# AN ENHANCED HYBRID ARCHITECTURE FOR DETECTING MALWARE IN ANDROID APPS USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES

**Monika Sharma, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir**
**Ajay Kaul, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir**
**Naveen Kumar Gondhi, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir**

## ABSTRACT

*Smartphones are multipurpose gadgets with a wide range of capabilities and constant communication. However, their adaptability, also leaves them open to viruses, endangering user security, privacy, and their financial security. The identification and classification of Android malware has consequently emerged as a critical area of cybersecurity research interest. A rise in malicious assaults on the Android platform has been caused by the high demand for the Android operating system, which has gained malware developers' attention. By stealing sensitive data and degrading the efficiency of devices, such attacks may seriously harm the user. This research paper examines the challenges of identifying Android malware. This study aims to identify malicious and benign files from large datasets using machine learning (ML) and deep learning (DL) techniques to develop efficient, accurate, and robust models for malware detection. We propose a novel ADAX-NETBoost approach, that outperforms existing classification methods with an impressive detection accuracy of 99.34% and 99.21% on Android Malgenome and Drebin dataset, respectively. The experimental results validate the effectiveness of our proposed approach in accurately detecting Android malware, outperforming earlier studies.*

**Keywords:** Hybrid Ensemble Learning Models, Classification Model, Permissions, API Calls, Android Malware**.**

## INTRODUCTION

The widespread use of mobile devices, especially Android smartphones, has markedly increased the risks of malware attacks. The most popular mobile operating system globally, Android, has become a prime target for cybercriminals (Kaur et al., 2024). Mobile applications are increasingly integrated into daily activities, providing users with an extensive range of services (Karbab et al., 2018). According to data provided by Statista (O'Dea, n.d.), there were 7.8 billion users of smartphones worldwide in 2023, with 72% of those being Android users. As of 2021, the global mobile user base was recorded at 7.1 billion individuals. Projections suggest this number will rise to 7.26 billion by 2022 and further expand to 7.49 billion by 2025, highlighting the continuous growth in mobile device adoption worldwide as illustrated in Figure 1 (Statista, 2023).

**Citation Information:** Sharma, M., Kaul, A., & Kumar Gondhi, N. (2025). An enhanced hybrid architecture for detecting malware in android apps using machine learning and deep learning techniques. *Academy of Marketing Studies Journal, 29*(3), 1-21.

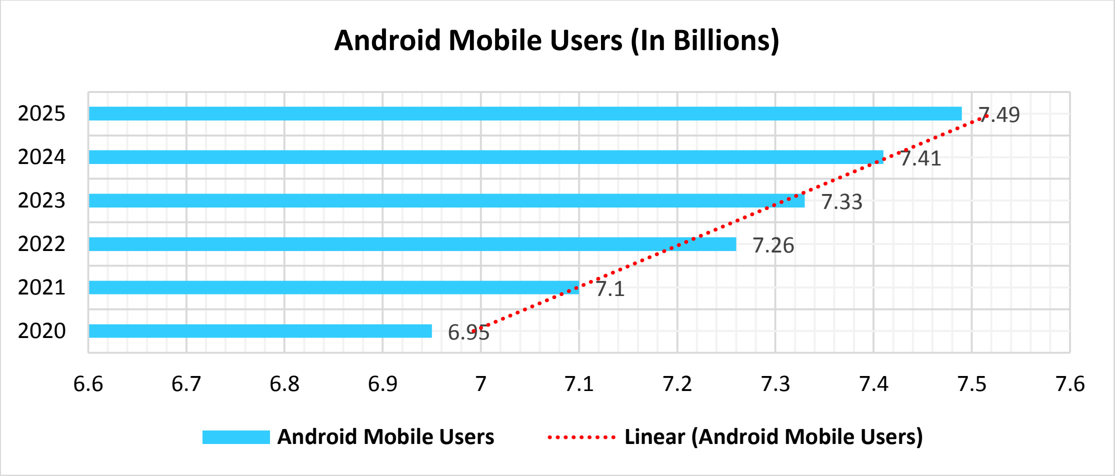


**FIGURE 1**
**PROJECTED GLOBAL MOBILE USER GROWTH FROM 2020 TO 2025**

Applications for mobile devices are often included with the operating system and can be downloaded directly from application stores like Google Play (Google Inc., n.d.) for Android and AppStore (Apple Inc., n.d.) for iOS. Before being published, these apps' security is carefully reviewed and tested. However, in other areas, these application centers might not be easily reachable or might not provide particular applications. In such cases, users may download installation packages from the internet (Wang et al., 2018). Installing applications from unknown sources exposes mobile devices to a significant risk. It can enable the covert installation of malware alongside the desired application or conceal malicious functionalities within seemingly benign applications, which can then covertly compromise user data (Liu et al., 2023). Malware frequently demonstrates one or more of these traits: forced installation, browser hijacking, unauthorized data acquisition and alteration, malicious user information collection, malicious installation, harmful bundling, and other behaviors that cause similar harm. Such activities may violate users' lawful rights and cause significant interest losses (Pan et al., 2020). The rise in malware installation packages for smartphones over the past eight years is summarized in Figure 2.

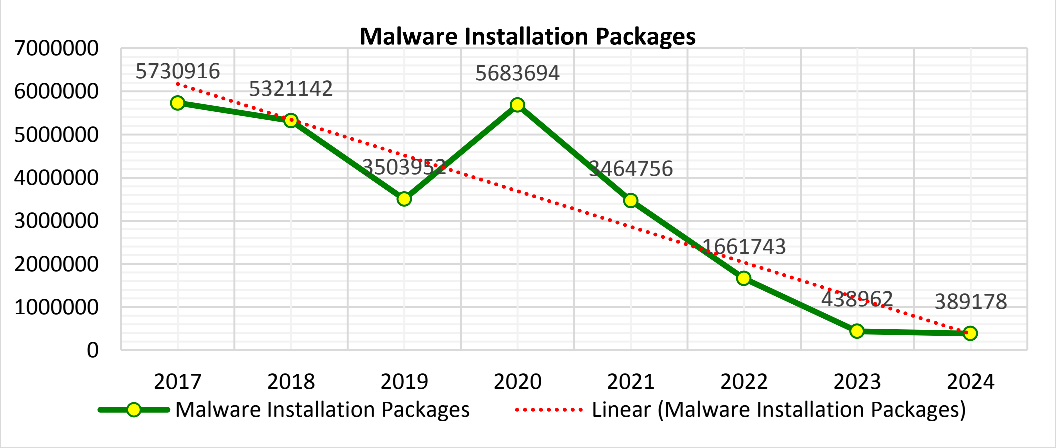


**FIGURE 2**
**ANDROID SMARTPHONE MALWARE INSTALLATION PACKAGES**

Android applications can be examined using static analysis and dynamic analysis, respectively. The reverse engineering technique is used in static analysis to extract program features such as permissions, APIs (Application Program Interfaces), and services (Qin et al., 2019). This approach to malware detection entails examining the source code of an application or verifying file permissions before the installation process (Talal et al., 2019). In contrast, dynamic analysis examines malicious software's behavior by examining its running code (Aslan & Samet, 2020). Dynamic analysis methods observe an application's behavior while running, gather data, including logs and network traffic, access confidential information, and use restricted API calls to find potentially malicious behaviors that static analysis may miss. Dynamic analysis can identify inappropriate behavior and provide insight into the program's behavior by analyzing the data (Wang et al., 2020).

