A Hybrid Static-Dynamic Analysis Model for Android Malware Detection: Design, Implementation and Comparative Assessment

Shatha Hamead Othman and Huda Abdulaali Abdulbaqi
Department of Computer Science, College of Science, Mustansiriyah University, 10052 Baghdad, Iraq
shathaalqaisy20@uomustansiriyah.edu.iq, huda.it@uomustansiriyah.edu.iq

Keywords: Android, Malware Detection, Hybrid Approach, Static Analysis, Dynamic Analysis.

Abstract: The attack surface for cyber threats targeting the Android platform has grown dramatically due to the widespread use of Android handsets, the openness of the Android ecosystem, coarse-grained authorization structures, and the invocation of third-party code. The effectiveness of machine learning methods in identifying Android malware has been shown by recent studies. In this work, we provide a hybrid analysis approach that combines static and dynamic analysis to detect Android malware in a dependable and efficient manner. This hybrid model increases detection precision and accuracy while also improving feature extraction procedures. Additionally, we compare the results of static and dynamic analysis with the hybrid approach and look at how each affects classification performance separately. According to experimental results, our hybrid model performs better than other models, achieving 98.9% accuracy, 99.1% precision, 98.3% recall, and 98.7% F1-score. These results indicate a 12% and 22% increase in detection accuracy, respectively, over static and dynamic analytic techniques. Additionally, our findings highlight the limitations of using static or dynamic analysis alone, in terms of detection accuracy, resource efficiency, and behaviour profiling of Android malware. Overall, the study highlights the effectiveness of hybrid analysis in enhancing malware detection systems and achieving more reliable and accurate security classifications.

1 INTRODUCTION

Android has emerged as the most popular smartphone operating system due to the quick development of mobile intelligent terminals [1]. In the mobile sector, Android OS held a global market share of over 71.9% as of July 2023. Google Play is the official app store running on Android smartphones. There were over 2.9 million applications on it as of May 2021. Approximately 2.5 million of them are categorized by AppBrain as standard apps, while 0.4 million are classified as low-quality apps. Android is more attractive to thieves due to its widespread use, but it is also more susceptible to viruses and malware [2]. However, because Android apps are widely distributed, open-source, and have coarse-grained permission management, they can be obtained from potentially dangerous third parties outside of the official Android Market, leaving the platform open to malware attacks [3]. To address these security concerns, various methods for detecting Android malware have been proposed [4]. One of the best methods for detection is machine learning [5]. There are a couple of primary sources of features in the machine learning-based Android malware detection implementation: both dynamic and static extraction [6].

The Dalvik bytecode, native code, manifest, sound, image, and inverted APK files are the sources of the static features. By executing APK files in the environment, the dynamic characteristics are retrieved from code execution, paths, variable value tracking, sensitive function calls, log records, and other behaviors that take place during the application's operation [4]. By employing sophisticated preprocessing to choose the best samples, providing rich data, and enabling machine learning for effective malware detection using the CIC-AndMal2017 dataset, this study suggests a reliable way for identifying malicious Android applications. The strategy makes use of machine learning methods, including Random Forest, Support Vector Machines, K-Nearest Neighbors, and Stacking Ensemble Model to classify applications using dynamic analysis, which deals with dynamic features, and static analysis, which concentrates on static features. Following the implementation of these strategies, the same model was then assessed through hybrid analysis, which combines static and dynamic

data to maximize feature extraction and improve malware detection efficacy and precision. When compared to the hybrid strategy, both individual approaches showed a considerable decrease in performance.

