The Scientific Temper (2025) Vol. 16 (11): 4987-4997

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.11.02

https://scientifictemper.com/

RESEARCH ARTICLE

Comparative Analysis of Machine Learning Algorithms for Malware Detection in Android Ecosystems

Pallavi M. Shimpi[1*], Nitin N. Pise[2]

Abstract

Android malware is a growing cybersecurity concern as malicious applications exploit vulnerabilities in the Android operating system to steal sensitive data, disrupt device functionality, or gain unauthorised control. The rising sophistication of these threats makes conventional signature-based detection techniques insufficient, highlighting the need for advanced learning-based solutions that adapt to evolving attack patterns. This study proposes a comparative evaluation of Machine Learning (ML) as well as Deep Learning (DL) models for Android malware detection using the RT-IoT2022 dataset, which contains diverse benign and malicious network traffic. Five models were implemented: Random Forest (RF), Support Vector Machine (SVM), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and a hybrid CNN-LSTM. Experimental analysis showed that while RF and SVM achieved strong baseline results and CNN effectively extracted spatial features, LSTM alone struggled to classify balanced classes. The proposed hybrid CNN–LSTM achieved the best results with 99.30% accuracy and 99.76% F1-score. These findings validate the superiority of hybrid architectures and provide a pathway for lightweight, real-time, and adversarial-resistant malware detection systems for Android and Internet of Things (IoT) environments.

Keywords: Random Forest, Malware Detection, Machine Learning, Android Ecosystem, Deep Learning

关键词：随机森林，恶意软件检测，机器学习，安卓生态系统，深度学习

Introduction

The Android revolution has changed how people retrieve information, interact, and perform business activities. Unfortunately, the same Android characteristics that attract users, its openness and popularity, have exposed it to Android cybersecurity problems (Kalsi , 2022). Cybersecurity threats are on the rise as malware aimed at Android systems becomes more sophisticated. Attackers are constantly inventing new and advanced ways of breaching device security, stealing sensitive personal data, compromising device functions, and in some cases, remotely accessing the unit and controlling it (Desani et al., 2021). There is an alarming increase in the number of malware variants attempting to breach Android systems, which calls for increasing mobile security infrastructure (Kaur et al., 2024). As mobile applications continue to be relied on extensively for personal and professional purposes, the secure status of Android devices relates directly to the larger issues of mobile data, privacy, finances, and trust (Parveen et al., 2023; Nguyen et al., 2021).

In such a scenario, Android malware seeks to exploit vulnerabilities, so early identification and proper categorisation are crucial. Proactive threat identification protects vulnerable systems from being exploited, greatly reducing damage while ensuring operational dependability and safeguarding the data on the device (Jung et al., 2019; Bromberg & Gitzinger, 2020). Legacy defences, like signature-based antivirus programs, are outdated and incapable of dealing with obfuscated new or fast-transforming malware (Almobaideen et al., 2025). These frameworks operate on

[1]Research Scholar, Department of Computer Engineering & Technology, Dr. Vishwanath Karad MIT World Peace University MIT Campus, Kothrud, Pune, Maharashtra, 411038, India.

[2] Professor, Department of Computer Engineering & Technology, Dr. Vishwanath Karad MIT World Peace University MIT Campus, Kothrud, Pune, Maharashtra, 411038, India.

***Corresponding Author:** Pallavi M. Shimpi, Research Scholar, Department of Computer Engineering & Technology, Dr. Vishwanath Karad MIT World Peace University MIT Campus, Kothrud, Pune, Maharashtra, 411038, India, E-Mail: mspallavi2710@gmail.com

How to cite this article: Shimpi, P.M., Pise, N.N. (2025). Comparative Analysis of Machine Learning Algorithms for Malware Detection in Android Ecosystems. The Scientific Temper, **16**(11):4987-4997.

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.11.02

Source of support: Nil

Conflict of interest: None.

Received: 11/10/2025 **Accepted:** 05/11/2025 **Published:** 22/11/2025

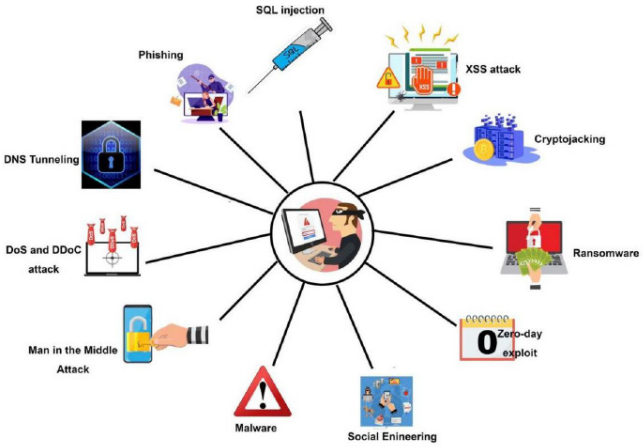


Figure 1: Types of cyberattacks (Akhtar & Feng, 2022a)

the premise of having a database of known threats. They are reactive, devoid of proactive anticipation of impending issues (Abualigah et al., 2021; Alzahrani & Balaji et al., 2023). A primary challenge is identifying novel zero-day attacks or unmapped malware variants that elude detection without extensive documented malware masquerading as benign files. Thus, multi-layered sophisticated Android malware defences need to be able to predict, identify, and disable threats automatically without human aid to provide full protection (Amin et al., 2020; Faris et al., 2020). Figure 1 illustrates different categories of cyberattacks within the digital realm or cyberspace.

ML approaches have arisen as formidable answers to this dire problem. Classic techniques lack such potential, whereas ML systems can learn from extensive data, discern intricate patterns, and draw judgments about unfamiliar circumstances (Ahmed et al., 2025; Alhebsi, 2022). Applying to the detection of Android malware, ML models have the potential to identify benign vs. malware apps autonomously through behaviour patterns, request permissions, network interactions, and code layout examination (Salehin et al., 2024). They can locate subtle abnormalities that human analysts or signature-based tools might overlook (Konwarh, R., & Cho, W. C. (2021); Rathore et al., 2020). Additionally, ML methods enable ongoing refinement; models can be retrained using new samples of malware to keep up with the constantly changing threat environment (Wang et al., 2023). Methods like supervised learning, unsupervised anomaly detection, and DL architectures such as CNN and Recurrent Neural Networks (RNN) have been highly promising in this area (Khan et al., 2025; Wajahat et al., 2024).

