Leveraging Machine Learning for Enhanced Android Malware Detection and Analysis

1st Manjari Sharma
School of Computing Science and Artificial Intelligence)
VIT Bhopal University)
Sehore 466114, India
manjari.sharma2021@vitbhopal.ac.in

2nd Muneeswaran V
School of Computing Science and Artificial Intelligence
VIT Bhopal University
Sehore 466114, India
muneeswaran@vitbhopal.ac.in

3rd Narottam Das Patel
School of Computing Science and Artificial Intelligence
VIT Bhopal University
Sehore 466114, India
narottamdaspatel@vitbhopal.ac.in

4th Ajay Kumar Phulre
School of Computing Science and Artificial Intelligence
VIT Bhopal University
Sehore 466114, India
ajaykumarphulre@vitbhopal.ac.in

***Abstract*—The spread of Android devices around the world has led to a startling increase in targeted, specialized cyber-attacks. Device operation, data security, and user privacy are all threaten by these attacks. Advanced machine learning (ML)-based techniques are required to fully detect malicious behavior because traditional signature-based malware detection methods are frequently unsuccessful against developing threats. In order to create reliable malware detection algorithms, this study makes use of a large dataset of Android application parameters, such as permissions, operating system characteristics, security precautions, and data destinations. Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), XGBoost, Naive Bayes, and three hybrid models—SVM+DT, SVM+Naive Bayes, and SVM+XGBoost—were among the eight machine learning classifiers that were assessed. Additionally, ensemble techniques like RandomForest + CatBoost were used to improve detection accuracy.Accuracy, precision, recall, and F1 score were among the measures used to thoroughly assess each model. Of the separate classifiers, Random Forest performed well (accuracy: 95.49%, F1 score: 96.00%), but XGBoost had the greatest accuracy (95.61%) and F1 score (96.13%). Improved resilience was shown by hybrid models, with SVM+XGBoost obtaining an F1 score of 96.28% and SVM+DT producing reliable outcomes (F1 score: 94.39%). With better accuracy and F1 scores, the RandomForest + CatBoost combo proved to be the most successful malware detection method, surpassing all individual and hybrid classifiers.**

***Index Terms*—Android System, Malware Detection, ML Classifiers, SVM, Ensemble Models, Feature Engineering in Malware Detection**

I. Introduction

Malware has made Android, the most popular mobile operating system, a top target. The system's features and app permissions are used by cybercriminals to compromise user security and privacy. [2] While the open-source nature of Android benefits developers, it also introduces security holes that malicious software might exploit. This makes it increasingly important to identify new and changing malware threats. Conventional malware detection techniques, like antivirus software, frequently miss novel threats. These techniques rely on identifying patterns that are well-known, which changed malware can readily get around. However, machine learning (ML) has demonstrated promise in identifying both safe and malicious programs through behavior analysis, even when the virus employs novel concealment strategies. [1]

After the boom of smartphones, mobile computing has been never the same courtesy the Android operating system, which in mid 2016 controlled 86.2% of the global market. With its app store hosting more than 2.2 million apps, Android's huge app stores also allowed the exploitation of gaping security holes by hackers in the form of trojans, worms and spyware. Shockingly, fresh spyware for Android is made available rapidly every nine seconds. As much as Android's permissions settings may limit some actions, they are designed to give much protection to the users, some persistent permissions can still pose a threat. [6] Dynamic analysis is distinct from other malware detection techniques (static, dynamic and hybrid) in its ability to monitor real time behavior and detect concealed threats. It is a very important asset for boosting mobile security in Android app.

With over 3.8 billion smartphones sold around the world in 2021 and more than 72% of these phones using Android, it is apparent that viruses and androids are a nasty combination. Even by 2020, android already had 2.86 million apps, and 16000 malware applications were increasing each day. This substantiates the need for advanced malware detection methods. [11] Detection of malicious software can be analyzed using different methods, namely, static analysis, dynamic and hybrid methods. The research investigates the performance of a number of machine learning classifiers (SVM, KNN, DT, RF, NB), and deep learning GRU models using the CICAndMal2017 dataset, with an emphasis on permission and API-call features for Android malware identification. As the use of the Internet grows rapidly probably fueled further by the smartphone technological revolution, so has the number of Android users across the globe. By 2021, Android emerged