Therefore, it is essential to compare how static, dynamic, and hybrid analysis affect an attack detection system's performance. Every analysis type has its own advantages and disadvantages. For example, static analysis looks at the application's code without running it, which makes scanning quicker and more comprehensive but may miss runtime behaviors. Although it frequently takes more time and processing resources, dynamic analysis, on the other hand, watches how the program behaves while it is running and can reveal dangerous activity that static approaches could miss. By utilizing the advantages of both methods, hybrid analysis may be able to maximize detection accuracy. Assessing each of the three approaches yields a thorough comprehension of how each one affects system performance separately and in combination. The main contribution of this paper is:

- Assess a number of machine learning classifiers using three different analysis types: static, dynamic, and hybrid.
- Contrast the outcomes of malware identification utilizing features based on static and dynamic analysis.
- Compare the cost of using various feature types in terms of the amount of time needed for training.
- A large dataset and a hybrid technique can achieve high classification accuracy.

The rest of this paper is displayed as follows: Section 2 provides an overview of the related work. The suggested methodology is introduced in Section 3 that includes a sub-subsection: 3.1 Dataset Description, 3.2 The Specifics of the Suggested Approach. The result and discussion are provided in section 4. Whereas section 5 summarizes the conclusions.

2 RELATED WORK

There are three methods to examine the malicious mobile application: (Hybrid analysis, dynamic analysis, and static analysis) based on various retrieved features, either dynamic, static, or both. This section presents pertinent studies on the application of machine learning algorithms and various analysis methodologies for Android malware detection using the CICAndMal2017 dataset.

2.1 Static Analysis-Based Features

The program's resources and source code are used for static analysis without executing the code [7]. In static malware analysis, we have extracted different types of static features i.e. API calls, intents, permissions and command strings. Omar N. Elayan and Ahmad M. Mustafa [8] they offer a method for identifying malware in Android applications. Additionally, they compare deep learning approaches with conventional machine learning techniques A portion of the CICAndMal2017 dataset, which includes 365 malware samples and 347 benign samples of Android apps, was used in this investigation. They extract two static features from Android applications: permissions and Application Programming Interface (API) calls. Hence, Experiments show that the deep learning algorithm performs better, with a precision rate of 98.2%. Recep Sinan ARSLAN [9] a suggested ensemble machine learning model is presented in this work to efficiently detect and categorize several kinds of malware, such as ransomware, scareware, adware and SMS malware. The CICAndMal-2017 dataset, was used to train and assess the model in issue. The classification of malware and benign software is not the main focus of this work; rather, it is the identification of malware types. 90.4% accuracy was attained in identifying the malware category in the studies that were carried out using 486 instances of malicious samples. Additionally, the F1-score, precision, and recall were 90.4%.

2.2 Dynamic Analysis-Based Features

Analyzing a running program's attributes is known as a dynamic analysis, it is performed while executing the code [10]. In dynamic malware analysis, we have extracted different types of dynamic features i.e. cryptographic operations, dynamic permissions, actions, IP address, information leaks and system calls. M.Gracea and Dr. M. Sughasiny [11] The study uses a hybrid Long Short-Term Memory (LSTM-SVM) model and an Aquila optimizer. The Aquila optimizer is used in the cross-validation feature extraction process to determine which dynamic features are most appropriate. The Android malware dataset CICAndMal2017 is used. For the suggested model, the performance metrics Accuracy, Precision, Recall, Error, Specificity, F1_score, Negative Predictive Value (NPV), False Positive Rate (FPR), and False Negative Rate (FNR) are 0.97, 0.94, 0.90, 0.03, 0.95, 0.93, 0.96, 0.10, and 0.02 per sec. Zhixing Xue, et al. [12] developed an ensemble-learning strategy based on stacking to detect and categorize

malware on Android devices Through the process of clustering, the model first eliminated any third-party data from the stream in order to obtain clean beginning information. (73) Statistical features, as the most appropriate for classification, and (86) flow-level attributes, which were later obtained by feature engineering. They construct the classification model using CICAndMal2017 dataset. The experiment's findings demonstrate that by excluding third-party traffic and utilizing relevant traffic features, classification accuracy improved to as much as 96.7%.