Numerous research papers have focused on detecting and evaluating Android malware by applying static and dynamic analysis methods (Su et al., 2016). Static analysis is currently regarded as a critical method for finding Android malware. Author (Jung et al., 2021) conducted feature selection applying Gini significance and domain expertise. They applied API requests and application permissions concurrently during the creation of the feature vectors. Following feature selection, 987 API requests and 79 permissions were utilized in the experiments. Upon evaluating the values derived from the grid search, it was revealed that the optimal result was achieved with 405 API calls and 25 permissions. In their studies, they achieved a classification accuracy of 96.51% utilizing Random Forest (RF). This paper (Şahin, Durmuş Özkan, Oğuz Emre Kural, Sedat Akleylek, 2021) proposes a revolutionary Android malware detection system utilizing filter-based feature selection approaches. The suggested methodology involves static detection of Android malware utilizing machine learning techniques. Application file permissions are utilized as features in the developed system. Dimension reduction is performed using eight distinct feature selection strategies to improve the runtime and efficacy of machine learning algorithms. This paper (Dabas et al., 2023) presents an innovative malware detection methodology for the Windows platform, utilizing API calls, feature selection, and machine learning techniques. It gathers information regarding API calls in three formats: usage, frequency, and sequences, to generate three distinct feature sets. The feature sets are enhanced using the TF-IDF technique and merged to produce a more comprehensive and resilient feature set, coupled with an API. The results of a number of tests revealed that the API integrated feature set performed better than previous feature sets, achieving 99.6% accuracy and above for all machine learning methods.

## Android Malware

Mobile malware refers to malicious software designed to target mobile operating systems, applications, and sensitive data stored on smartphones. Attackers often distribute these malicious apps through official app stores, third-party platforms, or by leveraging social engineering techniques to gain unauthorized access and exploit root privileges without user approval (Alzubaidi, 2021). These apps use various strategies to accomplish their attacking objectives, shown below:

**Spyware:** It applies to any software that tracks, gathers, and transmits private data to outside parties without the user's awareness or approval (Cinar & Kara, 2023).

**Worms:** Worms do not require attachment to pre-existing files or programs for propagation. They are especially good at rapid proliferation since they can execute and spread autonomously (Smmarwar et al., 2024a).

**Trojan:** Trojan horses are a type of malware that poses as a harmless application in order to trick users into downloading and installing it. This kind of software allows hackers to remotely access files, erase and alter data, build new virus versions, and track user activity by looking at their screen and logs, among other things (Qamar et al., 2019).

**Botnet:** This malicious software opens a system vulnerability and waits for commands by text message or a remote source (Sharma & Kaul, 2024).

**Ransomware:** It is software employed by attackers to encrypt the files of their victims. It indicates that the password is transmitted for a certain cost (Cinar & Kara, 2023).

**Adware:** Adware is a category of malicious software that compromises user privacy and security. This malware compromises the user by capturing screenshots, exfiltrating data, transmitting it to a remote server, or forcibly presenting adverts in the notification bar (Bayazit et al., 2023).

**Riskware:** Riskware malware functions as a normal program while being installed as malicious software on the user's device. It can alter the phone's settings and subsequently send the user to a malicious advertisement page. The attackers want to steal personal information and alter network settings (Nawshin et al., 2024).

## Contributions

In this research, we suggested an entirely novel ADAX-NETBoost method for identifying malicious and benign samples from massive datasets. This research brings forth the following principal contributions.

1. This study thoroughly analyzes architecture of Android, mechanisms of malware attacks, Android malware analysis and classification models. And, also examines prior literature for detecting malware.
2. Proposed a novel ADAX-NETBoost model to identify malicious and benign samples with better accuracy outcome. Evaluate the effectiveness of our proposed ADAX-NETBoost model on two extensively acknowledged and benchmarked datasets, conducting a thorough analysis of its results. Furthermore, we compared our proposed model with other classification models to gauge its performance relative to alternative approaches.
3. Subsequently, evaluate the accuracy of our proposed method in comparison to state-of-art models previously published in the existing literature.

The subsequent sections of the research are organized as follows: Section 2 addresses the background, encompassing Android architecture, Android malware analysis, and classification techniques. The literature study is covered in Section 3, and the problem statement is thoroughly explained in Section 4. The experimental methodology is detailed in section 5. Section 6 presents the experimental setting and results. Finally, the discussion, conclusion and future scope are addressed in Sections 7 and Section 8 respectively.

## Background: Android Architecture

Android is a smartphone-oriented software stack that operates within a sandboxed environment for application execution, as illustrated by Android architecture in Figure 3. The hardware of the phone is interfaced with by a customized embedded Linux system which builds up the supporting infrastructure (Liu et al., 2020).

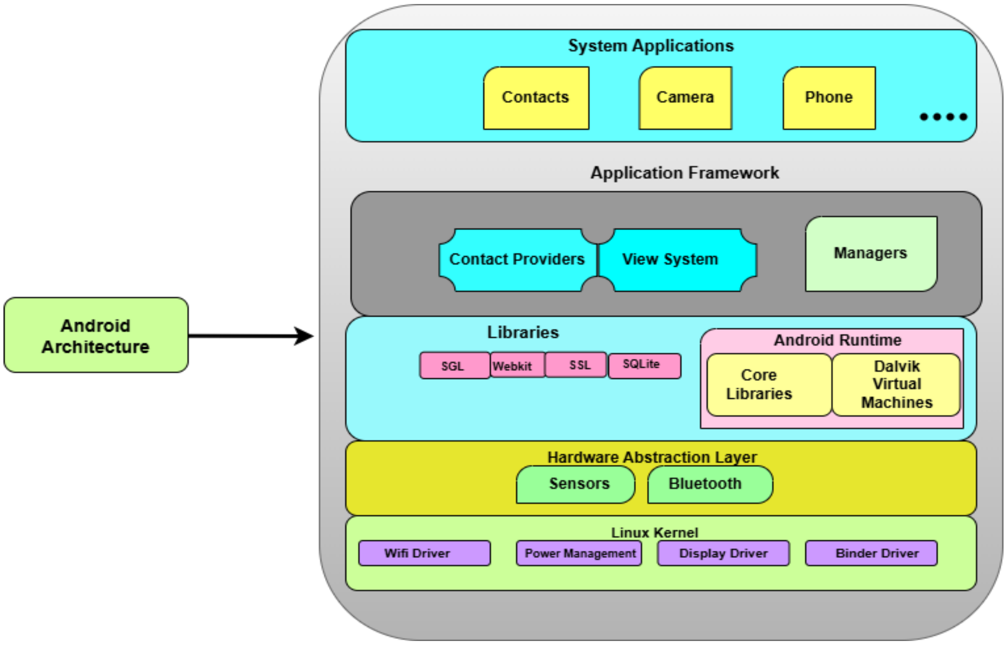


**FIGURE 3**
**ARCHITECTURE OF ANDROID**

The Linux operating system's middleware and application APIs are the only means of communication between programs and phones (Isohara et al., 2011). To gain a deeper understanding of the feature extraction process, this section provides a brief overview of the structure of an Android application. An Android Application Package (APK), typically obtained from the App Store, comprises components such as the AndroidManifest.xml file, executable code, class resources, and a Dalvik bytecode file (Haq et al., 2022). The vital components of an Android application, encompassing content providers, services, broadcast receivers, activities, and implementation classes, are defined within the Android Manifest file. This file serves as the primary configuration file for the application. The Android manifest file facilitates communication with other programs by containing API authorization permissions. Moreover, intent filters are identified in the Manifest file.

## Mechanisms of Malware Attacks

Malware installation on a victim's device is the hacker's only objective. Direct attacks are hard to execute since most systems are shielded by a security system. As a result, hackers attempt to mislead the system into executing the malicious code. Using documents or executable files is the most popular method for accomplishing this. For example, a hacker might show the victim a malicious ad that contains a malicious document attachment or a hyperlink (URL) to the harmful document's website. Embedded scripts or exploits begin to download or extract new malware immediately as the victim opens the document or click on ad as shown in Figure 4. The actual software, which resembles a backdoor or ransom ware that the hacker intends to install on the victim's device. Malicious documents, on the other hand, are typically not the last piece of malware to enter a system during an attack; rather, they are one of the compromised channels the hacker uses to get access.