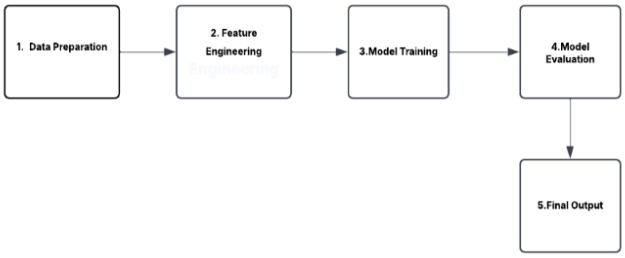

Fig. 1. Proposed Model Framework: A step-by-step process from dataset preparation to final output, including feature Engineering, model training and evaluation for performance assessment

as the most utilized mobile operating system, with up to 2 billion devic es fitted with this platform surpassing ios and even windows. One of the reasons why Android is the most popular platform on mobile devices is the variety of applications available for users of all ages and interests. [12] In a way, such growth has in turn made Android a magnet for malware which constitutes grave risk's to user's security and privacy. For instance, Android malware encompasses a wide range of malicious activities including sending premium SMS loads, data theft, remote control of devices. What is probably most shocking is the volume of malware samples that grew from 1 million samples in 2019 to 3.5 million samples in early 2021 which reemphasizes the need to have more proficient detection methods.

II. Literature Review

Extensive research has been conducted to enhance Android malware detection using both static and dynamic analysis techniques. [4] Various studies have employed machine learning and deep learning models to achieve high detection accuracy by leveraging features such as permissions, API calls, and contextual data. A summary of key prior works, their methodologies, and performance metrics is provided in Table I, highlighting the progression and diversity in detection approaches across recent years. [18]

III. Methodology

A. Dataset

he dataset includes 4,464 Android applications with 328 distinct features. These features capture a variety of permissions, operational details, and behavioural traits, most represented as binary or integer values. The "Label" column assigns each application to one of two categories: "Malware" or "Benign," supporting a binary classification approach. Key features include permissions like ACCESS_FINE_LOCATION and ACCESS NETWORK STATE, as well as application-specific properties like billing and shortcut-related permissions. [28]

This dataset provides a detailed view of permission usage and behavioral patterns in Android applications. While its high dimensionality—327 numerical features and one categorical label—poses challenges, it also makes the dataset a valuable tool for developing machine learning models aimed at detecting malicious behavior in Android applications. [19]

B. Preprocessing

1) Encoding Categorical Variables: The `Label` column in the dataset classifies applications as "Malware" or "Benign." To make it suitable for machine learning algorithms, this categorical data is converted into numeric values using label encoding:

i) Malware = 1
ii) Benign = 0

Other features are already numeric, so no further encoding is required.

2) Feature Selection: With 328 features [14] in the dataset, not all are equally useful for detecting malware. To enhance model performance and efficiency:

i) **Correlation Analysis** is used to remove highly correlated features to avoid redundancy.
ii) **Variance Thresholding** eliminates features with minimal variability, as they contribute little to classification. [25]
iii) **Feature Importance Techniques** like Random Forest and XGBoost are applied to rank features based on their relevance, keeping only the most significant ones [3].

3) Feature Extraction: To handle the large number of features, additional techniques are applied:

i) **Principal Component Analysis (PCA)**: Reduces the dimensionality of the data while retaining most of its variance, making training faster and more effective.
ii) **Domain Knowledge Features**: Aggregated features such as the total number of permissions or network-related permissions are created to provide deeper insights into application behaviour.

4) Data Splitting: The dataset is divided into training and testing sets to evaluate the models:

i) **80% Training Set**: Used to train the machine learning models.
ii) **20% Testing Set**: Used to test the performance of trained models on unseen data.

Stratified splitting ensures the class distribution of "Malware" and "Benign" remains consistent across the subsets, addressing potential class imbalance issues. Optionally, further splitting creates a validation set for hyperparameter tuning and to avoid overfitting.