2.3 Hybrid Analysis-Based Features

Hybrid analysis is a blend of dynamic and static analysis, which can improve the accuracy and efficiency of Android malware detection [1]. Jiayin Feng, et al. [13] suggested HGDetector, a multi-featured hybrid malware detection and category categorization technique. This method employs the dynamic network traffic features to generate the node interactivity graph and edge-node graph after extracting the static function call graph and creating the network behavior function call graph. Finally, the accuracy of detecting malware was then tested using a variety of classifiers in conjunction with the suggested HGDetector. The experimental findings show that the suggested HGDetector model's accuracy with MLP, LR, RF, and SVM classifiers in the CICAndMal2017 dataset is 97% with SVM.

3 METHODOLOGY

There are three primary phases in the proposed model for the Android malware detection system they are: (Data pre-processing phase, Feature selection phase, Classification phase). Each of these stages is essential to guaranteeing the detecting system's precision and effectiveness. Some of these phases require a number of steps to accomplish their objectives. Several machine learning methods are used in the classification step to effectively identify and categorize Android malware.

Figure 1, which illustrates the system's total workflow, provides a clear illustration of these phases.

3.1 Dataset Description

This study has utilized a portion of the CICAndMal2017 dataset, generated and released by Lashkari et al. [14] and available on the Canadian Institute of Cyber Security's website [15]. In this paper, 70,000 samples were chosen from the original dataset due to constraints on computing power and storage; this dataset consists of 18,000 malicious and 52,000 benign samples. There are 84 unique features in the dataset before applying any process. This dataset's malware samples are divided into four groups: (Adware, Ransomware, Scareware, SMS Malware). According to their names, each type carries out particular attacks. The samples include 42 distinct malware families from the four previously listed categories. Data is gathered on actual devices in three distinct states: To capture realistic behavior, this was done during installation, before to the restart, and following the restart [9].

3.2 The Details of the Suggested Methodology

The three phases of the suggested methodology are: data preprocessing phase, feature selection phase, and classification phase.

3.2.1 Data Preprocessing Phase

Our proposed work focus on employing sophisticated preprocessing to select the best samples, delivering rich data and enabling machine learning for efficient malware detection.

Dataset Preprocessing follows the procedure as below:

Step 1. Type conversion. Non-numeric columns (e.g., strings or categorical values) are transformed to numeric representations.

Step 2. Missing Value Handling (NaN). The usual procedure for dealing with missing data is to either fill NaN values or drop them.

Step 3. Drop high cardinality columns. This function removes columns when the ratio of unique values to total rows exceeds a predefined threshold (0.9).

Step 4. Duplication Removal. Duplicate rows are deleted from the dataset so that the final dataset only contains unique data points.

Step 5. Normalization (Standardization). Standard Scaler is used to standardize the numeric columns.

Step 6. Memory optimization. Changing "float64" columns to "float32" and "int64" columns to "int32".