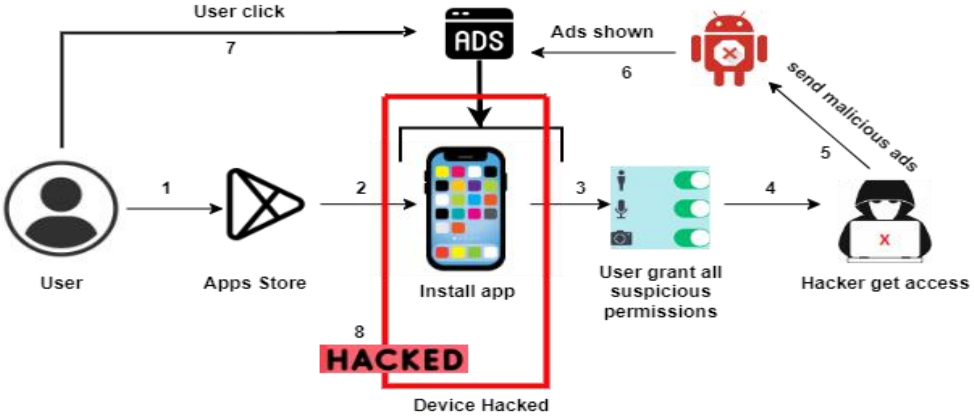


**FIGURE 4**
**PROCESS OF MALWARE ATTACKING ON MOBILE DEVICES**

## Android Malware Analysis

Malware analysis processes are the actions performed to identify malware. There are three approaches for malware analysis: static, dynamic, and hybrid. Static analysis is the process of examining the source code of a program without running it to detect any malicious code. Dynamic analysis involves executing the application on an appropriate test environment and monitoring it to identify the presence of malware. The hybrid analysis is obtained using static and dynamic analysis (Muttoo & Badhani, 2021).

1. **Static Analysis:** Static analysis is the most preferred and desired technique among researchers due to its rapid deployment, swift implementation, and, to a certain degree, its efficacy. This method examines the application's source code without executing it on a physical device or emulator (Bakour et al., 2019). Android apps are published to the app store using a file format known as an APK. APK files are decompiled with APKtool, which decodes the AndroidManifest.xml file to retrieve package names, environment details, intent features, permissions and components (Z. Wang et al., 2020).
2. **Dynamic Analysis:** This technique evaluates the performance of installed programs on smartphones to determine their safety or potential harm. Various challenges, including anomaly- or behavior-based detection methods for examining smartphone activity post-application installation, have been investigated in literature on dynamic analysis. These methodologies evaluate smartphone behavior to ascertain the maliciousness of an installed application (Talal et al., 2019).
3. **Hybrid Analysis:** To overcome the shortcomings of static and dynamic analysis, researchers started merging the two, a technique referred to as the hybrid approach (Kumar et al., 2019). The source code of the programs is regularly reverse-engineered to retrieve static features. The programs are then performed to retrieve dynamic properties in a separate system, like an emulator or an actual device. This analysis strategy is considered the most extensive and insightful one against its implementation and overhead challenges (Bakour et al., 2019).

## Classification Model

In this research work, we evaluate the five classification algorithms, i.e., Random Forest, Support Vector Machine (SVM), Artificial Neural Network (ANN), Adaboost, and XGBoost, with our novel proposed model on Android Malgenome and Drebin dataset. Table 1 shows a detailed description of classification algorithms.

**Table 1**
**A DETAILED TABULAR REPRESENTATION OF CLASSIFICATION ALGORITHMS**

| Ref. | ML/DL Classification Algorithms | Description | Advantages |
|---|---|---|---|
| (Yerima & Sezer, 2019)(W. Wang et al., 2018)<br>(Muzaffar et al., 2023)(Zhu et al., 2018)<br>(Xuan, 2021) | Random Forest | Random Forest is an ensemble learning system that generates predictions by combining different decision trees. Bootstrapping aggregation and random feature subsets generate an extensive collection of decision trees. | • Effectively processes high-dimensional data and accommodates numerous features.<br>• Provides good accuracy and robustness against overfitting.<br>• Can handle missing data and maintain accuracy in the presence of noisy data. |
| (Sheen et al., 2015)(Yerima et al., 2014)<br>(Fatima et al., 2019)(Hou et al., 2017)<br>(Faiz et al., 2021) | Support Vector Machine (SVM) | A supervised machine learning technique known as SVM categorizes data into classes by identifying the optimal hyperplane that maximizes the separation between categories. | • Efficient with small to moderately sized datasets and in high-dimensional spaces.<br>• It can manage nonlinear interactions because of its versatility, which comes from the variety of kernel functions it offers.<br>• Exhibits strong generalization capabilities and resilience to overfitting. |
| (Mikolov et al., 2013)(Xiao et al., 2017)<br>(Bayazit et al., 2021)(Taheri et al., 2020)<br>(Jiang et al., 2018)(Amin et al., 2022) | Artificial Neural Network (ANN) | The artificial neural network (ANN), made up of interconnected nodes (neurons) arranged in layers, was inspired by the structure and function of the human brain. | • Can model complex relationships and learn from large amounts of data.<br>• Robust against noise and can handle missing data.<br>• The flexible architecture allows for various network configurations. |
| (Ünver & Bakour, 2020)(Razak et al., 2018) | AdaBoost | A boosting technique called AdaBoost sequentially emphasizes the misclassified examples to combine numerous weak learners into a robust classifier. | • Achieves high accuracy by focusing on complex samples through iterative training.<br>• Can handle challenges involving binary and multiple classes in classification. |
| (Ling et al., 2019)(Daniel Arp1, Michael Spreitzenbarth2, Malte H¨ubner1, Hugo Gascon1, 2014)<br>(J. Wang et al., 2017)(Chen & Guestrin, 2016) | XGBoost | XGBoost is a gradient-boosting technique that employs an ensemble of weak predictive models with gradient descent optimization to get precise predictions. | • Fast and scalable due to its parallel computing capabilities.<br>• Handles missing values effectively.<br>• Provides built-in regularization techniques to prevent overfitting. |

## LITERATURE SURVEY

The authors provide a thorough analysis of the research on methods for detecting malware. Their research on malware detection is divided into three categories: reviews of feature selection (FS) techniques, reviews of machine learning (ML)-based techniques, and reviews of DL-based techniques. This article presents the authors' (Tyagi & Gautam, 2024) investigation and findings, which involve the evaluation of various machine learning models using 123,453 applications and 5,560 malware samples from the DREBIN dataset to categorize applications as malicious or benign. Authors applied SMOTE, an effective machine learning technique for addressing imbalance concerns, and implemented five distinct machine learning algorithms. The findings indicate that, among all techniques, the Sequential Neural Network (NN) model surpassed others, including Decision Tree (DT), Support Vector Machines (SVM) with various kernels (linear, polynomial, RBF), K-Nearest Neighbors (KNN), and Logistic Regression (LR), achieving an accuracy of 99%. The research introduces an innovative approach for developing a machine learning-based detection model applying freely accessible metadata. This work aims to develop a dependable model for classifying programs as malicious (1) or benign (0) based on the permissions they request, with a focus on Android malware detection. The primary tasks include a comprehensive analysis of Android permissions and metadata as essential indicators for identifying malware, alongside the creation and evaluation of a machine-learning model applying freely available metadata.