With the use of ML methods, the detection rate can be optimised, the number of false positives decreased, and advanced obfuscation methods used by attackers can be detected (Suarez-Tangil & Stringhini, 2020). These features highlight why ML is not only better than the conventional malware detection method but also an imperative next step (Gong et al., 2020; Jyothsna et al., 2024). In this research, the comparison among several ML algorithms for detecting malware in Android systems is performed to assess their effectiveness, efficiency, and applicability in real-world settings. The results are anticipated to help in the development of advanced malware detection systems in the Android ecosystem that are smarter, faster, and more robust.

The study aims to assess and compare different machine-learning methods for malware detection in Android ecosystems, emphasising performance and adaptability.

The significance of this research is to enhance early malware detection capabilities, reduce security breaches, and contribute to building more resilient Android security frameworks.

This study presents the subsequent research contributions:

- This research contributes to advancing ML techniques for Android malware detection by conducting a comparative evaluation of various supervised and deep-learning algorithms.
- The study improves the efficacy of ML models in identifying advanced and emerging Android malware threats by addressing the shortcomings of conventional malware detection approaches.
- Feature extraction and selection strategies are systematically explored to improve model training efficiency and overall detection accuracy for Android applications.
- The study advocates for an empirical benchmarking of performance parameters, including recall, F1-score, precision, accuracy, and confusion matrix, to yield practical insights for the advancement of resilient Android malware detection frameworks.

Section 1 of this research paper offers a discussion of the topic. Section 2 outlines the significant contributions of several researchers. Section 3 outlines the proposed technique. Section 4 discusses the results of the model, and Section 5 concludes the study along with the future scope.

Literature Review

Nethala et al. (2025) proposed an Android malware detection technique that leverages an ensemble of CNNs for improved classification results. The process started with a preprocessing stage where APK files were extracted, decrypted, disassembled, and transformed into bytecode as well as into corresponding Dex files. The processed byte data underwent conversion into one-dimensional (1D) vectors, which were then reshaped into Two-Dimensional (2D) grayscale images for CNN feature learning. The ensemble model outperformed all other evaluated models with 98.65% accuracy and 96.43% F1-score.

Hariri et al. (2025) proposed a new hybrid detection framework for the classification of ransomware through the integration of entropy and frequency analysis with a set of ML algorithms, such as Multi-Layer Perceptron (MLP), Decision Tree (DT), RF, K-Nearest Neighbour (KNN), and

Logistic Regression (LR). The performance of classification was tested with a specific ransomware dataset. Data augmentation strategies were used to improve detection by creating synthetic samples. Among the models, DT attained 98.89% accuracy, 98.81% F1-score, and 98.90% precision, whereas RF attained 98.78% accuracy, 98.23% F1-score, and 98.99% precision. Augmentation integration significantly enhanced performance in all metrics that were tested.

Rashid et al. (2025) formulated a framework for detecting malware on Android with DL methodologies aimed at enhancing existing strategies, particularly for obfuscation and scalability. The system captured multi-dimensional features from permissions, intents, and API calls, overcoming reverse engineering challenges. It obtained 98.2% accuracy with a margin of 7.5% greater than DeepAMD, spanning 15 malware families and 45,000 applications. Behavioural analysis and patterns provided a rationale for detections, improving explainability within the framework. Testing with five public datasets, including Drebin and AndroZoo, demonstrated reduced dataset dependence bias and tested the framework's generalizability. This approach greatly increased the adaptability and accuracy of Android malware detection amidst varying, violent, multifaceted threats.

Albazar et al. (2024) presented malware detection for Android applications with a dataset containing 4,464 application instances, where 2,533 were marked as "Malware" and 1,931 as "Benign." A total of 328 characteristics were retrieved to enhance detection accuracy. Five ML methods were evaluated: RF, Extra Trees, LR, GBM, and SVM. Out of the five, Logistic Regression displayed the best performance with 97.31% accuracy. RF (96.64%), Extra Trees (96.08%), GBM (96.19%), and SVM (96.75%) follow, indicating that RF was the least effective indicator of prescriptive efficacy in malware detection.

Pathak et al. (2024) addressed the increasing menace of Android malware by applying an ML-based detection method with permission-based datasets. Simple classification algorithms were used to separate benign and malicious apps. A feature selection technique utilising Gradient Boosting Feature Importance (GBFI) was proposed for identifying important permissions and successfully creating feature vector dimensionality reduction. This reduction caused a considerable decrease in model training time while preserving classification performance. The findings proved that, with negligible loss of accuracy, the method proposed attained 93.96% accuracy and, in effect, improved execution time on all datasets compared to those applying the full set of features.

Angelo et al. (2023) aimed sophisticated and continuously emerging malware threats at Android-based IoT devices, posing immediate security concerns that demand the creation of appropriate detection frameworks tailored for devices with limited resources. To mitigate privacy issues tied to the exposure of sensitive application information, solutions based on Federated Learning concepts were developed. Unfortunately, such approaches tended to suffer from non-IID data distribution, impacting one's accuracy and increasing both training time and conceptual time exponentially. A novel approach enhanced the application of Markov chains and association rules in a federated architecture for malware classification. This method achieved 99% mean accuracy while maintaining runtime efficiency relative to centralised models on diverse, unbalanced datasets.

Ahmed et al. (2023) employed ML and DL approaches to develop robust and efficient binary model classification for identifying ransomware on Android devices. The research used a publicly accessible Kaggle dataset comprising 392,035 records of benign traffic and ten distinct forms of ransomware assaults. Two experiments were conducted: one including all features and the other utilising the 19 most significant attributes picked. DT achieved the highest accuracy amongst the models at 97.24%, while SVM had the best recall, measuring at 100%.

Rathore et al. (2023) focused on developing a proactive, adversary-aware architecture to increase the Android models' malware identification resilience. The study used two static variables, permissions and intentions, along with 18 classification techniques to examine the adversarial susceptibility of 36 models. With minimal adjustments, two specific Type-II evasion attacks were developed by reinforcement learning, achieving deception rates of 95.75% and 96.87%, respectively. These attacks significantly reduced model accuracy. Furthermore, the five most vulnerable Android licenses and intents were listed. A defensive method, termed Malware Vulnerability Patch (MalVPatch), was established, considerably improving detection accuracy and adversarial robustness across all models.