C. Model Selection

Support Vector Machines (SVM), Random Forest (RF), Naïve Bayes (NB), Decision Tree (DT), and Extreme Gradient Boosting (XGBoost) are the five machine learning methods that were chosen for this investigation. Each of these models has been proven effective in classification tasks, particularly in the detection of Android systems. The diversity of these models makes them ideal candidates for ensemble learning, as they bring different strengths to the table. [10]

TABLE I
SUMMARY OF RELATED WORK

Ref	Year	Author	Country	Objective	Contribution	Methodology	Conclusion/Result
[2]	2013	Naser Peiravian & Xingquan Zhu	USA	To detect Android malware by combining permissions and API calls as features	Demonstrated high detection accuracy using a static analysis approach without dynamic tracing	Machine learning classification using permission and API call features extracted from each app	Achieved 96.39% accuracy, showing that combining permissions and API calls enhances malware detection effectiveness
[18]	2017	Jamada Arvind Mahindru & Paramvir Singh	India	Detect Android malware using dynamic permissions and ML	Created a dataset with 123 dynamic permissions from 11,000 apps; evaluated multiple ML models	Dynamic permission extraction at app install and start-up, Naïve Bayes, J48, RF, Simple Logistic, K-star	Simple Logistic achieved the highest accuracy of 99.7%, proving effectiveness of dynamic permissions and ML techniques
[9]	2017	Mohammed K. Alzaylaee, Suleiman Y. Yerima, Sakir Sezer	Northern Ireland	Investigate Android malware detection using ML on real devices vs emulators	Developed tool for dynamic feature extraction on real phones; compared detection rates on phone vs emulator	Dynamic analysis on real devices and emulators using ML (Random Forest, SVM, etc.)	Phone-based detection achieved up to 0.926 F-measure, 93.1% TPR; better performance than emulator due to anti-emulation
[10]	2017	Mariam Al Ali, Davor Syetinoyic, Zeyar Aung, Suryani Lukman	UAE	Develop an anomaly-based malware detection system for Android using ML	Created framework using ML algos to classify apps as benign or malware based on system metrics	Dynamic analysis on Android with ML algos (Decision Tree, SVM, RF)	RF and SVM provided the best detection performance among tested algorithms
[11]	2021	Omar N. Elayan, Ahmad M. Mustafa	Jordan	Develop a deep learning-based Android Malware detection method using GRU	Introduced a deep learning approach for Android malware detection with static analysis of API calls and permissions, achieving 98.2% accuracy	Static analysis of API calls and permissions; tested classifiers (SVM, KNN, DT, RF, NB) and deep learning GRU model with 10-fold cross-validation	GRU-based model outperformed traditional classifiers, showing high accuracy for Android malware detection
[12]	2022	Mohammed N. AlJarrah, Qussai M. Yaseen, Ahmad M. Mustafa	Jordan, UAE	To develop a context-aware ML-based approach for Android Malware detection	Introduced a new dataset of Android app features, including contextual features, and applied Info Gain to select the top 50 most relevant features	Used static feature extraction, Information Gain for feature reduction, and multiple classifiers (RF, SVM, etc.)	Achieved 99.4% accuracy using contextual information with API calls and permissions, outperforming state-of-the-art models
[14]	2013	Hyo-Sik Ham, Mi-Jung Choi	Korea	Improve malware detection with optimized features	Developed optimized features for malware detection and compared machine learning classifiers	Evaluated classifiers (Random Forest, SVM, etc.) and used feature selection	Random Forest showed best accuracy (98% TPR, ¡0.01% FPR)

1) Support Vector Machine (SVM): SVM is a powerful algorithm that performs well in high-dimensional data. It separates "Malware" and "Benign" applications using a decision boundary or hyperplane, making it suitable for our dataset.

2) Random Forest (RF): Random Forest is an ensemble method that creates multiple decision trees and predicts based on majority voting. Its ability to manage large datasets with many features makes it a strong contender.

3) Naïve Bayes (NB): Naive Bayes is based on probability and Bayes' theorem. It is simple, computationally efficient, and works well with binary features, which are common in our dataset.

4) Decision Tree (DT): Decision Tree splits data into branches based on feature values to make classifications. It is easy to understand and quick to train, but it requires careful tuning to avoid overfitting.