The author (Bhat et al., 2023) suggested approach focuses on malware behavioral analysis, which necessitates reconstructing Android malware's behavior. Using a feature selection strategy, redundant features are eliminated for effective malware detection and categorization. With an accuracy rating of 98.08%, the stacking method produces the best categorization results. The authors (Muzaffar et al., 2023) reimplemented 16 representative earlier studies and evaluated them utilizing an organized, relevant, and current dataset including 124,000 Android applications. They also performed new trials to fill in information gaps, and they used the results to figure out the best methods and features for malware detection on Android in modern environments. They found that employing only static features, accuracy of up to 96.8% may be attained. The author (Yizheng et al. 2023) proposes a unique hierarchical contrastive learning method and an extra sample selection strategy to continuously train the Android malware classifier. Their strategy reduces the false positive rate from 0.86% to 0.48% and the false negative rate from 14% to 9%. Authors (Gómez & Muñoz, 2023) employed the most recognized useful static features from relevant literature to train various machine learning models including a multilayer perceptron model from deep learning, for the classification and detection of certain Android malware families. Using more than 18,000 samples gathered from the CICMalDroid2020, CICMalDroid2017, and CICAndMal2017 datasets to achieving a 98.9% F1 score on MLP.

In a 2022 study, Guerra-Manzanares et al. (Guerra-Manzanares et al., 2022) addressed the problem of drifting concepts in malware detection for Android. To overcome this difficulty, they developed a system call-based approach for detecting and characterizing fraudulent Android apps. Their methodology entails real-time surveillance and examination of the sequence of system calls performed by apps. The research used the Multi machine learning Pool for classification on a testing set spanning seven years and the real-device subset of the KronoDroid dataset. A novel strategy to defend against zero-day assaults was put out by the author Millar et al.(Millar et al., 2021) employing a multi-view deep learning technique. They extracted feature vectors from the dataset's application's opcodes, permissions, and API calls. They trained an opcode feature vector-based CNN, a permission feature vector-based fully connected neural network, and an API call feature vector-based CNN. The outcomes from these three networks were used to train a fully connected neural network to categorize. The feature vector size was reported to be 23,226 by the authors, who did not employ preprocessing methods like feature selection throughout the feature extraction steps. In their testing, an F1-score of 0.9927 was obtained for the Drebin dataset, and an F1-score of 0.9963 was obtained for the AMD dataset.

To find advanced Android malware, the author (Zhang et al., 2020) created a framework called APIGraph. The framework uses the official Android API documentation to categorize comparable API requests into clusters. The authors (Faiz et al., 2021) present two novel multi-stage classification models for identifying malware in Android. The first model combines logistic regression and linear Support Vector Machine (SVM), while the second model, as suggested by the authors, combines K-means clustering, logistic regression, and linear SVM. There are two phases in each of the two multi-stage classification techniques for identifying Android malware that has been presented. The classification model performs training in the initial phase, followed by the application of a decision function to classify an application as benign or malicious. Our models can identify both malicious Android applications and colluding combinations of applications. A suggested malware classification methodology is presented in this research paper (Ünver & Bakour, 2020) for identifying malicious samples in the Android platform. The approach demonstrated forward in this study uses grayscale images gathered from Android application source files as an initial basis for spotting malware in the Android ecosystem. The suggested method was trained on datasets of collected grayscale images utilizing four local and three global feature types. The grayscale image dataset's local and global characteristics were employed to train multiple machine learning classification techniques, encompassing Random Forest, K-Nearest Neighbors, Decision Tree, Bagging, Gradient Boost, and AdaBoost. The proposed approach has an excellent classification accuracy of 98.75% and requires only 0.018 seconds of computation time for each sample.

## Problem Statement

Android malware presents a considerable risk to the security and privacy of mobile device users. With the rising popularity of Android smartphones, the possibility of encountering harmful applications and malware-infected software increases. Android malware detection and mitigation have become essential to enable the secure use of mobile devices. According to earlier studies, the technologies used for identifying Android malware exhibit a significant challenge. This research aims to develop an effective and accurate detection method to detect Android malware. The suggested methodology will employ machine learning techniques to enhance the precision and reliability of malware identification. Along with comparisons to earlier studies in the field, benchmarked malware datasets such as the Android Malgenome and Drebin datasets will be used to evaluate the effectiveness of the suggested model. The evaluation criteria, comprising accuracy, precision, recall, and F1-score, are highlighted to reduce the occurrence of false positives. The findings of this study may contribute to the advancement of better detection methods, the protection of user privacy and security on Android devices, and the proactive reduction of Android malware threats.

## Methodology

The proposed model introduced a hybrid methodology that integrates the strengths of machine learning (ML) and deep learning (DL) techniques to improve malware detection performance in Android systems. Adaboost, a widely utilised boosting algorithm, is recognised for its capacity to construct a resilient classifier through the repetitive training of weak classifiers. XGBoost is a gradient-boosting technique that proficiently manages complex datasets through the integration of gradient descent optimisation. Artificial Neural networks are widely recognised for their ability to identify intricate patterns and extract significant information from unstructured data. The addition of artificial neural network component enhances the complexity of the hybrid model. To develop a proposed model i.e. ADAX-NETBoost we combine these three ML and DL techniques to enhances the model's robustness and accuracy. The ADAX-NETBoost model enhances the detection of Android malware samples by including neural networks, which effectively identify complex interconnections and hidden patterns inside the malware samples. Figure 5 illustrates the architecture of the hybrid ensemble ADAX-NETBoost Algorithm.

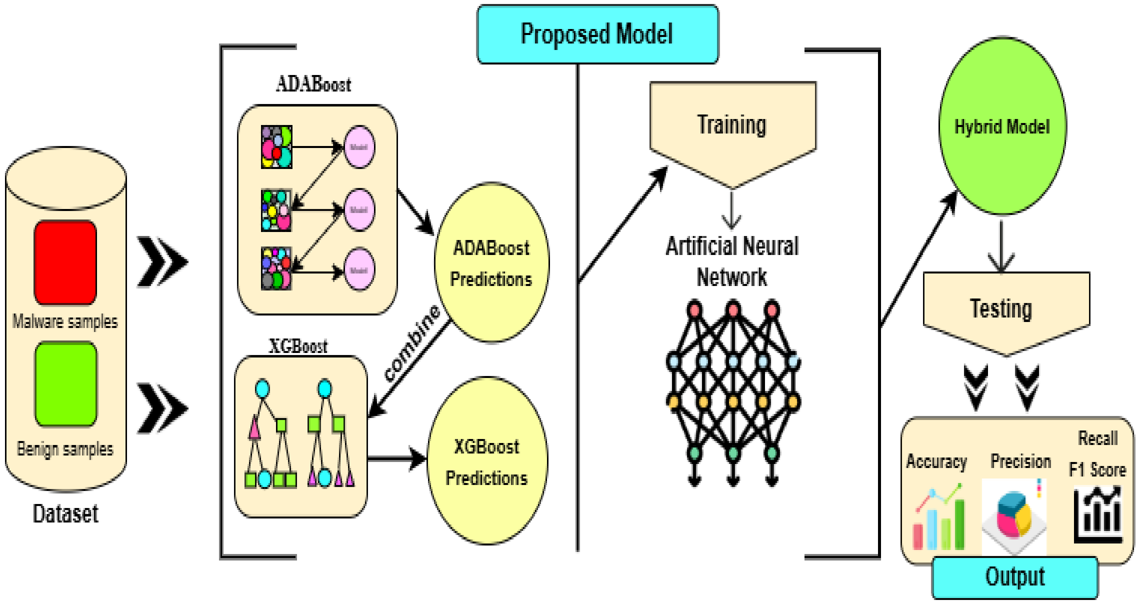


**FIGURE 5**
**ARCHITECTURE OF PROPOSED MODEL**

In the implementation, the dataset is initially processed through AdaBoost, whose predictions are integrated with those of XGBoost to refine classification results. These combined predictions are subsequently passed to ANN, which further enhances the model’s ability to detect complex malware patterns. The hybrid framework, termed ADAX-NETBoost, is evaluated using performance metrics such as accuracy, precision, recall, and F1-score. The results are then compared with those of existing classification models, demonstrating the effectiveness of the proposed approach in identifying Android malware. The essential procedures of the proposed ADAX-NETBoost model are outlined in Table 2.