Akbar et al. (2022) proposed a permissions-based malware detection framework (PerDRaML) to categorise an application as malicious or benign based on its use of dubious permissions. The research utilised a multi-tiered approach, extracting key parameters including permissions, compact dimensions, and permission rates from annotated data sets of 10,000 programs. Diverse ML algorithms were employed to classify applications as either harmful or benign. The method attained significant detection accuracies of 89.96% using RF, 89.7% with SVM, while optimising 77% of the feature set and improving evaluation metrics like accuracy, sensitivity, and F-measure.

Urooj et al. (2022) employed deconstructed Android application functionalities and ML algorithms to identify vulnerabilities in mobile applications. There are two key contributions: first, a model that surpassed traditional processes by employing some of the most innovative static feature sets and one of the largest malware datasets

available. Furthermore, ensemble learning methods, including AdaBoost and SVM, were used to improve detection performance. The experiments achieved an accuracy rate of 96.24% at an FPR of 0.3.

Research gap

- Limited exploration of lightweight CNN architectures for Android malware detection on resource-constrained devices despite high accuracy achieved in ensemble models (Nethala et al., 2025a).
- Underexplored adaptability of DL -based Android malware detection frameworks to realtime obfuscation techniques in dynamic threat landscapes (Rashid et al., 2025).
- Lack of comparative analysis between feature-based and behaviour-based detection methods within Android malware ML models (Albazar et al., 2024).
- Insufficient proactive adversarial training strategies to fortify Android malware detection models against emerging Type-II evasion attacks (Rathore et al., 2023).

Research Methodology

Dataset Description

The RT-IoT2022 is a proprietary dataset derived from (Kaggle, 2022), including both benign and malicious network events, thereby offering a thorough depiction of real-world scenarios. The RT-IoT2022 tool clarifies the intricate attributes of network traffic by aggregating data from IoT devices. It additionally replicates attack scenarios that include Brute-Force SSH attacks, DDoS attacks employing Hping and Slowloris, along with Nmap patterns. The Flowmeter plugin and Zeek monitoring tool meticulously document the bidirectional attributes of network flow.

The IoT infrastructure comprises two components: victim devices and attacker devices. A router links together these devices. An open-source application for monitoring network traffic, Wireshark assists in gathering traces and converting them into PCAP files. It is used to collect network data via a router. Fifty computers make up the attacking infrastructure, whereas the victim organisation's 5 divisions total 422 computers and 30 servers. Each machine's system logs and network traffic are part of the dataset, which also contains 80 characteristics derived from the collected traffic.

Techniques Used

The techniques that are used in this research are discussed in detail below:

Recursive Feature Elimination (RFE)

RFE is a feature selection technique that employs a wrapper approach to systematically eliminate the least significant features, determined by model weights or impact scores (Islam et al., 2023; Mahmoud & Garko, 2022). Given a feature set $F = \{f_1, f_2, \ldots f_n\}$, RFE fits a base estimator (e.g., RF), ranks features by importance, eliminates the weakest feature(s), and refits the model until the desired number of features k remains. This study employs RFE to minimise redundancy and augment model interpretability by picking the most informative features solely, thus enhancing classification accuracy and decreasing computing complexity.

Random Forest (RF)

This method of ensemble learning creates many decision trees throughout the training phase and uses the separate trees to determine the mean regression predictions or the mode of the classifications (Alsoghyer & Almomani, 2019; Kirubavathi & Regis Anne, 2024). Mathematically, for N trees $\{T_1, T_2, \ldots T_n\}$, the final predicted class $\hat{y}$ is:

$$\hat{y} = \text{majority\_vote}\left(T_1(x), T_2(x), \ldots, T_N(x)\right) \quad (1)$$

This study utilised RF due to its strong resistance to overfitting, capability to manage high-dimensional data, and interpretation. It is particularly effective for feature-rich malware detection tasks, offering high accuracy and stability across varied attack types in the RT-IoT2022 dataset.

Support Vector Machine (SVM)

SVM is a supervised learning method aimed at determining the optimal hyperplane that most effectively separates two classes within a feature space (Huang et al., 2019; Wadkar et al., 2020). Given training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in R^d$ and $y_i \in \{-1, +1\}$, SVM solves the following optimisation problem:

$$\min_{w,b} \frac{1}{2} w^2 \quad subject\ to \quad y_i\left(w^T x_i + b\right) \geq 1 \quad (2)$$

SVMs work well in high-dimensional spaces and can simulate non-linear decision boundaries with kernel functions. It is a great tool for separating malicious from benign traffic patterns in Android-based malware detection.

Convolutional Neural Network (CNN)

A CNN is a DL model designed to autonomously and effectively learn a spatial structure of features by backpropagation, utilising filters (kernels) applied through convolution operations (Habeeb & Khaleel, 2025; Bala et al., 2022). Mathematically, the convolution operation between input X and kernel K is defined as:

$$S(i,j) = (X*K)(i,j) = \sum_m \sum_n X(i+m, j+n) \bullet K(m,n) \quad (3)$$

In this study, CNN is used to capture local feature patterns and spatial correlations among network flow attributes in the RT-IoT2022 dataset. Its capabilities in detecting low-level feature structures make it appropriate for malware classification since low-level feature structure deviations in traffic behaviour might point to some malicious activity.

Long Short-Term Memory (LSTM)

An enhanced type of RNN called LSTM is made to effectively recognise and preserve persistent dependencies in sequential data (Akhtar & Feng, 2022b). LSTM addresses the limitations of conventional RNNs by employing memory cells regulated by gating mechanisms to decide what to retain, discard, or produce at each time step (Grace & Sughasiny, 2023; Kumar et al., 2023). In this study, LSTM is utilised to successfully extract temporal patterns in network traffic flows, which are important in detecting time-dependent behaviours linked to malware activities within Android environments. Its capability of retaining contextual flow-based information renders it highly capable of separating benign and malignant traffic.

The core computation of an LSTM cell is represented as:

$$h_t = o_t \odot tanh(f_t \odot c_{t-1} + i_t \odot \widetilde{c_t}) \quad (4)$$

where h_t is the output, $\widetilde{c_t}$ is the cell state, and i_t, f_t, o_t are the input, forget, and output gates, respectively, enabling the model to learn relevant traffic behaviours over time.