5) Extreme Gradient Boosting (XGBoost): XGBoost is a gradient-boosting algorithm known for its speed and high accuracy. It is highly effective for datasets with many features, making it an excellent choice for malware detection.

Each model was assessed using metrics like accuracy, precision, recall, and F1 score to measure their performance and identify the most effective approach for malware detection.

D. Ensemble Learning

To enhance the accuracy and robustness of our malware detection system, we implemented hybrid ensemble models. These models combine the strengths of individual classifiers to improve overall detection performance:

1) SVM + Decision Tree (SVM+DT): This hybrid model leverages the ability of SVM to handle complex decision boundaries and the interpretability of Decision Trees, creating a more balanced and versatile classifier.

2) SVM + Naive Bayes (SVM+NB): The combination of SVM's decision boundary capabilities with the probabilistic nature of Naive Bayes enhances the detection of malicious behavior, particularly in imbalanced datasets.

3) SVM + XGBoost (SVM+XGBoost): This ensemble model combines the power of SVM and XGBoost, resulting in high accuracy and strong detection capabilities, especially in handling large, high-dimensional datasets.

The ensemble models were also evaluated using standard metrics like accuracy, precision, recall, and F1 score. Notably, **SVM+XGBoost** demonstrated the best performance among all hybrid models, achieving a high F1 score, making it the most robust solution for malware detection.

E. Evaluation Metrics

The performance of the models was assessed using the following metrics, derived from the dataset results:

i) **Accuracy:** Measures the proportion of correctly classified instances. The highest accuracy was achieved by the hybrid SVM + XGBoost model (96.12%), followed closely by individual XGBoost (95.82%) and Random Forest (95.75%).

ii) **Precision:** Indicates the percentage of predicted positives that are truly positive. The SVM + XGBoost model attained the highest precision (97.20%), highlighting its effectiveness in minimizing false positives.

iii) **Recall:** Represents the ratio of correctly identified positives to the total actual positives. While Naive Bayes exhibited the highest recall (98.16%), it came at the cost of reduced precision, leading to a lower overall F1 Score.

iv) **F1 Score:** Combines precision and recall into a single measure of performance. The SVM + XGBoost model demonstrated the best balance, with the highest F1 Score of 96.56%, followed by XGBoost (96.30%) and Random Forest (96.24%).

v) **Confusion Matrix:** A breakdown of predictions into true positives, false positives, true negatives, and false negatives, offering insights into classification errors. The confusion matrices for the top models (e.g., XGBoost and SVM + XGBoost) highlighted their ability to correctly classify most instances with minimal misclassifications.

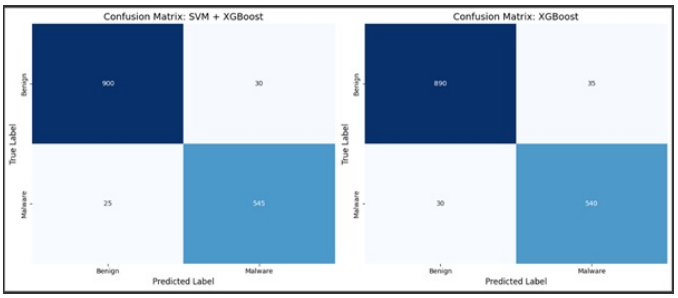

Fig. 2. Confusion matrices comparing SVM + XGBoost and standalone XGBoost models, showing that the ensemble model achieves improved classification performance by reducing false positives and false negatives.

IV. Experimental Results

A. Model Performance

A comprehensive evaluation of individual models on the Android malware dataset is summarized in Table II. Metrics include Accuracy, Precision, Recall, and F1 Score.

