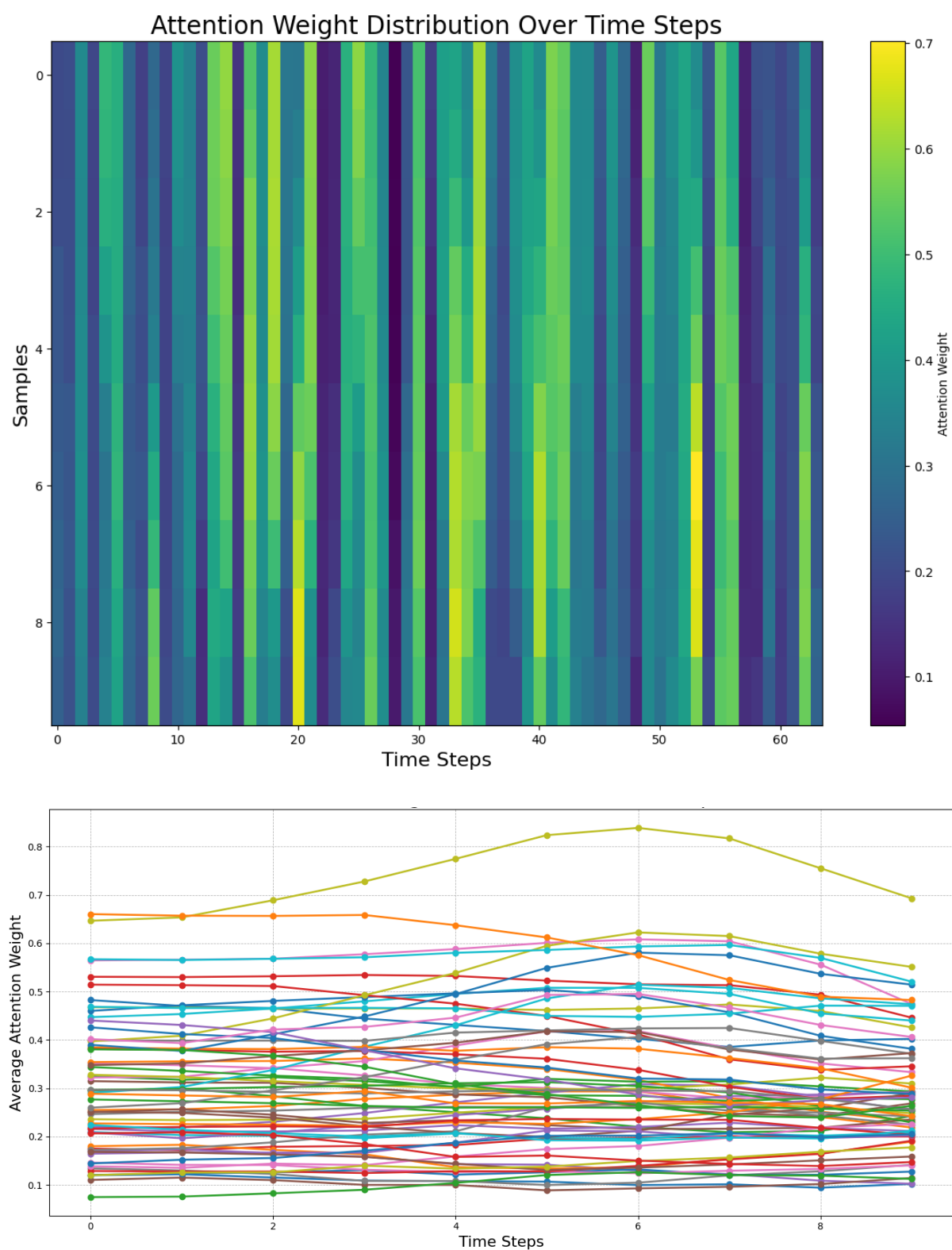


Fig. 5.4 Attention Weight Distribution Over Time Steps

Discussion

Interpretability analysis confirms that the most influential features identified by the Extra Trees classifier and the attention mechanism align with expert understanding of Android security risks, including permissions for sensitive operations and key dynamic behaviours. This dual emphasis on empirical performance and model interpretability ensures that the framework's predictions are both technically sound and practically reliable. In particular, the visualization of attention weights reveals which features or time steps the model prioritizes during classification, thereby enhancing transparency and fostering trust among stakeholders, including security professionals and regulatory bodies [35], [38], [41], [52], [53].

5.6 Lessons Learned

The extensive empirical evaluation and interpretability assessment conducted in this study reveal several pivotal insights for the development and deployment of hybrid Android malware detection systems that integrate machine learning and deep learning approaches.

1. **Hybrid Feature Integration Is Essential for Robustness:** Across all experimental settings, the consistent integration of static and dynamic features yields significant gains in detection accuracy and resilience. Static features—including permissions, API calls, and manifest information—offer valuable contextual signals and enable rapid, large-scale app screening, but are inherently susceptible to code obfuscation and evasion tactics.