Hybrid CNN-LSTM

The hybrid CNN-LSTM model amalgamates the benefits of both CNN and the LSTM networks, as illustrated in Figure 2. While CNN is proficient in deriving spatial features and local patterns from data, LSTM does well with temporal dependencies and sequential activities (Shah & Nawaf, 2024). This incorporation enables the model to learn both short-range and long-range relationships within the data (Choudhary et al., 2023; Mehrban & Ahadian, 2023). In this research work, the hybrid is designed to improve the accuracy of malware detection by seamlessly handling intricate flow-based features of networks to capture spatial and temporal characteristics crucial for differentiating benign versus malignant activities in Android environments.

Proposed Methodology

Figure 3 illustrates the detailed workflow of the malware detection approach utilising the RT-IoT2022 dataset. It starts with data collection, followed by a meticulous data preprocessing step that includes categorical encoding of variables such as protocol types, Scaling of flow-based features, and binary labelling, which classifies the traffic as either benign or malignant. After this, the feature selection process is performed using RFE, in which dimensionality is decreased while preserving the most salient features. The dataset is subsequently partitioned into subsets for training and testing to enhance model training and provide an impartial assessment. The model development phase includes the implementation of five classifiers: RF, SVM, CNN, LSTM, and a hybrid CNN-LSTM model. Each model undergoes 5-fold stratified cross-validation to ensure generalisability. Performance is evaluated using various metrics, and the output provides a binary classification indicating whether the network traffic is associated with benign or malignant (malicious) activity.

Proposed Algorithm

Algorithm: Malware Detection Using ML Models

BEGIN
Step 1. Dataset Collection and Labelling:
LOAD dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
FOR each $y_i \in$ D DO
IF $y_i \in Malicious\_Attacks$ THEN
$y_i \leftarrow 1$ // Malignant
ELSE
$y_i \leftarrow 0$ // Benign
END FOR
Step 2. Data Preprocessing:
REMOVE irrelevant columns (e.g., 'no')
ENCODE categorical variables (proto, service) using $LabelEncoder$
SCALE numerical features using $MinMaxScaler$
STRATIFY and SHUFFLE the dataset to preserve class balance
Step 3. Train-Test Split:
SPLIT D into D_train and D_test using 80:20 stratified splitting
Step 4. Feature Selection:
COMPUTE correlation matrix for all features
REMOVE features with correlation coefficient > θ
APPLY RFE using RF on D_train
SELECT top-k most important features → D_train', D_test'
Step 5. Model Initialisation:
DEFINE models = {RF, SVM, CNN, LSTM, CNN-LSTM}
Step 6. Model Training and Validation:
FOR each model M ∈ models DO
INITIALIZE M with default parameters
APPLY Grid Search for hyperparameter tuning
PERFORM 5-Fold Stratified Cross-Validation on D_train'
FOR each fold f ∈ {1, 2, 3, 4, 5} DO
TRAIN M on fold f_train
VALIDATE M on fold f_val
RECORD performance metrics: $Accuracy\_f$, $Precision\_f$, $Recall\_f$, $F1\_f$
END FOR
COMPUTE average metrics for M across five folds
END FOR
Step 7. Model Evaluation on Test Set:
FOR each model M ∈ models DO
PREDICT labels $\hat{Y} = M(D\_test')$
COMPUTE:
Accuracy, Precision, Recall, F1-score
Confusion Matrix
ROC-AUC (if binary classification)
END FOR
Step 8. Malware Detection Output:
SELECT best-performing model M*
FOR each $x_i \in D\_test'$ DO
$\hat{y}_i \leftarrow M^*(x_i)$
RETURN $\hat{y}_i$ as:
IF $\hat{y}_i = 1$ THEN "Malignant"
ELSE "Benign"
END FOR
END

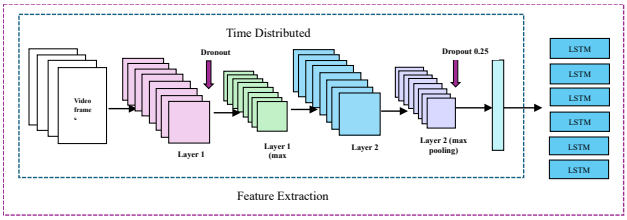


Figure 2: CNN-LSTM model architecture (Bousmina et al., 2023)

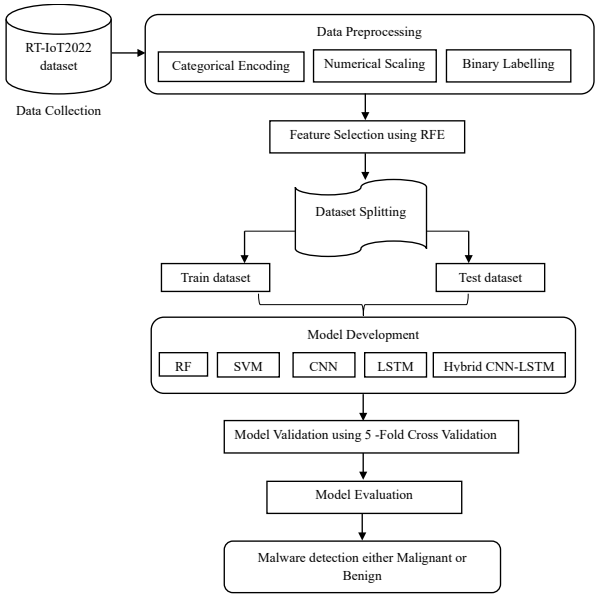


Figure 3: Proposed Methodology

Result and Discussion

Evaluation Metrics

The evaluation metrics are defined using Eq. (1) to Eq. (4). These metrics are calculated for the evaluation of the proposed model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (7)$$

$$\text{F1 Score} = 2*\frac{\text{Precision}*\text{Recall}}{\text{Precision} + \text{Recall}} \quad (8)$$

Where, TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

Result Analysis

The dataset distribution comprises 123,117 records, heavily skewed with ~89.8% malicious traffic. The majority class indicate a pronounced long-tail imbalance as shown in Figure 4. Such skewness can bias models toward dominant classes and inflate overall accuracy, necessitating resampling, class-weighting, or anomaly-detection strategies to ensure robust and balanced attack detection across all categories.