**Table 2**
**THE BASIC STEPS OF THE ADAX-NETBOOST ALGORITHM**

| | |
|---|---|
| Step 1 | Split the dataset into training and testing subsets. |
| Step 2 | Set the number of estimators, initialise the AdaBoost model with a base estimator, and specify any additional desired hyperparameters. |
| Step 3 | Train an AdaBoost model using the training dataset and provide predictions. |
| Step 4 | Augment the original features with the AdaBoost predictions. |
| Step 5 | Train an XGBoost model on the augmented training data and obtain predictions. |
| Step 6 | Combine the AdaBoost and XGBoost predictions as input for a neural network. |
| Step 7 | Design and train a neural network on the combined predictions and ground truth labels. |
| Step 8 | Evaluate the hybrid model's effectiveness using the testing data and relevant metrics. |

**Citation Information:** Sharma, M., Kaul, A., & Kumar Gondhi, N. (2025). An enhanced hybrid architecture for detecting malware in android apps using machine learning and deep learning techniques. *Academy of Marketing Studies Journal, 29*(3), 1-21.

## Experimental Configuration and Result

This section describes the system configuration, model evaluation metrics and results of the proposed approach. The thorough evaluation shows the excellence of the suggested methodology and its potential to improve Android malware detection systems' accuracy and dependability.

### System Configuration

The system architecture for our research necessitates a minimum of 8-16 GB RAM, a desired Intel Core i5 7th generation or superior CPU, at least 128 GB of storage, High-speed internet and an operating system compatible with Windows. The technology employed in our research encompasses Pandas, Scikit-Learn, Python, Matplotlib, Jupyter Notebook, Google Colab, and fundamental principles of machine learning. These instruments are essential for executing and evaluating the suggested methodology for Android malware detection. Python offers an efficient method for coding, while other modules facilitate data preprocessing, building models, and result visualization.

### Model Evaluation Metrics

i. **Confusion Metrics:** A confusion matrix is a technique used to divide the results of a binary classification task into four categories, as shown in Table 3.

**Table 3**
**CONFUSION METRICS**

| Class | Positive | Negative |
|---|---|---|
| Benign | TN | FP |
| Malware | FN | TP |

(a) True Positive (TP), where the maliciousness of the program is correctly predicted;
(b) False Negative (FN), where the program is incorrectly labeled as benign;
(c) True Negative (TN), where the application is accurately classified as benign; and
(d) False Positive (FP), where the application is mistakenly labeled malicious.

ii. **Accuracy:** The accuracy metric is a heuristic assessment that demonstrates the effectiveness of the categorization models. It illustrates the ratio of accurately predicted cases in the dataset to all other instances. It can be determined by dividing the frequency of events by the count of accurately predicted occurrences (Yizheng et al. 2023).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad \text{(i)}$$

iii. **Recall:** The percentage of malicious applications correctly classified compared to the total malicious applications.

$$\mathrm{Re}\,call = \frac{TP}{TP+FP} \qquad \text{(ii)}$$

In this context, TP represents the count of accurately predicted malware classifications, while FN refers to the number of wrongly identified instances of malware in the dataset.

iv. **Precision:** This ratio shows the ratio of accurately predicted positive instances to the total number of instances that were expected to be positive.

$$Prec = \frac{TP}{TP+FP} \qquad \text{(iii)}$$

v. **F1-score:** This metric is commonly preferred for performance measurements, especially when there is an imbalanced class distribution, as it provides a more practical evaluation of the model's overall performance (Yizheng et al. 2023). By calculating the harmonic mean of precision and recall, the F1 score is a statistic that considers both(Muzaffar et al., 2023).

$$\text{F1\_Score} = \frac{2*\Pr ec*\text{Re}\,call}{\Pr ec + Recall} = \frac{2*TP}{2*TP+FP+FN} \qquad (\text{iv})$$

## Evaluation Result

This research paper's evaluation and comparison section aims to thoroughly evaluate and contrast the performance of the proposed ADAX-NETBoost model compared to classification models, namely random forest, SVM, ANN, AdaBoost and XGBoost. During evaluation two well-established and benchmarked Android malware datasets used: Android Malgenome and Drebin datasets. The Android Malgenome dataset (Zhou & Jiang, 2012) collection comprises diverse Android application files categorized as either malicious or benign. In total, the Malgenome contains 3798 applications of varying types, of which 1260 are classified into 49 distinct malware families. The remaining 2538 applications are regarded as benign. The Drebin dataset (Daniel Arp1, Michael Spreitzenbarth2, Malte H¨ubner1, Hugo Gascon1, 2014) contains numerous Android programs, namely 15,036 app samples, which include 9,476 benign and 5,560 malicious instances, along with 215 features meticulously gathered for Android malware analysis. The incorporation of these varied datasets facilitates a thorough assessment of the models' efficacy across distinct malware and benign applications.

## Performance Evaluation of Proposed Model with State-of-The-Art Methods

This research uses a various classification methods for separate training and testing of the dataset. A variety of classification models were employed for comparative analysis, including Random Forest, SVM, ANN, AdaBoost, XGBoost, and the suggested ADAX-NETBoost model. The ADAX-NETBoost model exhibited outstanding performance on the Android Malgenome and Drebin datasets, attaining accuracy rates of 99.34% and 99.21%, respectively. The suggested model's detection performance is compared to existing classification techniques on the Drebin dataset and the Android Malgenome dataset in Tables 4 & 5, respectively.

**Table 4**
**COMPARISON OF THE DETECTION PERFORMANCE OF THE PROPOSED MODEL WITH EXISTING CLASSIFICATION METHODS ON ANDROID MALGENOME DATASET**

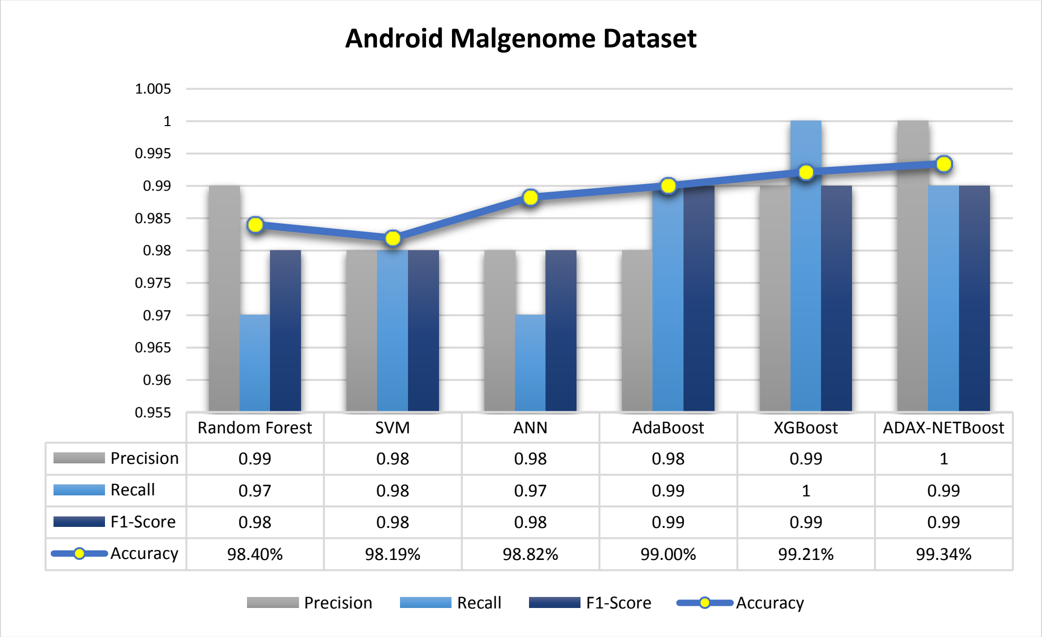
| **Algorithm** | **Android Malgenome Dataset** | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| Random Forest | 98.40% | 0.99 | 0.97 | 0.98 |
| SVM | 98.19% | 0.98 | 0.98 | 0.98 |

| | | | | |
|---|---|---|---|---|
| ANN | 98.82% | 0.98 | 0.97 | 0.98 |
| AdaBoost | 99.00% | 0.98 | 0.99 | 0.99 |
| XGBoost | 99.21% | 0.99 | 1.00 | 0.99 |
| **ADAX-NETBoost** | **99.34%** | **1.00** | **0.99** | **0.99** |