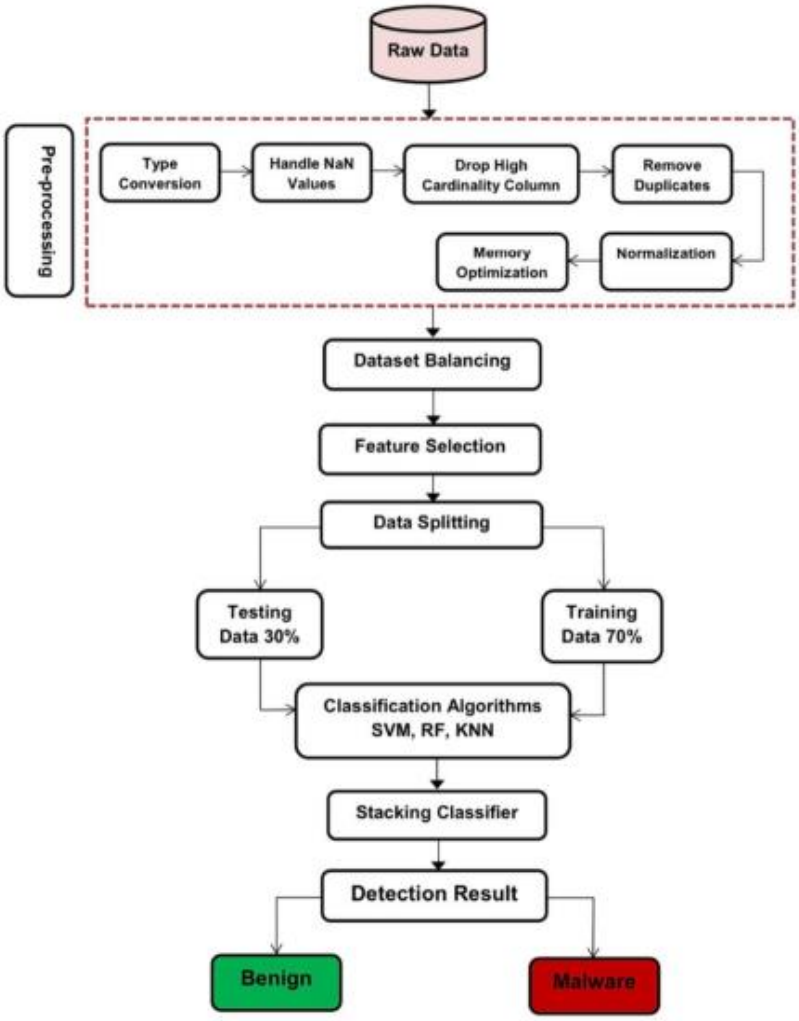

Figure 1: The suggested model.

After applying these steps, class imbalance frequently arises from unequal class distribution. This phase creates a balanced malware detection dataset derived from CIC-AndMalS2017. The ADASYN algorithm resolves dataset imbalance by generating synthetic samples for the minority class (malware) through data distribution analysis, achieving a 60% benign to 40% malware balance.

3.2.2 Feature Selection Phase

Feature selection enhances malware detection effectiveness by eliminating redundant and irrelevant features [1]. Selecting optimal features improves accuracy, efficiency, and model interpretability. A filter-based technique (CFS) was applied, beginning with the mathematical foundation of the Pearson correlation coefficient [16]. The Pearson correlation between two random variables, X and Y, can be calculated mathematically as illustrated in (1) and (2):

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}. \quad (1)$$

where *Cov (X, Y)* is the covariance between x and y, and σ_x and σ_y are the standard deviations of x and y. In a sample of size n, it can be calculated as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_{i-}\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_{i-}\bar{y})^2}}. \quad (2)$$

where the sample means of x_i and y_i are denoted by $\bar{x}$ and $\bar{y}$ respectively.

Each feature's correlation with the target class was computed. A threshold of 0.2 was then applied to identify features with strong linear associations to the

target. This threshold was chosen to detect weak relationships while avoiding information loss.

3.2.3 The Classification Phase

Once the feature selection process is complete, 30% of the dataset is used for testing, while 70% is used for training. In this phase, we decided to implement machine learning methods in the classification of the malware detection system. Since labels are present in the sample data set, supervised machine learning has been used in the study with four machine learning methods: Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), and Stacking Ensemble Model. They are used for the purpose of comparing performance and enhancing the validity of the models. We go over each algorithm's features in turn, as follows:

3.2.3.1 Support Vector Machine (SVM)

SVM generates one or more hyperplanes, and the most effective one separates data into classes, with the most important class division occurring [17]. Our implementation of SVM on the CIC-AndMal2017 dataset utilize the RBF kernel to manage non-linear relationships and perform hyperparameter Tuning uses the parameter grid as follows:

param_grid = { 'C': [0.1, 1, 10, 100, 1000],
'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
'kernel': ['rbf']}.

The use of cross-validation is intended to alternate between different values of C and gamma in order to find the optimal parameters, which will then be used in the subsequent classification. after applying 5-fold cross-validation for each candidate, choose parameters that optimize cross-validation accuracy as follows:

('svm_rbf', SVC(gamma=0.1, C=100)).