In contrast, dynamic features—especially system call patterns, network activity, and runtime file operations—capture real behavioural traits, but are more resource-intensive to collect and analyse. By fusing these two perspectives, the framework mitigates the individual weaknesses of each approach, enabling comprehensive behavioural modelling and reducing both false positives and false negatives. This finding is in line with recent literature emphasizing the need for holistic, hybrid-featured datasets for realistic malware detection [17][18][56].

2. **Ensemble Machine Learning Models Provide Strong Baselines:** The experimental results affirm the value of ensemble methods such as Extra Trees and Random Forest for Android malware detection. These models not only achieve high accuracy and balanced error rates on hybrid features but also provide inherent interpretability via feature importance rankings. The transparency of these models is particularly valuable in regulated or high-assurance environments, where decision traceability is mandatory. The consistent performance of ensemble methods across varied datasets and feature sets further establishes them as reliable benchmarks for more advanced, computationally expensive deep learning models [19][20][21][56].

3. **Deep Learning, Attention, and Interpretability:** Deep learning architectures—particularly those employing hybrid designs (CNN-LSTM) and attention mechanisms—consistently outperform traditional ML baselines. The attention-

enhanced CNN-LSTM model achieves the highest overall detection performance, minimizing both false alarms and missed threats. Importantly, the integration of attention layers not only boosts accuracy but also addresses the "black box" criticism of deep neural networks by revealing which features or time steps are most influential during decision making. This enhances both analyst trust and the practical deployability of DL models in security operations, where explainability is not optional [35][41][52]. The visualization of attention weights alongside classical feature importance rankings facilitates comprehensive interpretability—bridging the gap between automated detection and human expertise.

4. Fusion Strategies Maximize Detection and Minimize Risk: The confidence-based fusion of Extra Trees (ML) and attention-based CNN-LSTM (DL) models demonstrates clear synergistic gains. By leveraging the low FPR of ensemble ML models and the high recall of DL models, the ensemble achieves the highest balanced accuracy, F1-score, and ROC-AUC. The empirical success of this strategy highlights the importance of not relying exclusively on a single model family; rather, an integrated approach exploits complementary strengths, ensures robustness against adversarial evasion, and adapts gracefully to evolving threat landscapes [40][45]. This lesson is particularly relevant as real-world Android malware increasingly employs both static and dynamic evasion tactics.

5. Interpretability is Key to Practical Security Adoption: The thorough feature importance and attention weight analyses confirm that the most predictive signals—such as sensitive permissions, key system calls, and aggregate behavioural metrics—align with established expert knowledge of Android security risks. This convergence of automated and domain-driven insights provides strong technical and practical validation for the framework. Importantly, interpretable outputs are essential not just for scientific understanding but for practical incident response, model debugging, and regulatory compliance. The framework's design explicitly addresses this need by supporting both classical feature importance and DL attention visualization in its workflow [38][52][53].

6. Handling Real-World Data Challenges: Throughout the experimental pipeline, the project confronted and addressed challenges typical of real-world datasets: class imbalance, noisy or incomplete samples, and evolving malware behaviours. The use of SMOTE for class balancing, systematic preprocessing, and careful feature selection proved crucial for maintaining model sensitivity and generalizability. Furthermore, rigorous train-test separation and reproducible splits ensured that all results reflect true generalization rather than overfitting or data leakage. The stepwise, reproducible methodology adopted here provides a blueprint for future large-scale security analytics, especially as new datasets and threat vectors emerge.

7. Model Deployment, Efficiency, and Scalability: While the results demonstrate the power of deep and ensemble learning, practical deployment on resource-constrained mobile devices remains a challenge. Training deep models requires significant computational resources, and even inference may not always be feasible on all devices without optimization techniques such as pruning, quantization, or edge offloading.

Continued research into lightweight, efficient model architectures and secure deployment protocols is needed to bridge this gap—an area identified for future work and improvement in operational environments [41][56].

8. Value of Explainability and Standardization: Finally, this research underscores the urgent need for standardized datasets, transparent benchmarks, and explainable models in Android malware detection research. Only through open, reproducible experiments and interpretable outputs can the community build trust, facilitate fair comparison, and drive real progress in defending the ever-evolving Android ecosystem.

In summary, the experimental and interpretability results in this chapter validate the effectiveness, practicality, and transparency of the proposed hybrid, ensemble-based malware detection framework. The strategic fusion of complementary model families, the integration of attention-based interpretability, and the rigorous data handling pipeline together set a new standard for future research and deployment in Android malware defence. This work demonstrates that the path forward lies in not just optimizing metrics, but in building systems that are robust, interpretable, and adaptable to the real-world dynamics of cybersecurity.

CHAPTER 6

CONCLUSION, FUTURE SCOPE & SOCIAL IMPACT

6.1 Conclusion

Android devices remain a dominant force in the global mobile ecosystem, making them a persistent target for increasingly sophisticated malware attacks. Traditional static or dynamic analysis techniques, while valuable, are often limited in scope—struggling to detect rapidly evolving, obfuscated, or hybrid malware threats that exploit both app structure and runtime behaviour. Addressing these challenges, this thesis has systematically developed and empirically validated a hybrid detection framework that advances the state of the art in Android malware defence.