**Table 5**
**COMPARISON OF DETECTION PERFORMANCE OF THE PROPOSED MODEL WITH EXISTING CLASSIFICATION METHODS ON DREBIN DATASET**

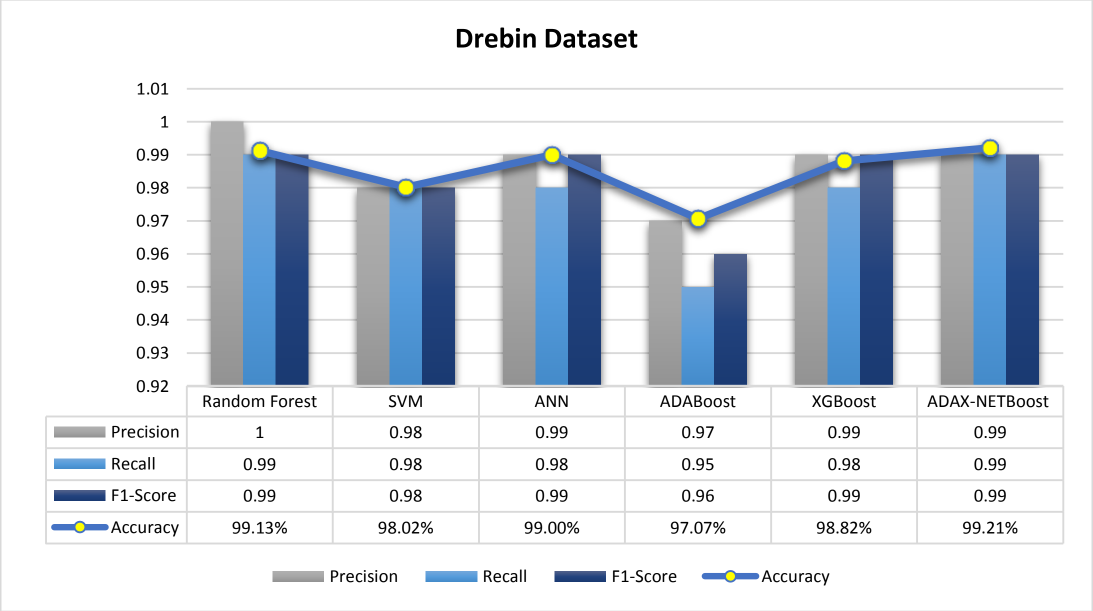
| **Algorithms** | **Drebin Dataset** | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| Random Forest | 99.13% | 1.0 | 0.99 | 0.99 |
| SVM | 98.02% | 0.98 | 0.98 | 0.98 |
| ANN | 99.00% | 0.99 | 0.98 | 0.99 |
| AdaBoost | 97.07% | 0.97 | 0.95 | 0.96 |
| XGBoost | 98.82% | 0.99 | 0.98 | 0.99 |
| **ADAX-NETBoost** | **99.21%** | **0.99** | **0.99** | **0.99** |

The outcomes show that the proposed ADAX-NETBoost model performs better than other classification algorithms on the Android Malgenome and Drebin dataset. Figures 6 and 7 present the performance comparison between the proposed model and existing classification models on the Android Malgenome and Drebin dataset, respectively.



| | Random Forest | SVM | ANN | AdaBoost | XGBoost | ADAX-NETBoost |
|---|---|---|---|---|---|---|
| Precision | 0.99 | 0.98 | 0.98 | 0.98 | 0.99 | 1 |
| Recall | 0.97 | 0.98 | 0.97 | 0.99 | 1 | 0.99 |
| F1-Score | 0.98 | 0.98 | 0.98 | 0.99 | 0.99 | 0.99 |
| Accuracy | 98.40% | 98.19% | 98.82% | 99.00% | 99.21% | 99.34% |

**FIGURE 6**
**EVALUATION OF PERFORMANCE COMPARING THE PROPOSED MODEL WITH THE CLASSIFICATION MODEL USING THE MALGENOME DATASET**



| | Random Forest | SVM | ANN | ADABoost | XGBoost | ADAX-NETBoost |
|---|---|---|---|---|---|---|
| Precision | 1 | 0.98 | 0.99 | 0.97 | 0.99 | 0.99 |
| Recall | 0.99 | 0.98 | 0.98 | 0.95 | 0.98 | 0.99 |
| F1-Score | 0.99 | 0.98 | 0.99 | 0.96 | 0.99 | 0.99 |
| Accuracy | 99.13% | 98.02% | 99.00% | 97.07% | 98.82% | 99.21% |

**FIGURE 7**
**EVALUATION OF PERFORMANCE COMPARING THE PROPOSED MODEL WITH THE CLASSIFICATION MODEL USING THE DREBIN DATASET**

## Comparative Analysis

The effectiveness of identifying Android malware was demonstrated by comparing the performance of our suggested framework against the other cutting-edge detection methods. We conducted a comprehensive examination of related methods that have been employed in the past, including machine learning-based approaches and general classification-based techniques. Table 5 shows the comparative results between proposed model and the past research work.

**Table 6**
**TABULAR COMPARATIVE RESEARCH OUTLINING THEIR PERFORMANCE FINDINGS AND THE PROPOSED MODEL**

| S.No. | Author | Approach | Dataset | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|---|
| 1. | S. Millar et. al., (Millar et al., 2020) | Discriminative Adversarial Network (DAN) | Drebin | 97.30% | 0.98 | 0.96 | 0.97 |
| 2. | T. Kim et. al., (Kim et al., 2019) | Deep Neural Network | Android MalGenome and VirusShare | 98% | 0.98 | 0.99 | 0.99 |
| 3. | H.Fereidooni et.al.(Fereidooni et al., 2016) | ANASTASIA | MODROID, Drebin, Android Malgenome and VirusShare | 97% | 0.97 | 0.97 | 0.97 |
| 4. | X.Qin et al.(Qin et al., 2019) | Deep Belief Network | VirusShare, Android Malgenome, and Androzoo | 98.71% | 0.98 | 0.99 | 0.98 |
| 5. | Karbab et | MalDozer, | Android | 96% | 0.98 | 0.99 | 0.99 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | al.(Karbab et al., 2018) | DNN | Malgenome, Drebin | | | | |
| 6. | A.El Fiky (El Fiky*, 2020) | Deep-Droid | Android Malgenome, Drebin | 98.5% | 0.99 | 0.98 | 0.98 |
| **7.** | **ADAX-NETBoost (Proposed)** | **AdaBoost, XGBoost with ANN** | **Android Malgenome, Drebin** | **99.34% and 99.21%** | **1.0** | **0.99** | **0.99** |

The results presented in Table 6 indicate that our framework excels in terms of both detection accuracy and F-measure values when compared to other methods. The investigation of malware detection using several classification methods that use dynamic and static analytical techniques has been a significant emphasis in many research studies. Artificial intelligence-based malware detection technologies, including machine learning, deep learning, and hybrid approaches, are outperforming traditional methods. Figure 8 represents a comparative graphical analysis of the proposed model with previous literature work.