Classify the test data using the learned SVM model. The two classes are clearly distinguished by this hyper-plane. Hence, this algorithm determines if an application is malicious or benign.

3.2.3.2 Random Forest (RF)

RF is an ensemble method that at each split point selects a random sample of features to determine feature importance [18]. This study trains 100 decision trees; using 100 trees is a practical choice as it provides accurate and stable results, reduces variance, and balances performance and speed. It computes entropy before and after each split and selects the split with the highest information gain. The majority vote across all trees determines the final prediction. RF was implemented as the best estimator in the stacking model with:

('rf', Random Forest Classifier(n_estimators =100,))

The highest accuracy was achieved using these estimators across the dataset, where the predicted target with the highest number of votes is the final prediction.

3.2.3.3 K-Nearest Neighbor (KNN)

K-Nearest Neighbor measures distances between data points to classify new instances based on proximity to training points [19]. This work implemented KNN starting with 40 neighbors and computed error rates for k-values from 1 to 39. As shown in Figure 2, k=4 was selected for the final stacking model as it achieved the optimal balance between low error and reduced overfitting risk.

3.2.3.4 Stacking Ensemble Model

Figure 2 illustrates the proposed ensemble model incorporating SVM, KNN, and RF classifiers as base learners. Each model was trained independently on training and test sets. These base models were then integrated into the stacking ensemble framework. The stacking model uses the three base models as individual learners with a Random Forest classifier as the meta-learner. In the stacking framework, predictions from all models are combined using malware-category accuracy weights. GridSearch from scikit-learn was employed to optimize each model's parameters.

3.2.4 Performance Metrics

Accuracy, precision, recall, and F1-score metrics were employed to evaluate classifiers and identify the top-performing model [20].

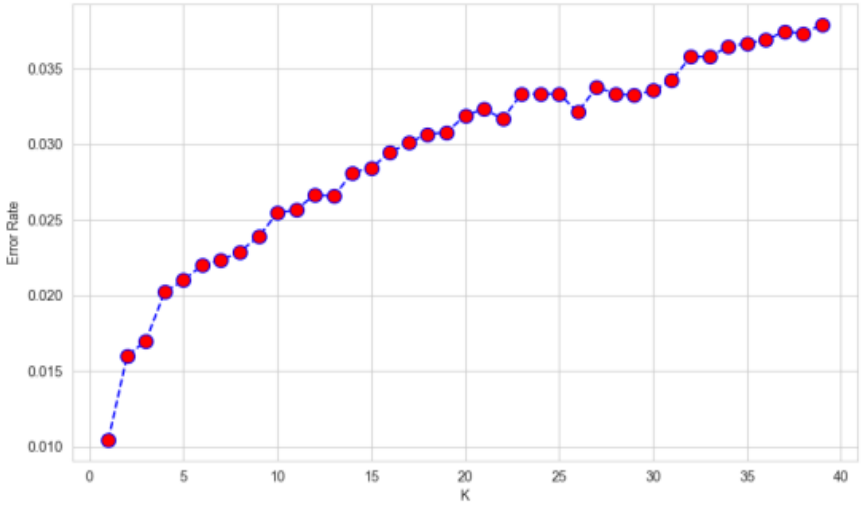

Figure 2: KNN error rate.

Accuracy measures the overall percentage of correct predictions among all instances. Precision indicates the proportion of true positive predictions relative to all positive predictions, reflecting prediction reliability. Recall, also called sensitivity or the true positive rate, measures the percentage of actual positive instances successfully detected by the model. The F1-score combines precision and recall into a balanced metric, particularly useful for imbalanced datasets where both false positives and false negatives have equal importance.