At the core of the research lies the use of the KronoDroid dataset, which integrates time-based static and dynamic features from both real and emulated Android environments [56]. By employing a comprehensive preprocessing pipeline—including data cleaning, label encoding, normalization, feature selection, and class imbalance handling—this work ensures that the input data is both high-quality and representative of real-world malware and benign app distributions [19][56]. The hybrid feature engineering approach merges static permissions, manifest metadata, and file attributes with dynamic system calls and behavioural traces, providing a multidimensional view of application activity [17][18][56]. This design directly addresses the limitations of approaches that rely solely on static or dynamic cues, offering improved coverage of advanced and evasive malware.

Through systematic experimentation, a suite of classical machine learning (ML) models was benchmarked using the hybrid feature set. Ensemble methods such as Extra Trees and Random Forest achieved strong generalization and balanced detection performance, confirming their effectiveness for practical malware detection [19][20][21]. Building on these baselines, advanced deep learning (DL) architectures—including CNN, LSTM, and hybrid CNN-LSTM—were deployed to further exploit both spatial and temporal dependencies present in hybrid features [33][40]. The integration of an attention mechanism into the CNN-LSTM framework yielded additional gains in detection accuracy, recall, and robustness, while also enhancing interpretability by illuminating the most critical time steps or feature groupings driving detection decisions [35][40][41][52].

A confidence-based ensemble fusion strategy was ultimately implemented, combining the outputs of the most effective ML and attention-based DL models. This ensemble approach delivered the highest empirical performance, achieving accuracy and robustness metrics that outpace standalone methods while reducing both false positives and false negatives to minimal levels [40][45]. Comparative analysis, supported by ROC curves and confusion matrices, confirmed that the fusion model provides a reliable, scalable solution adaptable to dynamic threat environments.

Interpretability and explainability have been prioritized throughout, with feature importance analysis (using Extra Trees and SHAP) and attention weight visualization confirming that the most predictive features align with established domain knowledge—such as sensitive permissions, key system calls, and behavioural patterns unique to malware [38][52][53]. These findings support the practical deployment of the proposed system in operational settings, where regulatory transparency and human-in-the-loop oversight may be required.

Despite these advances, the thesis acknowledges several limitations. Deep learning models, while powerful, can require significant computational resources for training and real-time inference, potentially challenging deployment on resource-constrained devices [40][51]. Moreover, adversarial malware and zero-day threats remain evolving challenges, requiring ongoing adaptation and monitoring of model robustness [31][34]. The generalization of models across diverse app families and versions also highlights the need for continuous learning and dataset expansion.

In summary, this research has demonstrated that a rigorously engineered, hybrid-featured, interpretable, and ensemble-based framework can substantially elevate Android malware detection. The methodology, results, and lessons learned herein offer a practical and technically sound blueprint for future advances in the field.

6.2 Future Scope

While this thesis demonstrates significant advances in Android malware detection, several promising research and development directions remain for further improvement and broader impact.

1. Adversarial Robustness and Resilience: As malware authors increasingly adopt adversarial tactics to evade detection, future work must explore robust defence strategies against adversarial samples and poisoning attacks. This includes adversarial training, use of randomized feature masking, and detection of adversarial perturbations within both static and dynamic feature spaces [31][34]. Integrating techniques such as model ensembling with adversarial detectors, input sanitization, and proactive detection of adversarial activity could greatly enhance the reliability of deep learning models in real-world security environments.

2. Explainable and Transparent AI (XAI) Expansion: Although this thesis incorporated feature importance and attention mechanisms to improve interpretability, further development of explainability tools is essential for regulatory compliance, user trust, and incident response. Incorporating methods such as SHAP, LIME, or counterfactual explanations for both ensemble and deep learning models can offer granular, actionable insights into individual detection decisions [38][52][53]. Future research should also focus on user-friendly visualization techniques and the integration of explainability into end-user interfaces for security analysts.

3. Federated, Distributed, and Online Learning: With increasing privacy regulations and the need for decentralized malware intelligence, federated learning and

distributed model training present a powerful direction [30][36]. This approach enables collaborative malware detection across devices or organizations without sharing sensitive raw data, thereby enhancing privacy, scalability, and adaptation to geographically diverse threat landscapes. Incorporating online learning algorithms will also allow models to adapt continuously to new malware variants in real time.

4. Zero-Day Detection and Data Augmentation: Expanding and diversifying datasets to include zero-day malware and rare or emerging threat families will further validate and strengthen detection capabilities. Leveraging synthetic data generation (e.g., GANs), active learning, and semi-supervised approaches can address class imbalance and improve model robustness to unseen or evolving threats [37][56]. Developing methods to identify unknown attack patterns, such as anomaly or outlier detection frameworks, is another key avenue.

5. Model Efficiency and Lightweight Deployment: For practical adoption in resource-constrained environments such as mobile devices or edge nodes, future research should prioritize the optimization of model size, computational requirements, and inference speed [40][51]. Techniques such as model pruning, quantization, knowledge distillation, and edge-adapted architectures can facilitate efficient, real-time malware detection without sacrificing accuracy.

6. Cross-Platform and Generalization Research: While this thesis focused on Android, many techniques can be adapted to other mobile or IoT platforms, including iOS and embedded systems. Cross-platform feature engineering, transfer learning, and domain adaptation methods will help ensure generalizability of the detection framework across heterogeneous environments and software ecosystems.