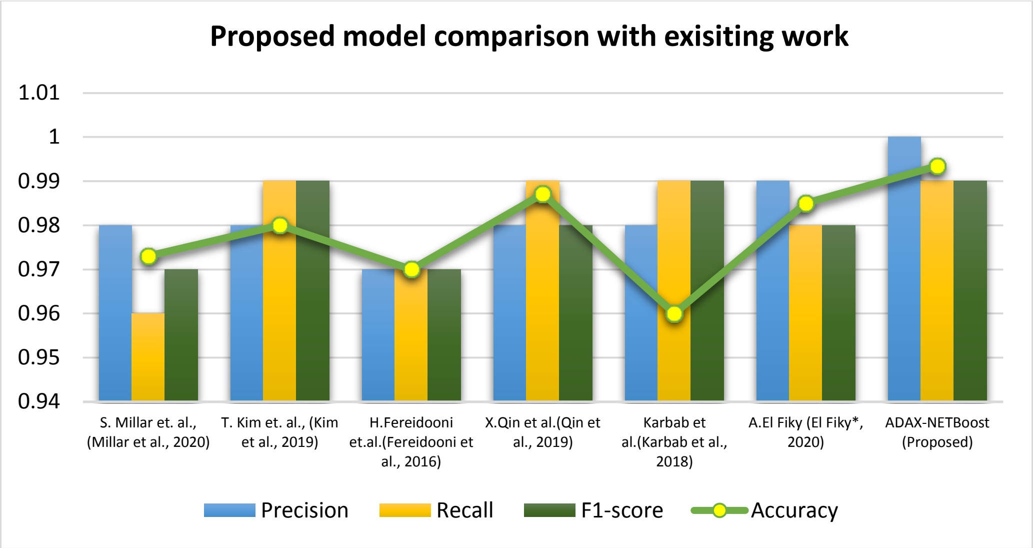


**FIGURE 8**
**COMPARISON BETWEEN THE PROPOSED APPROACH AND PRIOR RESEARCH**

Our research proposes ADAX-NETBoost model, which combines the AdaBoost-XGBoost with ANN classifier to evaluate the Android Malgenome and Drebin dataset. The results of the experiments demonstrate that our method performs exceptionally well compared to the current previous work, with detection accuracy of 99.34% and 99.21%, respectively.

## DISCUSSION

The main purpose of this research was to address the issue of detecting Android malware. To resolve this problem, two significant datasets were used: the Android Malgenome dataset and the Drebin dataset. The purpose of the ADAX-NETBoost model was to improve Android malware detection's precision and reliability. The proposed ADAX-NETBoost model was further evaluated in comparison to other popular classification models, including XGBoost, AdaBoost, support vector machines (SVM), random forest, and artificial neural networks (ANN). As a result, malware detection techniques for mobile devices may become more accurate and reliable. The evaluation considers the performance of each model on the two datasets. According to the results of the Drebin and Android Malgenome datasets, the proposed ADAX-NETBoost model demonstrated enhanced accuracy, achieving exceptional accuracy rates of 97.21% and 99.34%, respectively. Additionally, the effectiveness of the proposed ADAX-NETBoost model was evaluated in comparison to earlier studies in the field of Android malware detection. The purpose of the comparison was to evaluate the extent to which the proposed model improved overall accuracy and reduced false positives. The outcomes showed that the proposed ADAX-NETBoost model successfully distinguished between malicious and benign applications with an excellent accuracy rate of 99.34% on the Android Malgenome dataset and 99.21% on Drebin dataset.

## CONCLUSION AND FUTURE WORK

The market share of the Android operating system, which powers on a significant percentage of these devices, has experienced a stunning increase in response to the tremendous growth in the use of mobile devices. As a result, hackers have started using malware more regularly to target these devices. This paper proposes a novel ADAX-NETBoost approach to identify malicious and benign files from extensive datasets. Additionally, our proposed ADAX-NETBoost approach achieved a remarkable detection accuracy of 99.34% and 99.21%, outperforming six other classification approaches, demonstrating our method's effectiveness. Through comparison with previous studies, we have demonstrated the efficacy of our suggested approach in detecting Android malware. Along with their growing popularity, Android operating systems are becoming the target of more malware attacks, which highlights the need for effective malware detection tools to lessen the risks of such attacks. The issue of Android malware detection has a possible solution in our suggested method. According to the results of this research, it can be said that the proposed approach is a useful one for real-time malware detection on mobile devices. The promising findings of our proposed ADAX-NETBoost approach for Android malware detection indicate that there is potential for further research and improvement in this area. To increase the effectiveness of the detection system, one possible future option is to add more applications and increase the dataset. Furthermore, a more comprehensive analysis can be performed by employing attributes such as API, Opcode, and behavioral features, as well as by combining various dimension reduction techniques. This could lead to more accurate and reliable malware detection techniques for mobile devices. The described approach can also be enhanced to discover different types of malicious software on various platforms. Future studies will examine ways to increase the dataset and include new applications.

### Declarations

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

## REFERENCES

Alzubaidi, A. (2021). Recent Advances in Android Mobile Malware Detection: A Systematic Literature Review. IEEE Access, 9, 146318–146349.

Amin, M., Shah, B., Sharif, A., Ali, T., Kim, K. Il, & Anwar, S. (2022). Android malware detection through generative adversarial networks. *Transactions on Emerging Telecommunications Technologies, 33*(2), 1–29.

Aslan, O., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. IEEE Access, 8, 6249–6271.

Bakour, K., Ünver, H. M., & Ghanem, R. (2019). The Android malware detection systems between hope and reality. In SN Applied Sciences (Vol. 1, Issue 9). Springer International Publishing.

Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2021). Neural Network Based Android Malware Detection with Different IP Coding Methods. 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 1–6.

Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2023). Protecting Android Devices From Malware Attacks: A State-of-the-Art Report of Concepts, Modern Learning Models and Challenges. IEEE Access, 11(September), 123314–123334.

Bhat, P., Behal, S., & Dutta, K. (2023). A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. Computers & Security, 130, 103277.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Vols. 13-17-Augu, pp. 785–794).

Cinar, A. C., & Kara, T. B. (2023). The current state and future of mobile security in the light of the recent mobile security threat reports. *Multimedia Tools and Applications, 82*(13), 20269–20281.

Dabas, N., Ahlawat, P., & Sharma, P. (2023). An Effective Malware Detection Method Using Hybrid Feature Selection and Machine Learning Algorithms. *Arabian Journal for Science and Engineering, 48*(8), 9749–9767.

Daniel Arp1, Michael Spreitzenbarth2, Malte H¨ubner1, Hugo Gascon1, K. R. (2014). Drebin: Effective and explainable detection of Android malware in your pocket. Proc. Netw. Distrib. Syst. Secur. Symp., 14(1), 23_26.

El Fiky*, A. H. (2020). Deep Droid: Deep Learning for Android Malware Detection. *International Journal of Innovative Technology and Exploring Engineering, 9*(12), 122–125.

Faiz, M. F. I., Hussain, M. A., & Marchang, N. (2021). Android Malware Detection Using Multi-stage Classification Models BT - Complex, Intelligent and Software Intensive Systems (L. Barolli, A. Poniszewska-Maranda, & T. Enokido (eds.); pp. 244–254). Springer International Publishing.

Fatima, A., Maurya, R., Dutta, M. K., Burget, R., & Masek, J. (2019). Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning. 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 220–223. https://doi.org/10.1109/TSP.2019.8769039

Fereidooni, H., Conti, M., Yao, D., & Sperduti, A. (2016). ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications. 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 1–5. https://doi.org/10.1109/NTMS.2016.7792435

Gómez, A., & Muñoz, A. (2023). Deep Learning-Based Attack Detection and Classification in Android Devices. In Electronics (Vol. 12, Issue 15).

Guerra-Manzanares, A., Luckner, M., & Bahsi, H. (2022). Android malware concept drift using system calls: Detection, characterization and challenges. Expert Systems with Applications, 206, 117200.

Haq, I. U., Khan, T. A., Akhunzada, A., & Liu, X. (2022). MalDroid: Secure DL-enabled intelligent malware detection framework. IET Communications, 16(10), 1160–1171.

Hou, S., Saas, A., Chen, L., Ye, Y., & Bourlai, T. (2017). Deep Neural Networks for Automatic Android Malware Detection. 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 803–810.