4 RESULT AND DISCUSSION

In addition to testing the suggested ensemble model, machine learning methods were also used in this study. Initially, to train the classifiers, the CIC-AndMal-2017 dataset was used, with a size of 37 GB containing 70,000 samples selected from the original dataset. After performing the preprocessing stage in the malware detection system on the selected dataset, the samples were reduced to 30,000 samples (consisting of 12,000 malicious and 18,000 benign samples), and the 84 features in the dataset were minimized to 80 features. Following this, the feature selection stage was applied to the balanced dataset, which contained 60% benign samples and 40% malicious samples. A filter-based feature selection technique (CFS) was applied, selecting 57 features that had a high correlation to the target based on the determined threshold (0.2). The top 15 features are represented in Figure 3.

The supervised learning structure utilized the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), and a Stacking classifier. Table 1 presents the performance outcomes of the three methods and ensemble model under static analysis using static features for malware classification.

Table 2 shows the results from dynamic analysis using dynamic features.

Table 3 displays the outcomes of the three algorithms and ensemble model under hybrid analysis, which combines static and dynamic features, for malware classification.

Depending on the kind of feature analysis used with the same dataset, Android malware detection systems' efficacy can vary greatly. In this study, four machine learning algorithms – SVM, KNN, RF, and an ensemble stacking model – were used to examine the classification results from static, dynamic, and hybrid analyses. All models performed rather well in static analysis, which collects information without running the application. With an accuracy of 88.2% and an F1-score of 88.3%, the ensemble stacking classifier was the best in the group. Random Forest came in second with an accuracy of 87.1%. Static features (such as permissions and API calls) are useful in detecting established malware patterns, as evidenced by the reliable prediction findings and low computing cost. However, advanced obfuscation techniques that avoid static detection can make this method ineffective.

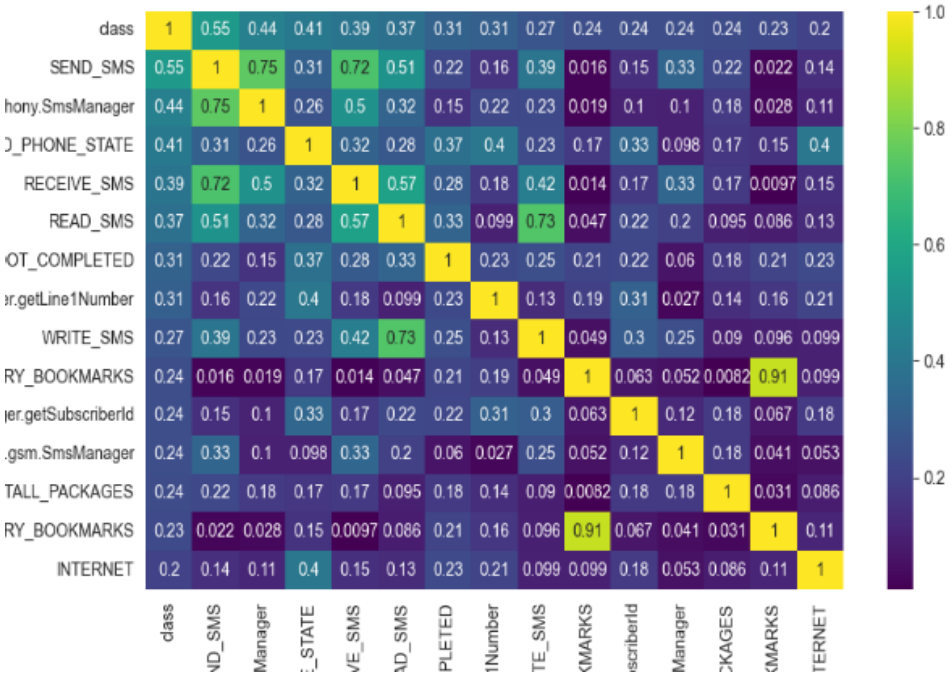

Figure 3: The top 15 features.

Table 1: Classification results based on static analysis.

Algorithm	Accuracy	Precision	Recall	F1-score	Time (sec)
SVM	86.2%	86.6%	84.9%	85.7%	1.10
KNN	85.5%	85.9%	84.6%	85.2%	0.70
RF	87.1%	86.5%	87.8%	87.1%	3.10
Stacking	88.2%	87.6%	89.1%	88.3%	9.20

Table 2: Classification results based on dynamic analysis.