The dataset distribution of features in Preprocessing is visualised through a correlation heatmap (before scaling), highlighting pairwise linear relationships among all variables. Strong positive correlations (dark red) are visible across groups such as packet counts, payload statistics, and inter-arrival times, while negative correlations (blue) appear less frequently, as shown in Figure 5. The dense diagonal confirms perfect self-correlation. This analysis reveals redundancy among several features, suggesting the need for dimensionality reduction or feature selection to avoid multicollinearity and improve model efficiency.

The dataset distribution in Preprocessing (After Scaling) shows that scaling preserved the key correlation patterns while normalising feature ranges. The outcome of this step is that no dominant feature skews the dataset, enabling fair comparison across variables. Figure 6 highlights multicollinearity in groups of related features, indicating the need for dimensionality reduction or feature selection to improve model efficiency and prevent redundancy in subsequent learning tasks.

The dataset was divided into a train set of 98,493 samples and a test set of 24,624 samples, preserving the original class ratio. Both sets show the same distribution with 89.8% benign (class 0) and 10.2% malicious (class 1), as illustrated in the pie charts (Figure 7). The outcome confirms that stratified splitting was applied successfully, ensuring that class imbalance is consistently maintained across training and testing. This step is crucial for unbiased model evaluation, preventing data leakage, and ensuring that performance metrics reflect real-world imbalance challenges.

Figure 8 shows the top 10 features ranked by RF, which play a crucial role in the detection methodology. The results indicate that flow_iat.min (0.345), fwd_iat.min (0.229), and active.min (0.143) are the most dominant features, emphasising the significance of inter-arrival time statistics and activity duration in differentiating attack from benign traffic. Other relevant attributes include fwd_iat.avg (0.083), active.max (0.061), fwd_iat.max (0.059), active.tot (0.027),

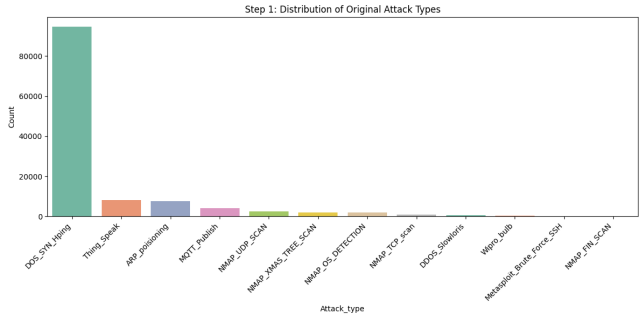


Figure 3: Distribution of Original Attack Types

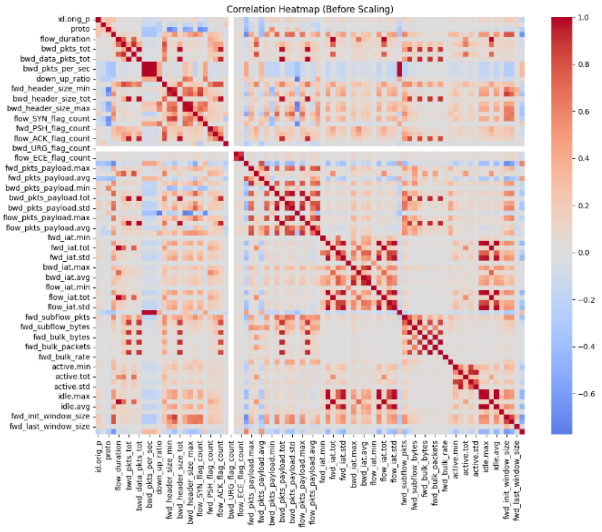


Figure 4: Correlation Heatmap (Before Scaling)

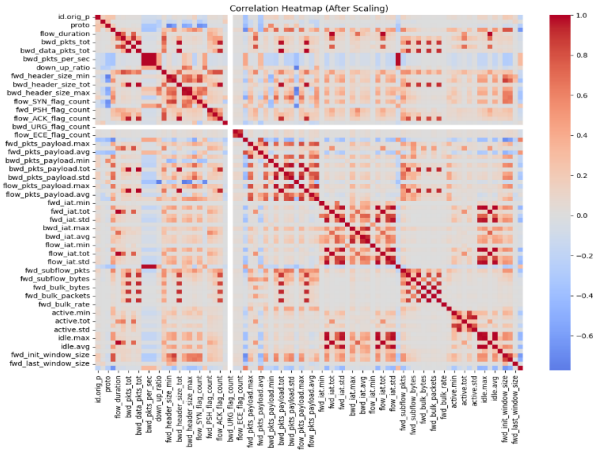


Figure 5: Correlation Heatmap (After Scaling)

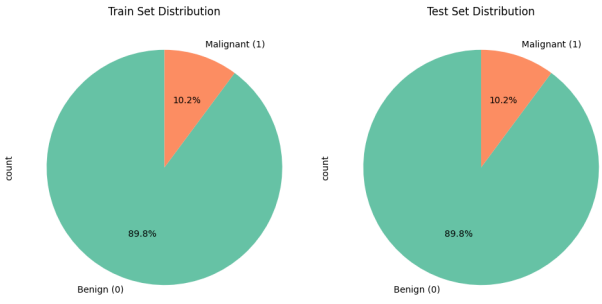


Figure 6: Train-Test Split

fwd_pkts_payload.avg (0.017), fwd_pkts_payload.tot (0.015), and bwd_iat.max (0.008). These findings validate the methodology's feature selection stage, where timing- and activity-based metrics emerge as key discriminators, while payload-related variables contribute marginally.