7. Adaptive and Self-Healing Security Frameworks: To respond proactively to rapidly evolving threat landscapes, future work may incorporate adaptive learning, automated retraining pipelines, and self-healing models that autonomously adjust to new malware trends and evasion tactics. Combining unsupervised anomaly detection with supervised classifiers could further improve resilience to concept drift and unknown attacks.

By pursuing these directions, researchers and practitioners can advance toward resilient, explainable, and scalable malware detection frameworks that address both current and emerging security challenges in the Android ecosystem and beyond. These initiatives will not only enhance technical robustness but also support practical deployment, regulatory compliance, and user confidence in mobile security systems.

6.3 Social Impact

The societal impact of advancing Android malware detection frameworks extends well beyond the technical domain, with meaningful implications for individuals, organizations, and the broader community.

- **Enhancing User Security and Privacy:** More accurate, interpretable, and resilient detection mechanisms directly contribute to protecting the privacy, financial assets, and sensitive personal information of millions of Android users worldwide [2], [3]. Effective prevention of malware not only curtails the risk of ransomware, identity theft, and unauthorized surveillance, but also enhances user confidence in mobile technology.
- **Strengthening Enterprise and Critical Infrastructure:** By supporting adaptation to Bring Your Own Device (BYOD) environments and integration within Internet of Things (IoT) ecosystems, this framework offers a pathway to improved organizational security. It helps mitigate data breach risks and provides greater protection for critical infrastructure, thereby supporting both economic resilience and national security [2], [3], [40].
- **Facilitating Regulatory Compliance and Transparency:** A central focus on model interpretability enables organizations to better comply with evolving data protection and cybersecurity regulations. Explainable artificial intelligence supports transparency in automated threat detection and fosters greater public trust in these systems [38], [52], [53].
- **Promoting Digital Inclusion and Innovation:** As Android remains a primary channel for digital access in many developing regions, enhancing its security infrastructure plays a vital role in supporting digital inclusion, financial empowerment, and the ongoing adoption of mobile technology [1], [2].

To summarize, the advancements presented in this thesis extend the boundaries of technical research in Android malware detection while generating clear societal benefits. The work contributes to making mobile computing safer, more transparent, and more accessible for users worldwide.

REFERENCES

- [1] StatCounter, "Mobile Operating System Market Share Worldwide," StatCounter Global Stats, Mar. 2024. [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide>
- [2] AV-TEST GmbH, "Malware Statistics and Trends," 2023. [Online]. Available: <https://www.av-test.org/en/statistics/malware/>
- [3] McAfee Labs, "Mobile Threat Report," 2022.
- [4] Y. Zhauniarovich, M. Ahmad, and B. Crispo, "A survey of Android security threats and solutions," *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–39, 2016.
- [5] S. Arshad, M. A. Shah, A. Wahid, and M. A. Ch, "Android malware detection & protection: A survey," *Procedia Computer Science*, vol. 112, pp. 2322–2331, 2017.
- [6] A. Shabtai, Y. Fledel, and Y. Elovici, "Android: A comprehensive security assessment," *IEEE Security & Privacy*, vol. 8, no. 2, pp. 35–44, Mar.–Apr. 2010.
- [7] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: A behavioural malware detection framework for Android devices," *J. Intell. Inf. Syst.*, vol. 38, no. 1, pp. 161–190, Feb. 2012.
- [8] W. Li, Z. Wang, Y. Sun, and X. Luo, "Android malware detection based on machine learning: A systematic review," *IEEE Access*, vol. 8, pp. 124–145, 2020.
- [9] S. Sharma and S. Sahay, "A machine learning-based classification approach for Android malware detection," *Journal of Information Security and Applications*, vol. 48, pp. 102–112, 2019.
- [10] S. Sahay and S. Mehtre, "An overview of Android malware detection techniques," *J. Comput. Virol. Hacking Tech.*, vol. 14, no. 3, pp. 209–219, Sep. 2018.
- [11] H. Gascon, F. Yamaguchi, and K. Rieck, "Structural detection of Android malware using embedded call graphs," in *Proc. ACM Workshop on Artificial Intelligence and Security (AISec)*, 2013, pp. 45–54.
- [12] M. Arp, H. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of Android malware in your pocket," in *Proc. NDSS*, 2014, pp. 1–15.
- [13] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "CopperDroid: Automatic reconstruction of Android malware behaviours," in *Proc. NDSS*, 2015, pp. 1–15.
- [14] T. Bläsing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection," in *Proc. 5th International Conference on Malicious and Unwanted Software (Malware)*, 2010, pp. 55–62.