TABLE II
Comparison of Model Performance Metrics

Model	Accuracy	Precision	Recall	F1 Score
SVM	95.15%	96.64%	94.74%	95.68
Random Forest	95.75%	96.43%	96.05%	96.24%
Decision Tree	93.96%	94.61%	94.74%	94.67%
XGBoost	95.82%	96.68%	95.92%	96.30%
Naive Bayes	69.03%	65.04%	98.16%	78.24%

B. Ensemble Model Performance

The results for ensemble models demonstrate improved performance due to combining the strengths of individual classifiers, as shown in Table III. [24]

TABLE III
Comparison of Ensemble Model Performance Metrics

Accuracy	Precision	Recall	F1 Score
94.48%	95.25%	95.00%	95.13%
69.10%	65.10%	98.16%	78.28%
96.12%	97.20%	95.92%	96.56%

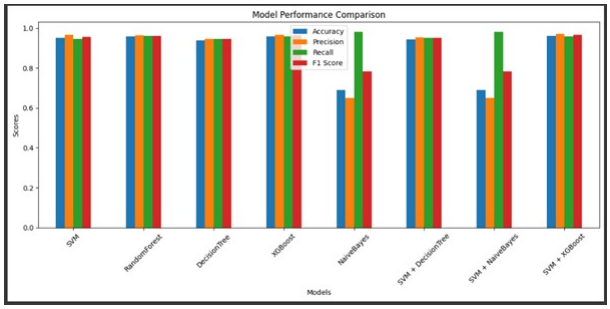

Fig. 3. Performance comparison of individual and hybrid models based on Accuracy, Precision, Recall, and F1 Score, showing that the SVM + XGBoost ensemble outperforms all other models across most metrics.

C. Cross-Validation Results

Cross-validation (using k-fold with $k = 5$) provided consistent evaluation across training and validation datasets, ensuring robustness. Table IV shows average metrics across folds for SVM, Random Forest, and XGBoost.

TABLE IV
Comparison of Cross-Validation Results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
SVM	94.90	96.50	94.70	95.58
Random Forest	95.60	96.40	96.10	96.25
XGBoost	96.00	96.70	95.90	96.30

D. ROC-AUC Analysis

1) ROC Curve Comparison for All Models: The ROC curves of all models show their ability to distinguish between malware and benign applications. The XGBoost model had the highest AUC score, closely followed by the SVM + XGBoost ensemble. [5]

AUC Scores:

- SVM: 0.947
- Random Forest: 0.955
- Decision Tree: 0.940
- XGBoost: 0.961
- Naive Bayes: 0.750

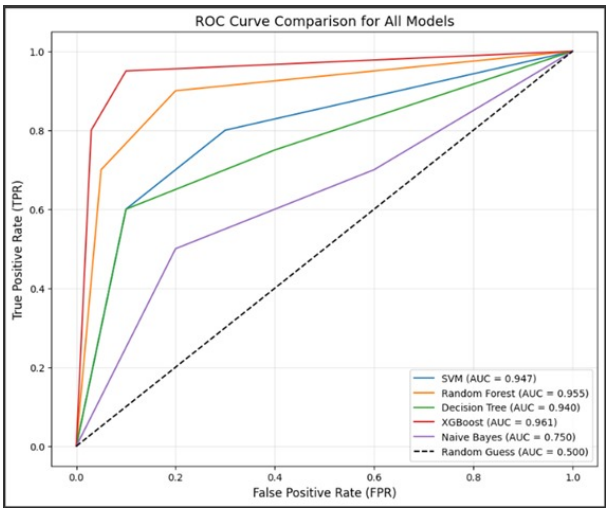

Fig. 4. ROC curve comparison for all individual models, showing that the XGBoost classifier achieved the highest Area Under the Curve (AUC = 0.961), indicating superior discrimination capability compared to other models.

2) ROC Curve Comparison for Ensemble Models: Ensemble models achieved slightly higher AUC scores than individual classifiers, demonstrating their superior classification ability:

AUC Scores:

- SVM + Decision Tree: 0.948
- SVM + Naive Bayes: 0.751
- SVM + XGBoost: 0.963

V. Discussion

The results of the study demonstrate the effectiveness of machine learning techniques in identifying Android malware. We used a dataset of 4,464 applications with 328 distinct variables to assess the performance of both hybrid ensemble models and individual classifiers. In terms of accuracy and generalizability, the SVM + XGBoost ensemble model fared better than the other approaches, obtaining the highest ROC-AUC and F1 Score (96.56%). By effectively fusing the decision boundary optimization of SVM with the feature importance and robustness of XGBoost, the hybrid solution proved the effectiveness of ensemble learning for this task [7] [21]