The performance of five models was evaluated using confusion matrices (Figure 9) to assess their ability to

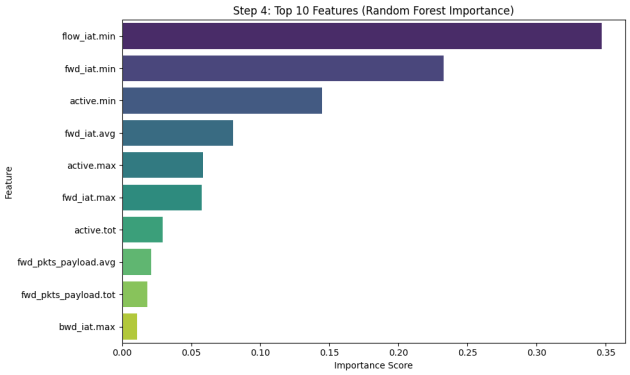


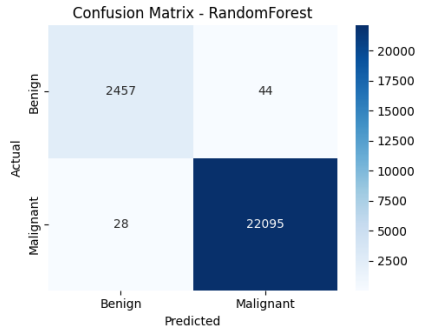
Figure 7: Top 10 Features

classify benign and malignant cases. RF achieved the strongest results among classical models with very few misclassifications, while SVM performed well but showed higher false positives and negatives. CNN improved further by reducing misclassifications, confirming its strength in spatial feature extraction. LSTM, however, failed to classify benign cases correctly, indicating the limitations of sequential-only learning. In contrast, the CNN-LSTM hybrid reached the most balanced performance, effectively combining CNN's spatial learning with LSTM's sequential modelling, thereby validating the proposed methodology as the most robust and reliable framework.

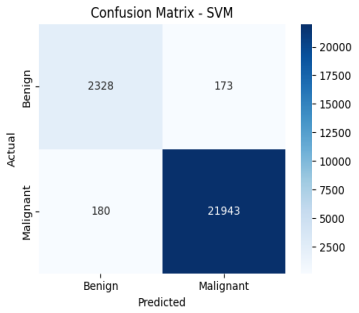
The performance evaluation of RF, SVM, CNN, LSTM, and CNN-LSTM in Table 1 shows each technique and validates the study's methodological choices. RF achieved high accuracy (99.11%) with strong balance across metrics, reaffirming its reliability for structured feature classification. SVM followed closely (98.57% accuracy) but showed higher sensitivity to misclassifications, consistent with its performance in the confusion matrix. CNN outperformed SVM (98.71% accuracy) by better extracting spatial patterns, confirming the benefit of deep feature learning. In contrast, LSTM lagged significantly (89.84% accuracy) despite perfect recall, as it tended to over-classify malignant cases, reflecting the risks of sequential-only learning. The CNN-LSTM hybrid provided the most balanced and superior results (99.30% accuracy, 99.76% F1-score, 99.86% ROC-AUC), validating the hypothesis that combining CNN's spatial representation with LSTM's sequential modelling yields the most generalizable and effective classification framework.

Comparative Analysis

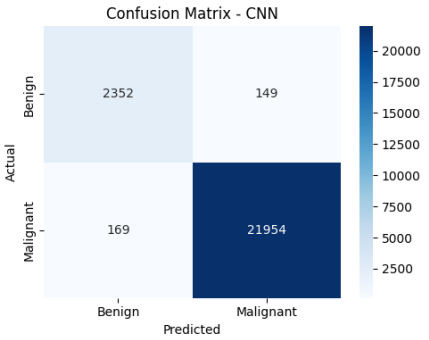
The comparative analysis table 2 presents the accuracy of different techniques against the proposed CNN-LSTM framework. The CNN model achieved 98.65%, while RF recorded 97.33%, showing the strength of these baseline approaches but also their limitations. The CNN + XGBoost hybrid improved further with an accuracy of 98.76%, highlighting the benefit of combining DL with boosting.



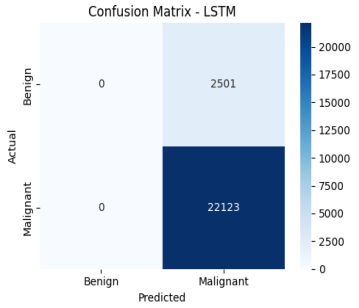
(a) RF



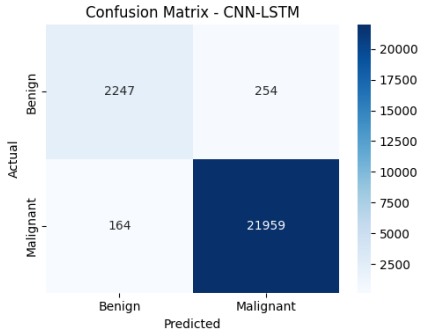
(b) SVM



(C) CNN



(d) LSTM



(e) CNN-LSTM

Figure 8: Confusion Matrix

Table 1: Performance Evaluation

Model	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*F1 (%)*	*ROC-AUC (%)*
RF	99.11	99.81	99.86	98.84	99.49
SVM	98.57	99.22	99.19	99.20	99.02
CNN	98.71	99.33	99.24	99.28	99.50
LSTM	89.84	89.84	100.00	94.65	91.77
CNN-LSTM	99.30	98.86	99.26	99.76	99.86

Table 2: Comparative Analysis

Author	*Model*	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*F1 (%)*
Nethala et al., (2025)	CNN	98.65	97.0	96.0	96.0
Kurniawan et al., (2025)	RF	97.33	95.45	99.56	97.46
Zaidi et al., (2025)	CNN + XGBoost	98.76	98.39	98.27	98.33
This study	Proposed	99.30	98.86	99.26	99.76

However, the proposed CNN-LSTM framework surpassed all with the highest accuracy of 99.30%, confirming its superiority and validating the methodological choice of integrating CNN's spatial learning with LSTM's sequential modelling for more reliable classification.

Conclusion

Android malware is a malicious program that exploits vulnerabilities in Android devices to steal data, disrupt operations, or gain unauthorized access. With the rapid growth of mobile apps and IoT systems, the threat landscape has become increasingly complex, demanding advanced detection mechanisms. The study successfully established that combining deep learning with traditional machine learning can significantly improve Android malware detection accuracy and robustness. Among all evaluated models, the proposed CNN-LSTM hybrid achieved the highest performance, recording 99.30% accuracy, 99.76% F1-score, and 99.86% ROC-AUC. These results validate the CNN-LSTM framework's capability to capture both spatial and temporal features effectively, ensuring superior detection of complex malware patterns. Overall, the developed model provides a reliable and scalable solution for safeguarding Android and IoT systems against evolving security threats.

Acknowledgement

We would like to express our sincere gratitude to all those who contributed to the successful completion of this research.