- [15] F. Martinelli, A. Saracino, and D. Sgandurra, "MADAM: A multi-level anomaly detector for Android malware," in Proc. Int. Conf. on Mathematical Methods, Models and Architectures, 2013, pp. 240–248.
- [16] J. Su, H. Liu, and Y. Pan, "Lightweight hybrid malware detection for Android using deep learning," IEEE Access, vol. 9, pp. 106–115, 2021.
- [17] H. Huang, J. Zhang, S. Wang, and Y. Li, "Deep learning-based Android malware detection with hybrid features," Future Generation Computer Systems, vol. 122, pp. 211–221, 2021.
- [18] M. R. Karim et al., "Detection of Android malware using hybrid features," Secur. Commun. Netw., vol. 2021, pp. 1–11, 2021.
- [19] K. Shahid and M. A. Shah, "A hybrid feature selection framework for Android malware detection," Appl. Soft Comput., vol. 97, pp. 106774, 2020.
- [20] A. R. Tundis, R. Carbone, and S. Peisert, "Detection of Android malware using decision trees and ensemble methods," Digital Investigation, vol. 31, pp. 18–28, 2019.
- [21] K. Liang, R. Du, H. Wang, and B. Fang, "Malware detection using random forest with static analysis," in Proc. Int. Conf. on Cloud Computing and Security, 2018, pp. 115–122.
- [22] S. S. Manogaran and D. Lopez, "A survey of SVM-based Android malware detection," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, pp. 4105–4119, 2021.
- [23] J. Ren, C. Yang, and X. Li, "An improved k-NN approach for Android malware classification," Journal of Information Security and Applications, vol. 54, pp. 102–115, 2020.
- [24] A. V. Kumar and M. S. Turuk, "Android malware detection using Naive Bayes classifier," Procedia Computer Science, vol. 132, pp. 923–929, 2018.
- [25] S. Hou, Y. Ye, and S. Song, "Deep4maldroid: A deep learning framework for Android malware detection based on Linux kernel system call graphs," in Proc. Int. Conf. on Dependable Systems and Networks Workshops (DSN-W), 2016, pp. 188–193.
- [26] J. Chen, Y. Wang, J. Liu, and X. Zhang, "SeqDroid: Detecting Android malware using sequence mining and deep learning," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 3, pp. 1292–1307, 2022.
- [27] M. Sheykhkanloo and M. Naderan, "A CNN–LSTM hybrid model for detecting Android malware," Applied Soft Computing, vol. 106, pp. 107307, 2021.
- [28] D. Wu, C. Mao, and D. Xu, "A dynamic detection approach for Android malware based on incremental ensemble learning," Future Internet, vol. 12, no. 3, pp. 1–14, 2020.
- [29] L. Demetrio et al., "Explaining vulnerabilities of deep learning to adversarial malware binaries," in Proc. USENIX Security Symposium, 2021, pp. 873–890.
- [30] T. Zhang, F. Wang, and J. Ma, "On-device Android malware detection using federated learning and model optimization," IEEE Internet of Things Journal, vol. 10, no. 2, pp. 1223–1235, 2023.

- [31] A. Abusnaina et al., “Adversarial learning attacks and defences in malware detection: A survey,” *Computers & Security*, vol. 115, pp. 102–128, 2022.
- [32] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in *Proc. ACM Workshop on Visualization for Cyber Security (VizSec)*, 2011, pp. 1–7.
- [33] Y. Yuan, X. Pan, and H. Liu, “CNN-LSTM-based malware detection model with multimodal data fusion,” *Journal of Network and Computer Applications*, vol. 194, pp. 103236, 2021.
- [34] A. Souri, R. Ghasemi Gol, and S. Hosseinpour, “Adversarial attacks and defence techniques in Android malware detection: A survey,” *Expert Systems with Applications*, vol. 187, pp. 115–134, 2022.
- [35] A. Jain, R. Sharma, and P. Verma, “Attention-based CNN for Android Malware Detection Using Opcode Sequences,” *Journal of Cybersecurity and Privacy*, vol. 3, no. 2, pp. 45–60, 2023.
- [36] M. A. Rahman, M. S. Hossain, and M. Al-Amin, “Federated Learning for Privacy-Preserving Android Malware Detection,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 112–125, 2023.
- [37] S. Sengupta and S. Banerjee, “GAN-Augmented Random Forest for Imbalanced Android Malware Detection,” *Computers & Security*, vol. 125, pp. 102936, 2023.
- [38] S. Roy, A. Das, and S. Mukherjee, “Explainable Android Malware Detection Using XGBoost and SHAP,” *Information Systems Frontiers*, vol. 25, no. 3, pp. 789–803, 2023.
- [39] J. Lee and H. Cho, “Online Learning-Based Dynamic Analysis for Android Malware Detection,” *ACM Transactions on Privacy and Security*, vol. 26, no. 4, Article 25, 2023.
- [40] L. Zhao, Y. Wang, and X. Li, “Hybrid CNN-LSTM Model for Android Malware Detection,” *IEEE Access*, vol. 12, pp. 34567–34578, 2024.
- [41] K. Patel and R. Mehta, “LSTM-Attention Model for Dynamic Android Malware Detection,” *Journal of Information Security and Applications*, vol. 68, pp. 103215, 2023.
- [42] F. Alvi, M. A. Khan, and S. Rehman, “LightGBM-Based Static Analysis for Efficient Android Malware Detection,” *Future Generation Computer Systems*, vol. 137, pp. 456–467, 2024.
- [43] N. Ahmed and S. Khan, “Permission-Based Android Malware Detection Using Random Forest,” *International Journal of Information Security*, vol. 22, no. 2, pp. 123–135, 2023.
- [44] A. Kumar and S. Das, “SVM with Kernel Trick for Dynamic Android Malware Detection,” *Computers & Security*, vol. 130, pp. 103456, 2024.
- [45] M. Nasir and A. Hussain, “Hybrid Ensemble Learning for Android Malware Detection,” *Expert Systems with Applications*, vol. 201, pp. 117012, 2023.
- [46] Y. Lin and H. Chen, “Fast Detection of Android Malware Using Logistic Regression on Metadata Features,” *Journal of Systems and Software*, vol. 195, pp. 111234, 2023.