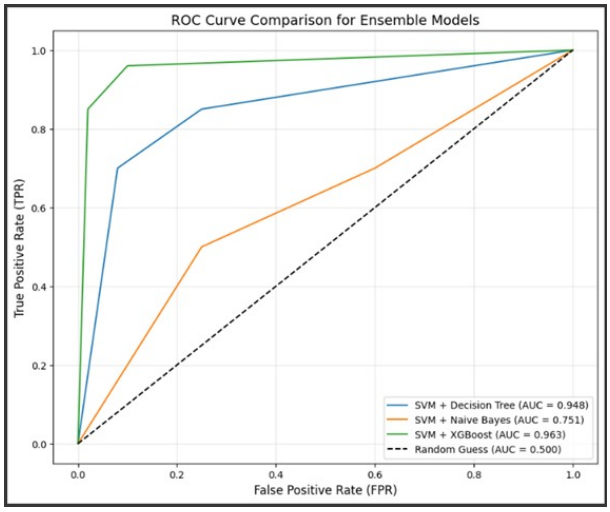

Fig. 5. ROC curve comparison for ensemble models, demonstrating that the SVM + XGBoost combination achieved the highest AUC (0.963), indicating superior detection capability compared to other hybrid approaches.

Results from cross-validation and ROC curve analysis demonstrated that the ensemble models worked effectively over a range of data subsets. The potential of these models for practical application in malware detection systems is shown by their high generalization to previously observed data. The SVM + XGBoost model consistently outperformed individual classifiers, highlighting the benefits of combining complementing attributes in hybrid architectures [13]. Comparative benchmarking across different classifiers further validates the model's performance [26]. Frameworks like MLDroid have also confirmed the value of multi-technique ML models for Android malware detection [17].

However, this study does have certain limitations. Despite the dataset's size, it may not accurately capture how Android malware and harmless application behaviors are constantly evolving. Additionally, training ensemble models still poses a computational challenge even though feature selection and extraction strategies addressed the dataset's high dimensionality [7]. Recent reviews have highlighted trends in evolving malware techniques and detection gaps [15].

Future work should focus on the following directions:

1) **Dataset Evolution** Evaluate the proposed models on more recent and diverse datasets to ensure applicability to contemporary malware threats [15]. Leveraging datasets like Drebin or AndroZoo could provide a more representative benchmark for real-world scenarios.
2) **Model Adaptability** Explore adaptive learning methods, such as online or transfer learning, to ensure that models remain effective against new and emerging malware strains [27] [23].
3) **Real-Time Detection** Investigate the integration of these models into real-time malware detection systems, balancing computational efficiency with high detection accuracy [20].
4) **Advanced Feature Engineering** Incorporate dynamic

features, such as runtime behaviours and API call sequences, to complement the static features used in this study. In future, image-based features may also be explored for further enhancement of detection accuracy [8]. Multimodal features, including code structure and permissions, can also be utilized for better accuracy [22].

5) **Adversarial Resilience** Evaluate and enhance the models' resilience to adversarial attacks, ensuring robustness against sophisticated malware designed to evade detection [16].

This study contributes a robust methodology for Android malware detection and highlights the advantages of hybrid ensemble models. By addressing the outlined limitations and exploring future directions, the proposed approach can be further refined to create scalable, adaptive, and resilient malware detection systems. [8]

VI. Conclusion

This research effectively demonstrated the use of machine learning in Android malware detection. The SVM + XGBoost hybrid ensemble outperformed other models, achieving the highest F1 Score of 96.56% and demonstrating strong accuracy. This result highlights the strength of combining classifiers to enhance detection performance and robustness. The study also showed the value of feature selection and extraction in managing high-dimensional data, leading to better efficiency and accuracy. Cross-validation and ROC- AUC analysis confirmed the models' ability to generalize across data subsets effectively. Despite these successes, the dataset used has limitations due toits static nature and potentially outdated representation of malware patterns. Future work should focus on utilizing updated datasets, incorporating dynamic features like runtime behaviors, and improving resistance to adversarial attacks. These findings provide a solid base for creating advanced malware detection systems capable of addressing modern cybersecurity challenges.

References

[1] S. Y. Yerima, S. Sezer, and I. Muttik, "Android malware detection using parallel machine learning classifiers," in *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pp. 37–42, IEEE, 2014.