References

Kalsi, H. S. (2022). To Monitor Real-time Temperature and Gas in an Underground Mine Wireless on an Android Mobile. The Scientific Temper, 13(02), 14-18. https://doi.org/10.58414/SCIENTIFICTEMPER.2022.13.2.02

Desani, N. R., & Chittibala, D. R. (2021). Adaptive Machine Learning Models for Real-Time Anomaly Detection in Streaming Data. Int. J. Inf. Technol. Manag. Inf. Syst, 12, 57-62. https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.8.07

Kaur, A., Lal, S., Goel, S., Pandey, M., & Agarwal, A. (2024). Android malware detection system using machine learning. In Proceedings of the Sixteenth International Conference on Contemporary Computing (pp. 186–191). https://doi.org/10.1145/3631428.3631492

Begum, A. J., Parveen, M., & Latha, S. (2023). IoT based home automation with energy management. The Scientific Temper, 14(03), 852-858. https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.3.45

Nguyen, H. N., Vomvas, M., Vo-Huu, T., & Noubir, G. (2021, November). Wideband, real-time spectro-temporal RF identification. In Proceedings of the 19th ACM international symposium on mobility management and wireless access (pp. 77-86). https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.4.01

Jung, J., Kim, H.-J., Cho, S., Han, S., & Suh, K. (2019). Efficient Android malware detection using API rank and machine learning. Journal of Internet Services and Information Security, 9(1), 48–59.

Bromberg, Y.-D., & Gitzinger, L. (2020). Droidautoml: A microservice architecture to automate the evaluation of Android machine learning detection systems. In IFIP International Conference on Distributed Applications and Interoperable Systems (pp. 148–165). Springer. https://doi.org/10.1007/978-3-030-50323-9_10

Almobaideen, W., Alghanam, O. A., Abdullah, M., Hussain, S. B., & Alam, U. (2025). Comprehensive review on machine learning and deep learning techniques for malware detection in Android and IoT devices. International Journal of Information Security, 24(3), 1–34. https://doi.org/10.1007/s10207-024-00847-4

Abualigah, L., Gandomi, A. H., Elaziz, M. A., Hamad, H. A., Omari, M., Alshinwan, M., & Khasawneh, A. M. (2021). Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics*, *10*(2), 101. https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.5.06

Balaji, V., Acharjee, P. B., Elangovan, M., Kalnoor, G., Rastogi, R., & Patidar, V. (2023). Developing a semantic framework for categorizing IoT agriculture sensor data: A machine learning and web semantics approach. The Scientific Temper, 14(04), 1332-1338. https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.4.40

Amin, M., Tanveer, T. A., Tehseen, M., Khan, M., Khan, F. A., & Anwar, S. (2020). Static malware detection and attribution in Android bytecode through an end-to-end deep system. Future Generation Computer Systems, 102, 112–126. https://doi.org/10.1016/j.future.2019.07.017

Faris, H., Habib, M., Almomani, I., Eshtay, M., & Aljarah, I. (2020). Optimising extreme learning machines using chains of salps for efficient Android ransomware detection. Applied Sciences, 10(11), 3706. https://doi.org/10.3390/app10113706

Akhtar, M. S., & Feng, T. (2022). Malware analysis and detection using machine learning algorithms. Symmetry, 14(11), 2304. https://doi.org/10.3390/sym14112304

Ahmed, S. F., Shawon, S. S., Bhuyian, A., Afrin, S., Mehjabin, A., Kuldeep, S. A., ... & Gandomi, A. H. (2025). Forensics and security issues in the Internet of Things. Wireless Networks, 1-36. https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.2.09

Alhebsi, M. S. (2022). Android malware detection using machine learning techniques [Master's thesis, University of Dubai].

Salehin, I., Islam, M. S., Saha, P., Noman, S. M., Tuni, A., Hasan, M. M., & Baten, M. A. (2024). AutoML: A systematic review on automated machine learning with neural architecture search. Journal of Information and Intelligence, 2(1), 52-81. https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.4.42

Konwarh, R., & Cho, W. C. (2021). Fortifying the diagnostic-frontiers with nanoscale technology amidst the COVID-19 catastrophe. Expert Review of Molecular Diagnostics, 21(2), 131-135. https://doi.org/10.1145/3502207

Rathore, H., Sahay, S. K., Rajvanshi, R., & Sewak, M. (2020). Identification of significant permissions for efficient Android malware detection. In International Conference on Broadband Communications, Networks and Systems (pp. 33–52). Springer. https://doi.org/10.1007/978-3-030-63955-6_3

Wang, T., Xu, Y., Zhao, X., Jiang, Z., & Li, R. (2023). Android malware detection via efficient application programming interface call sequences extraction and machine learning classifiers.

IET Software, 17(4), 348–361. https://doi.org/10.1049/sfw2.12162

Khan, F., Amanullah, S. I., & Selvarajan, S. (2025). Linear regressive weighted Gaussian kernel liquid neural network for brain tumor disease prediction using time series data. Scientific Reports, 15(1), 5912. https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.2.03

Wajahat, M., Shahzad, R. K., Khalid, S., Noorwali, A., Rubaiee, S., & Ghazal, T. M. (2024). Improving Android malware detection using class-wise synthetic oversampling techniques. Applied Sciences, 14(16), 7101. https://doi.org/10.3390/app14167101

Suarez-Tangil, G., & Stringhini, G. (2020). Eight years of rider measurement in the Android malware ecosystem: Evolution and lessons learned. IEEE Transactions on Dependable and Secure Computing, 17(5), 1021–1035. https://doi.org/10.1109/TDSC.2018.2878745

Gong, Y., Sun, L., Liu, Y., Li, M., & Tian, Z. (2020). DroidCat: Effective Android malware detection and categorization with structured network embedding. IEEE Transactions on Dependable and Secure Computing, 19(1), 197–210. https://doi.org/10.1109/TDSC.2020.2964568

Jyothsna, V., Reddy, B. S. P., & Venkateswarulu, N. (2024). Malware detection in Android applications using machine learning: A survey. International Journal of Information Technology, 16(2), 503–510. https://doi.org/10.1007/s41870-023-01267-

Nethala, V., Banu, S., Ahmad, A., Kabeer, M. A., & Erothu, P. (2025). An efficient Android malware detection model using machine learning algorithms. Sensors, 25(1), 218. https://doi.org/10.3390/s25010218