Isohara, T., Takemori, K., & Kubota, A. (2011). Kernel-based behavior analysis for android malware detection. Proceedings - 2011 7th International Conference on Computational Intelligence and Security, CIS 2011, 1011–1015.

Jiang, H., Turki, T., & Wang, J. T. L. (2018). DLGraph: Malware Detection Using Deep Learning and Graph Embedding. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 1029–1033.

Jung, J., Park, J., Cho, S. J., Han, S., Park, M., & Cho, H. H. (2021). Feature engineering and evaluation for android malware detection scheme. *Journal of Internet Technology, 22*(2), 423–440.

Karbab, E. M. B., Debbabi, M., Derhab, A., & Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. DFRWS 2018 EU - Proceedings of the 5th Annual DFRWS Europe, 24, S48–S59.

Kaur, A., Lal, S., Goel, S., Pandey, M., & Agarwal, A. (2024). Android Malware Detection Using Machine Learning. IC3-2024: Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing, 186–191.

Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2019). A multimodal deep learning method for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 773–788.

Kumar, N., Mukhopadhyay, S., Gupta, M., Handa, A., & Shukla, S. K. (2019). Malware classification using early stage behavioral analysis. Proceedings - 2019 14th Asia Joint Conference on Information Security, AsiaJCIS 2019, 16–23.

Ling, J., Wang, X., & Sun, Y. (2019). Research of Android Malware Detection based on ACO Optimized Xgboost Parameters Approach BT - Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019). 364–371.

Liu, C., Lu, J., Feng, W., Du, E., Di, L., & Song, Z. (2023). MOBIPCR: Efficient, accurate, and strict ML-based mobile malware detection. Future Generation Computer Systems, 144, 140–150.

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A Review of Android Malware Detection Approaches Based on Machine Learning. IEEE Access, 8, 124579–124607.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings, 1–12.

Millar, S., McLaughlin, N., Del Rincon, J. M., Miller, P., & Zhao, Z. (2020). DANdroid: A Multi-View Discriminative Adversarial Network for Obfuscated Android Malware Detection. CODASPY 2020 - Proceedings of the 10th ACM Conference on Data and Application Security and Privacy, 353–364.

Millar, S., McLaughlin, N., Martinez del Rincon, J., & Miller, P. (2021). Multi-view deep learning for zero-day Android malware detection. *Journal of Information Security and Applications, 58,* 102718.

Muttoo, S. K., & Badhani, S. (2021). An analysis of malware detection and control through Covid-19 pandemic. Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development, INDIACom 2021, 637–641.

Muzaffar, A., Hassen, H. R., Zantout, H., & Lones, M. A. (2023). A Comprehensive Investigation of Feature and Model Importance in Android Malware Detection. 1–18.

Nawshin, F., Gad, R., Unal, D., Al-Ali, A. K., & Suganthan, P. N. (2024). Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey. Computers and Electrical Engineering, 117, 109233.

O’Dea, S. (n.d.). Forecast number of mobile users worldwide from 2020 to 2025.

Pan, Y., Ge, X., Fang, C., & Fan, Y. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. IEEE Access, 8, 116363–116379.

Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems, 97*, 887–909.

Qin, X., Zeng, F., & Zhang, Y. (2019). MSndroid: The android malware detector based on multi-class features and deep belief network. ACM International Conference Proceeding Series.

Razak, M. F. A., Anuar, N. B., Othman, F., Firdaus, A., Afifi, F., & Salleh, R. (2018). Bio-inspired for Features Optimization and Malware Detection. *Arabian Journal for Science and Engineering, 43*(12), 6963–6979.

Şahin, Durmuş Özkan, Oğuz Emre Kural, Sedat Akleylek, E. K. (2021). A novel Android malware detection system: adaption of filter‑based feature selection methods.pdf. *Journal of Ambient Intelligence and Humanized Computing*.

Sharma, M., & Kaul, A. (2024). A review of detecting malware in android devices based on machine learning techniques. *Expert Systems, 41*(1), 1–25.

Sheen, S., Anitha, R., & Natarajan, V. (2015). Android based malware detection using a multifeature collaborative decision fusion approach. *Neurocomputing, 151*(P2), 905–912.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024a). Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review. Telematics and Informatics Reports, 14(March).

Statista. (2023). Forecast number of mobile users worldwide 2020–2025.

Su, X., Zhang, D., Li, W., & Zhao, K. (2016). A deep learning approach to android malware feature learning and detection. In *2016 IEEE Trustcom/BigDataSE/ISPA* (pp. 244-251). IEEE.

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., & Conti, M. (2020). Similarity-based Android malware detection using Hamming distance of static binary features. Future Generation Computer Systems, 105, 230–247.

Talal, M., Zaidan, A. A., Zaidan, B. B., Albahri, O. S., Alsalem, M. A., Albahri, A. S., Alamoodi, A. H., Kiah, M. L. M., Jumaah, F. M., & Alaa, M. (2019). Comprehensive review and analysis of anti-malware apps for smartphones. In Telecommunication Systems (Vol. 72, Issue 2). Springer US.

Tyagi, T., & Gautam, M. (2024). Machine Learning based Malware Detection in Android: A Practical Approach. 2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC), 1417–1424.

Ünver, H. M., & Bakour, K. (2020). Android malware detection based on image-based features and machine learning techniques. *SN Applied Sciences*, *2*(7), 1299.

Wang, H., Liu, Z., Liang, J., Vallina-Rodriguez, N., Guo, Y., Li, L., Tapiador, J., Cao, J., & Xu, G. (2018). Beyond Google play: A large-scale comparative study of Chinese android app markets. Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, Section 5, 293–307.

Wang, J., Li, B., & Zeng, Y. (2017). XGBoost-Based Android Malware Detection. 2017 13th International Conference on Computational Intelligence and Security (CIS), 268–272.

Wang, Z., Liu, Q., & Chi, Y. (2020). Review of android malware detection based on deep learning. IEEE Access, 8, 181102–181126.

Xiao, X., Wang, Z., Li, Q., Xia, S., & Jiang, Y. (2017). Back-propagation neural network on Markov chains from system call sequences: A new approach for detecting Android malware with system call sequences. *IET Information Security*, *11*(1), 8–15.

Xuan, C. Do. (2021). Detecting Android Malware Based on Analyzing Abnormal Behaviors of APK File. *International Journal of Computer Science and Network Security*, *12*(3), 464–471.

Yerima, S. Y., & Sezer, S. (2019). DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. *IEEE Transactions on Cybernetics*, *49*(2), 453–466.

Yerima, S. Y., Sezer, S., & Muttik, I. (2014). Android Malware Detection Using Parallel Machine Learning Classifiers. 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, 37–42.

Yizheng Chen, Zhoujie Ding, D. W. (2023). Continuous Learning for Android Malware Detection. *Cryptography and Security*, *13*(3), 1484.

Zhang, X., Zhang, Y., Zhong, M., Ding, D., Cao, Y., Zhang, Y., Zhang, M., & Yang, M. (2020). Enhancing State-of-the-art Classifiers with API Semantics to Detect Evolved Android Malware. *Proceedings of the ACM Conference on Computer and Communications Security*, 757–770.

Zhou, Y., & Jiang, X. (2012). Dissecting Android Malware: Characterization and Evolution. *Proceedings - IEEE Symposium on Security and Privacy*, *4*, 95–109.

Zhu, H.-J., Jiang, T.-H., Ma, B., You, Z.-H., Shi, W.-L., & Cheng, L. (2018). HEMD: a highly efficient random forest-based malware detection framework for Android. *Neural Computing and Applications*, *30*(11), 3353–3361.

**Received:** 23-Jan-2025, Manuscript No. AMSJ-25-15652; **Editor assigned:** 24-Jan-2025, PreQC No. AMSJ-25-15652(PQ); **Reviewed:** 19-Feb-2025, QC No. AMSJ-25-15652; **Revised:** 23-Feb-2025, Manuscript No. AMSJ-25-15652(R); **Published:** 17-Mar-2025