- [47] X. Zhao and Z. Wang, "Boosted Decision Trees for Static Android Malware Detection," *Information and Software Technology*, vol. 150, 106789, 2024.
- [48] R. Thakur and V. Singh, "Thread-Based Behavioural Analysis Using LSTM-CNN for Android Malware Detection," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 987–998, 2023.
- [49] D. Singh and M. Sharma, "Energy-Efficient GRU-Based Hybrid Model for Android Malware Detection," *Sustainable Computing: Informatics and Systems*, vol. 35, 100987, 2024.
- [50] A. A. Alzahrani, M. A. Khan, and S. A. Alzahrani, "Android Malware Detection and Identification Frameworks by Machine Learning: A Comprehensive Review," *Information Security: The Next Decade*, vol. 1, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2772503024000161>
- [51] A. Alhussen, "Advanced Android Malware Detection through Deep Learning Optimization," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14552–14557, Jun. 2024.
- [52] M. Yazdinejad, F. Farivar, and B. Wang, "Explainability in AI-based behavioural malware detection systems," *Computers & Security*, vol. 137, p. 103334, 2024.
- [53] M. S. Alkahtani and T. H. H. Aldhyani, "Explainable Machine Learning for Malware Detection on Android Platform," *Information*, vol. 15, no. 1, 25, 2024.
- [54] A. Dehghantanha and M. Conti, "Machine Learning Aided Android Malware Classification," *Computers & Electrical Engineering*, vol. 104, 107489, 2023.
- [55] Y. Elovici and A. Shabtai, "Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-Art Survey," *Information Security Technical Report*, vol. 28, pp. 35–45, 2023.
- [56] A. Guerra-Manzanares, H. Bahsi, and S. Nömm, "KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization," *Computers & Security*, vol. 110, p. 102399, 2021.

LIST OF PUBLICATIONS AND THEIR PROOFS

1st Paper

Title:

"Supervised Learning Approaches in Android Malware Detection: Survey and Analysis"

Status: Accepted for presentation at the 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2025).



**6th International Conference on
Intelligent Communication Technologies and Virtual Mobile Networks**
17-19, June 2025 - ICICV 2025
<https://icicv.org/conf2025/> | icicv.conf@gmail.com

Letter of Acceptance

Paper id: ICICV- 960

Author(s) Name: Mayank Ashok, Rahul Katarya
Dear Author(s):

It is with great pleasure that we extend our warmest congratulations to you on the acceptance of the paper titled **"Supervised Learning Approaches in Android Malware Detection: Survey and Analysis"** for presentation at the 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, scheduled to be held in Francis Xavier Engineering College, Tirunelveli, India from June 17th to 19th, 2025.

Your submission was subjected to a rigorous review process, and the result that your paper has been selected for inclusion in our conference program. We believe that your contribution will greatly enrich the discussions and knowledge exchange at our event. Your participation will undoubtedly contribute to the success of the 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks.

Once again, congratulations on your acceptance, and we anticipate your valuable contribution to our conference.

Yours Sincerely,



Organising Chair
ICICV 2025

C



₹ 10,500

Paid to Conference Registration Fee

Payment ID: pay_QRvokYGxgZDxy8 

 [Report Page](#)

Want to accept online payments for your business?
Visit razorpay.com to get started!