Algorithm	Accuracy	Precision	Recall	F1-score	Time (sec)
SVM	79.0%	79.3%	77.8%	78.5%	1.00
KNN	78.1%	78.6%	77.2%	77.9%	0.80
RF	80.2%	79.6%	80.8%	80.2%	4.50
Stacking	81.0%	80.3%	81.7%	81.0%	12.30

Table 3: Classification results based on hybrid analysis.

Algorithm	Accuracy	Precision	Recall	F1-score	Time (sec)
SVM	97.9%	98.2%	96.6%	97.4%	7.80
KNN	96.2%	96.0%	94.5%	95.2%	5.44
RF	97.9%	97.5%	97.2%	97.3%	0.03
Stacking	98.9%	99.1%	98.3%	98.7%	36.46

Table 4: The comparison between the previous studies and proposed work.

Study	Dataset	Methods	Results
[8]	CICAndMal2017	(GRU)	98.2%
[11]	CICAndMal2017	LSTM-SVM	97%
[9]	CICAndMal2017	(Stacking)	90.4%
[12]	CICAndMal2017	(Stacking)	96.7%
[13]	CICAndMal2017	HGDetector model	97%
The proposed study	CICAndMal2017	(SVM), (KNN), (RF) and Stacking Ensemble Model	98.9%

KNN exhibited the quickest runtime (0.70 seconds) with just a slight performance degradation, indicating a trade-off between predictive power and efficiency. All measures showed a decline in model performance when relying only on dynamic aspects, such as runtime actions or system call behaviors. The F1-scores for each classifier decreased by roughly 7–8% compared to static analysis, and the ensemble model achieved the highest accuracy of 81.0%. This implies that dynamic analysis by itself might not provide enough context or consistency to enable extremely precise classification.

The reduced recall and precision point to possible challenges in extrapolating from sparse behavioral data or noise in execution environments. Performance was clearly best achieved by hybrid analysis that combined static and dynamic features. The ensemble stacking classifier outperformed both individual analysis approaches by achieving an accuracy of 98.9% and an F1-score of 98.7%. This illustrates how combining behavioral and structural information yields a more complete view of the application, improving classifiers' ability to distinguish between benign and malicious activity. The classification process took 36.46 seconds to complete, which would limit its applicability in real-time or resource-constrained contexts. With nearly similar performance and a remarkably short computation time (0.03 seconds), Random Forest likewise performed quite well in this environment, demonstrating its suitability for real-time detection applications. Thus, the best approach for high-stakes situations like mobile malware detection is hybrid analysis combined with ensemble models and a sizable dataset. This approach offers high accuracy and resistance to evasion strategies, but at a greater computational cost. Depending on the particular operational requirements, the best model selection should balance these trade-offs. The comparison between our proposed work and previous studies is shown in Table 4.

5 CONCLUSIONS

This study proposed a strategy for identifying Android malware and comparing the impact of using static, dynamic, and hybrid analysis with a stacking-based ensemble learning approach. The proposed model was trained with the CIC-AndMal-2017 dataset. The model first applied advanced preprocessing steps (converting data types to numeric, addressing missing values, etc.). It then performed a filter-based feature selection technique (CFS) and selected the top 15 features most correlated to the target and suitable for malware classification. Next, a 70% training set and 30% testing set were created from the dataset. Finally, the model classified malware using machine learning algorithms: Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), and a Stacking Ensemble Model. According to experimental results, using both static and dynamic data types improved Android malware detection performance compared to using either data type alone. This approach achieved 98.9% classification accuracy with the stacking classifier. In conclusion, combining both feature types with ensemble learning improved detection performance by 12% and 22% compared to static and dynamic approaches respectively.

ACKNOWLEDGMENTS

The authors thank the Department of Computer Science, College of Science, Al-Mustansiriyah University, for supporting this work.