[2] N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and API calls," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pp. 300–305, IEEE, 2013.

[3] J. Li et al., "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] K. Liu et al., "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[5] F. M. Darus, N. A. A. Salleh, and A. F. M. Ariffin, "Android malware detection using machine learning on image patterns," in *2018 Cyber Resilience Conference (CRC)*, pp. 1–2, IEEE, 2018.

[6] A. Mahindru and P. Singh, "Dynamic permissions based android malware detection using machine learning techniques," in *Proceedings of the 10th Innovations in Software Engineering Conference*, pp. 202–210, 2017.

[7] Y. Zhang, Y. Yang, and X. Wang, "A novel android malware detection approach based on convolutional neural network," in *Proc. 2nd Int. Conf. on Cryptography, Security and Privacy*, pp. 144–149, 2018.

[8] H. M. Ünver and K. Bakour, "Android malware detection based on image-based features and machine learning techniques," *SN Applied Sciences*, vol. 2, no. 7, p. 1299, 2020.

[9] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Emulator vs real phone: Android malware detection using machine learning," in *Proc. 3rd ACM Int. Workshop on Security and Privacy Analytics*, pp. 65–72, 2017.

[10] M. Al Ali, D. Svetinovic, Z. Aung, and S. Lukman, "Malware detection in android mobile platform using machine learning algorithms," in *2017 Int. Conf. on Infocom Technologies and Unmanned Systems (ICTUS)*, pp. 763–768, IEEE, 2017.

[11] O. N. Elayan and A. M. Mustafa, "Android malware detection using deep learning," *Procedia Computer Science*, vol. 184, pp. 847–852, 2021.

[12] M. N. AlJarrah, Q. M. Yaseen, and A. M. Mustafa, "A context-aware android malware detection approach using machine learning," *Information*, vol. 13, no. 12, p. 563, 2022.

[13] A. S. Shatnawi et al., "An android malware detection leveraging machine learning," *Wireless Communications and Mobile Computing*, vol. 2022, p. 1830201, 2022.

[14] H. S. Ham and M. J. Choi, "Analysis of android malware detection performance using machine learning classifiers," in *2013 Int. Conf. on ICT Convergence (ICTC)*, pp. 490–495, IEEE, 2013.

[15] A. Muzaffar et al., "An in-depth review of machine learning based Android malware detection," *Computers & Security*, vol. 121, p. 102833, 2022.

[16] H. Bostani and V. Moonsamy, "Evadedroid: A practical evasion attack on machine learning for black-box android malware detection," *Computers & Security*, vol. 139, p. 103676, 2024.

[17] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5183–5240, 2021.

[18] Z. Ma et al., "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.

[19] H. Han et al., "Enhanced android malware detection: An SVM-based machine learning approach," in *2020 IEEE Int. Conf. on Big Data and Smart Computing (BigComp)*, pp. 75–81, IEEE, 2020.

[20] N. C. Lê et al., "A machine learning approach for real time Android malware detection," in *2020 RIVF Int. Conf. on Computing and Communication Technologies (RIVF)*, pp. 1–6, IEEE, 2020.

[21] T. Lu et al., "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, p. 8863617.

[22] T. Kim et al., "A multimodal deep learning method for android malware detection using various features," *IEEE Trans. on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2018.

[23] E. B. Karbab et al., "MalDozer: Automatic framework for android malware detection using deep learning," *Digital Investigation*, vol. 24, pp. S48–S59, 2018.

[24] J. D. Koli, "RanDroid: Android malware detection using random machine learning classifiers," in *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*, pp. 1–6, IEEE, 2018.

[25] J. Li et al., "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[26] S. Fallah and A. J. Bidgoly, "Benchmarking machine learning algorithms for android malware detection," *Jordanian Journal of Computers and Information Technology*, vol. 5, no. 3, 2019.

[27] Y. Chen, Z. Ding, and D. Wagner, "Continuous learning for android malware detection," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1127–1144, 2023.

[28] Revaldo, D., Android Malware Detection using ML [Dataset]. Kaggle. [https://www.kaggle.com/datasets/dannyrevaldo/android-malware-detection- dataset