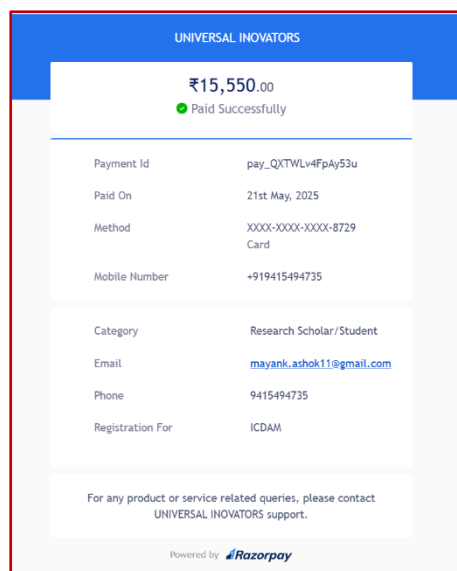
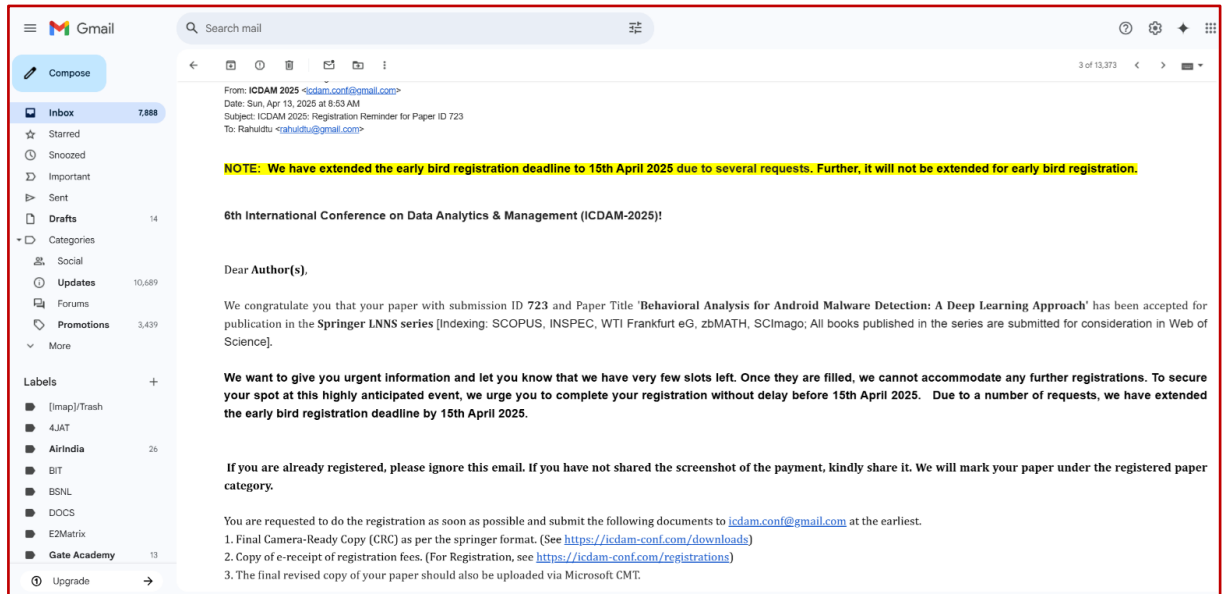
**IEEE
ComSoc**
IEEE Communications Society

2nd Paper

Title:

"Behavioural Analysis for Android Malware Detection: A Deep Learning Approach"

Status: Accepted for presentation at the 6th International Conference on Data Analytics & Management (ICDAM-2025).




3rd Paper

Title:

"Android Malware Detection Framework Using Attention-Based Deep Learning”

Status: Submitted to the IEEE 3rd International Conference on Self Sustainable Artificial Intelligence Systems (WCONF-2025).

2025 3rd World Conference on Communication & Computing : Submission (947) has been created. Inbox x

**Microsoft CMT** <noreply@msr-cmt.org>
to me ▾

Sun 27 Apr, 10:08 ★ ☹ ↶ ⋮

Hello,

The following submission has been created.

Track Name: WCONF2025

Paper ID: 947

Paper Title: Android Malware Detection Framework Using Attention-Based Deep Learning

Abstract:
The rising complexity of Android malware, characterized by polymorphism, runtime evasion, and behavior masking, necessitates more adaptive and interpretable detection mechanisms. This paper presents a comprehensive detection framework that integrates classical ensemble classifiers with an attention-driven deep learning architecture to identify malicious Android applications with high precision. The framework is trained and validated on the Kromdroid dataset, a hybrid corpus encompassing over 140,000 samples sourced from both physical devices and emulator environments. Static and dynamic features are systematically extracted and fused into a unified representation. A CNN-LSTM architecture enhanced with attention mechanisms is employed to learn spatial and temporal behavior patterns, while an Extra Trees ensemble model offers complementary insights with low inference latency. The final classification leverages a confidence-based fusion strategy that combines the strengths of both models. The proposed system demonstrates an accuracy of 99.9%, alongside significantly reduced false positive and false negative rates, highlighting its potential for real-world deployment. By balancing predictive performance with architectural efficiency, the framework provides a scalable foundation for future mobile malware defense systems.

Created on: Sun, 27 Apr 2025 04:38:10 GMT

Last Modified: Sun, 27 Apr 2025 04:38:10 GMT

Authors:

- mayank_ashok11@gmail.com (Primary)
- rahu10t@gmail.com

Secondary Subject Areas: Not Entered

Submission Files:

Android Malware Detection Framework Using Attention-Based Deep Learning.pdf (396 Kb, Sun, 27 Apr 2025 04:38:03 GMT)

Submission Questions Response: Not Entered

Thanks,
CMT team.