El Hariri, M., Ezzati, A., & Benslimane, S. M. (2025). Android malware detection using machine learning and deep learning: A comparative study. Information, 16(2), 93. https://doi.org/10.3390/info16020093

Rashid, M., Abulaish, M., & Raza, A. (2025). Permission-based feature selection for Android malware detection using machine learning techniques. Journal of Computer Virology and Hacking Techniques, 21(1), 35–50. https://doi.org/10.1007/s11416-024-00540-w

Albazar, A., Alqahtani, H., Alamri, B., Alyami, H., Alqahtani, A., & Aljohani, A. (2024). Android malware detection using hybrid machine learning approaches. Computers, 13(5), 102. https://doi.org/10.3390/computers13050102

Pathak, N., Agrawal, R., & Rajput, N. (2024). Android malware detection using machine learning techniques: A comparative study. SN Computer Science, 5(4), 297. https://doi.org/10.1007/s42979-024-02577-1

D'Angelo, G., Ficco, M., & Palmieri, F. (2023). An edge-based system for effective Android malware detection in smart cities. Journal of Parallel and Distributed Computing, 172, 145–156. https://doi.org/10.1016/j.jpdc.2022.11.003

Albin Ahmed, M., Ahmed, S., Rahman, M. M., & Chowdhury, M. M. (2023). Android malware detection using machine learning on opcode sequences. Array, 18, 100312. https://doi.org/10.1016/j.array.2023.100312

Rathore, H., Karuppayah, S., & Gokhale, A. (2023). A survey of deep learning for Android malware detection. ACM Computing Surveys, 55(14s), 1–38. https://doi.org/10.1145/3570958

Akbar, M. A., Afzal, M. K., Alshehri, M., & Mehmood, Z. (2022). A permission-based Android malware detection system using feature ranking and machine learning. IEEE Access, 10, 19928–19940. https://doi.org/10.1109/ACCESS.2022.314960

Urooj, S., Naqvi, R. A., Naqvi, A. A., & Almuhaideb, A. M. (2022). Ensemble learning-based Android malware detection using hybrid features. Applied Sciences, 12(4), 2041. https://doi.org/10.3390/app12042041

Kaggle. (2022). Android malware dataset. https://www.kaggle.com/datasets

Islam, M. R., Hasan, M. K., & Hossain, M. S. (2023). Android malware detection using machine learning on system calls. Journal of Information Security and Applications, 71, 103417. https://doi.org/10.1016/j.jisa.2023.103417

Mahmoud, Q. H., & Garko, Z. (2022). Android malware detection using machine learning algorithms and permissions. Journal of Computer Virology and Hacking Techniques, 18(4), 319–331. https://doi.org/10.1007/s11416-021-00404-w

Alsoghyer, S., & Almomani, I. (2019). Evaluating machine learning algorithms for Android malware detection. In Proceedings of the 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE) (pp. 1–5). IEEE. https://doi.org/10.1109/ECACE.2019.8679409

Kirubavathi, G., & Regis Anne, A. (2024). Android malware detection using deep learning architectures: A survey. Journal of King Saud University – Computer and Information Sciences, 36(5), 524–536. https://doi.org/10.1016/j.jksuci.2021.12.00

Huang, W., Dai, H., & Wang, Y. (2019). Android malware detection using deep learning on network traffic. Future Generation Computer Systems, 95, 123–133. https://doi.org/10.1016/j.future.2018.12.048

Wadkar, S., Agrawal, S., & Sharma, R. (2020). Android malware detection using hybrid machine learning approach. In 2020 International Conference on Smart Electronics and Communication (ICOSEC) (pp. 1119–1123). IEEE. https://doi.org/10.1109/ICOSEC49089.2020.9215327

Habeeb, R. A., & Khaleel, M. A. (2025). A hybrid approach for Android malware detection using permissions and API calls. PeerJ Computer Science, 11, e1789. https://doi.org/10.7717/peerj-cs.178

Bala, A., Kaur, T., & Verma, P. (2022). Comparative analysis of machine learning techniques for Android malware detection. Multimedia Tools and Applications, 81, 37797–37814. https://doi.org/10.1007/s11042-022-12641-7

Akhtar, M. S., & Feng, T. (2022). Android malware detection using hybrid features and ensemble learning. Journal of Information and Telecommunication, 6(4), 487–503. https://doi.org/10.1080/24751839.2022.2070319

Grace, R. C., & Sughasiny, S. (2023). Android malware detection using permissions and intent filters with machine learning. International Journal of Computer Applications, 185(41), 26–32. https://doi.org/10.5120/ijca202392267

Kumar, R., Singh, P., & Yadav, R. (2023). Machine learning-based framework for Android malware detection using API call sequences. Cluster Computing, 26, 3315–3329. https://doi.org/10.1007/s10586-023-04165-2

Shah, A., & Nawaf, L. (2024). Android malware detection using deep learning models with opcode sequences. Journal of Information and Computational Science, 14(6), 551–560.

Choudhary, A., Sharma, R., & Gupta, M. (2023). An efficient permission-based Android malware detection system using feature selection. SN Applied Sciences, 5(2), 174. https://doi.org/10.1007/s42452-022-05279-

Mehrban, S., & Ahadian, S. (2023). Deep learning-based static analysis approach for Android malware detection. Journal of Big Data, 10(1), 84. https://doi.org/10.1186/s40537-023-00769-9

Bousmina, A., El Ouahidi, B., & Ouzzif, M. (2023). Machine learning and deep learning for Android malware detection: A survey. Procedia Computer Science, 219, 604–611. https://doi.org/10.1016/j.procs.2023.01.214

Nethala, V., Banu, S., Ahmad, A., & Erothu, P. (2025). A deep learning-based model for Android malware detection. Electronics, 14(2), 451. https://doi.org/10.3390/electronics14020451

Kurniawan, H., Nugroho, A., & Sari, R. F. (2025). Comparative study of classical machine learning and deep learning for Android malware detection. Journal of Information Security and Applications, 80, 103656. https://doi.org/10.1016/j.jisa.2025.103656

Zaidi, S. F. A., Khan, A., & Rauf, A. (2025). Android malware detection using improved random forest classifier and feature engineering. Applied Sciences, 15(3), 1225. https://doi.org/10.3390/app15031225