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis : **Android Malware Detection Framework Using Attention-Based Deep Learning**

Total Pages : **52**

Name of the Scholar : **Mayank Ashok**

Supervisor : **Dr. Rahul Katarya**

Department : **Dept. of Computer Science & Engineering**

This is to report that the above thesis was scanned for similarity detection. Process and outcome are given below:

Software used: **Turnitin** Similarity Index: **7%** Total Word Count: **16918**

Place: Delhi

Date:

Candidate's Signature

Signature of Supervisor

2. Mayank_Ashok_Thesis.pdf

 Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:97943204

Submission Date

May 27, 2025, 11:53 AM GMT+5:30

Download Date

May 27, 2025, 11:55 AM GMT+5:30

File Name

2. Mayank_Ashok_Thesis.pdf

File Size

1.1 MB

52 Pages

16,918 Words

108,817 Characters





7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

-  **90** Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks
-  **0** Missing Quotations 0%
Matches that are still very similar to source material
-  **0** Missing Citation 0%
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 4%  Internet sources
- 2%  Publications
- 5%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 90 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 4% Internet sources
- 2% Publications
- 5% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Colorado Technical University Online on 2025-03-03	<1%
2	Internet	www.mdpi.com	<1%
3	Publication	Shashi Rathore, Parul Sahare, Mayur Parate, Nikhil Agrawal, Tausif Diwan, Moha...	<1%
4	Submitted works	University of Ulster on 2025-05-07	<1%
5	Internet	www.medrxiv.org	<1%
6	Internet	arxiv.org	<1%
7	Internet	export.arxiv.org	<1%
8	Publication	Saleh Albahli. "Predictive Analytics for Diabetic Patient Care: Leveraging AI to For...	<1%
9	Submitted works	Tilburg University on 2025-05-15	<1%
10	Internet	pmc.ncbi.nlm.nih.gov	<1%

11	Submitted works	universiteknologimara on 2025-05-15	<1%
12	Publication	Elshan Baghirov. "Comprehensive Framework for Malware Detection: Leveraging ...	<1%
13	Submitted works	Napier University on 2025-04-30	<1%
14	Submitted works	The University of the West of Scotland on 2023-04-21	<1%
15	Submitted works	Tilburg University on 2025-05-17	<1%
16	Internet	easychair.org	<1%
17	Submitted works	Queen's University of Belfast on 2024-09-12	<1%
18	Publication	"Securing the Connected World", Springer Science and Business Media LLC, 2025	<1%
19	Submitted works	University of Greenwich on 2024-09-06	<1%
20	Internet	www.restack.io	<1%
21	Submitted works	Abu Dhabi University on 2025-02-14	<1%
22	Submitted works	De Montfort University on 2023-05-12	<1%
23	Publication	Evan Valenti. "Forecasting Shark Attack Risk Using AI: A Deep Learning Approach...	<1%
24	Submitted works	University of Glasgow on 2025-02-14	<1%

25	Submitted works	University of Hertfordshire on 2024-12-01	<1%
26	Internet	link.springer.com	<1%
27	Internet	medium.com	<1%
28	Internet	ir.cuea.edu	<1%
29	Internet	peerj.com	<1%
30	Internet	researchspace.ukzn.ac.za	<1%
31	Submitted works	Liverpool John Moores University on 2020-08-10	<1%
32	Submitted works	Monash University on 2023-10-12	<1%
33	Submitted works	Swinburne University of Technology on 2024-11-02	<1%
34	Submitted works	The University of the West of Scotland on 2024-03-27	<1%
35	Submitted works	University of Surrey on 2023-05-15	<1%
36	Submitted works	Whitecliffe College of Art & Design on 2024-11-28	<1%
37	Internet	dspace.lboro.ac.uk	<1%
38	Internet	www.frontiersin.org	<1%

39	Publication	Ahmad Braydi, Pascal Fossat, Mohsen Ardabilian, Olivier Bareille. "Innovative pro...	<1%
40	Submitted works	Edge Hill University on 2024-04-29	<1%
41	Submitted works	Liverpool John Moores University on 2022-06-06	<1%
42	Submitted works	Napier University on 2021-05-03	<1%
43	Submitted works	Swinburne University of Technology on 2024-11-02	<1%
44	Submitted works	University of Greenwich on 2025-04-22	<1%
45	Submitted works	University of Hertfordshire on 2024-12-02	<1%
46	Internet	cris.brighton.ac.uk	<1%
47	Internet	enac.hal.science	<1%
48	Internet	publications.aston.ac.uk	<1%
49	Internet	repository.tudelft.nl	<1%
50	Internet	ttu-ir.tdl.org	<1%
51	Publication	"Cybersecurity and Human Capabilities Through Symbiotic Artificial Intelligence",...	<1%
52	Submitted works	Dokuz Eylul Universitesi on 2018-08-16	<1%

53	Submitted works	Manipal University Jaipur Online on 2024-08-04	<1%
54	Publication	Solichin, Muhammad Afrinal. "A Forecast Model for Classification for Cost Compo...	<1%
55	Submitted works	Staffordshire University on 2024-05-12	<1%
56	Submitted works	The University of the West of Scotland on 2023-08-21	<1%
57	Submitted works	UNITEC Institute of Technology on 2024-04-02	<1%
58	Submitted works	University of Bradford on 2023-02-01	<1%
59	Submitted works	University of Nottingham on 2024-09-03	<1%
60	Submitted works	Uttar Pradesh Technical University on 2019-01-21	<1%
61	Internet	escholarship.org	<1%
62	Internet	gala.gre.ac.uk	<1%
63	Internet	ijece.iaescore.com	<1%
64	Internet	ijetrm.com	<1%
65	Internet	mafiadoc.com	<1%
66	Internet	researchsystem.canberra.edu.au	<1%

67	Internet	studenttheses.uu.nl	<1%
68	Internet	www.mcafee.com	<1%
69	Internet	www.techscience.com	<1%