REFERENCES

[1] M. Li, Z. Fang, J. Wang, L. Cheng, Q. Zeng, T. Yang, Y. Wu, and J. Geng, "A systematic overview of Android malware detection," Appl. Artif. Intell., vol. 36, no. 1, p. e2007327, 2022, [Online]. Available: https://doi.org/10.1080/08839514.2021.2007327.

[2] R. Srinivasan, S. Karpagam, M. Kavitha, and R. Kavitha, "An analysis of machine learning-based Android malware detection approaches," in Proc. Int. Conf. Electron. Circuits Signal. Technol., 2022.

[3] N. Jafaar and B. M. Nema, "Geolocation Android mobile phones using GSM/UMTS," Baghdad Sci. J., vol. 16, no. 1, Art. no. 34, 2019, doi: 10.21123/bsj.2019.16.1(Suppl.).0254.

[4] N. A. Sadkhan, Z. O. Ahmed, and R. N. Ajmi, "Assessing the potential of wild mushrooms as bioindicators for environmental pollution prediction using machine learning," Int. J. Des. Nat. Ecodyn., vol. 20, no. 2, pp. 439-446, Feb. 2025.

[5] B. AlKindy, O. B. Jamil, H. Al-Nayyef, and W. Alkendi, "A machine learning approach for identifying five types of horizontal ocular disorders using Haar features," Al-Mustansiriyah J. Sci., vol. 36, no. 1, pp. 69-83, Mar. 2025, doi: 10.23851/mjs.v36i1.1597.

[6] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," Digit. Investig., vol. 13, pp. 22-37, 2015.

[7] B. Amro, "Malware detection techniques for mobile devices," Int. J. Mob. Netw. Commun. Telemat., vol. 7, pp. 1-10, 2017.

[8] O. N. Elayan and A. M. Mustafa, "Android malware detection using deep learning," Procedia Comput. Sci., vol. 184, pp. 847-852, 2021.

[9] R. S. Arslan, "Identify type of Android malware with machine learning based ensemble model," in Proc. 5th Int. Symp. Multidiscip. Stud. Innov. Technol. (ISMSIT), Oct. 2021, pp. 628-632.

[10] T. Ball, "The concept of dynamic analysis," in *Software Engineering—ESEC/FSE'99*, Berlin, Germany: Springer, 1999, pp. 216-234.

[11] M. Gracea and M. Sughasiny, "Malware detection for Android application using Aquila optimizer and hybrid LSTM-SVM classifier," EAI Endorsed Trans. Scalable Inf. Syst., vol. 10, no. 1, 2022.

[12] Z. Xue, W. Niu, X. Ren, J. Li, X. Zhang, and R. Chen, "A stacking-based classification approach to Android malware using host-level encrypted traffic," J. Phys.: Conf. Ser., vol. 2024, no. 1, p. 012049, Sep. 2021.

[13] J. Feng, L. Shen, Z. Chen, Y. Lei, and H. Li, "HGDetector: A hybrid Android malware detection method using network traffic and function call graph," Alexandria Eng. J., vol. 114, pp. 30-45, 2025.

[14] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark Android malware datasets and classification," in Proc. Int. Carnahan Conf. Secur. Technol. (ICCST), 2018, pp. 1-7.

[15] "Number of smartphone and mobile phone users worldwide in 2020/2021: Demographics, statistics, predictions," [Online]. Available: https://www.unb.ca/cic/datasets/andmal2017.html, [Accessed: Jan. 11, 2020].

[16] E. S. Alomari, R. R. Nuiaa, Z. A. A. Alyasseri, H. J. Mohammed, N. S. Sani, M. I. Esa, and B. A. Musawi, "Malware detection using deep learning and correlation-based feature selection," Symmetry, vol. 15, p. 123, 2023, doi: 10.3390/sym15010123.

[17] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," IEEE Access, vol. 6, pp. 33789-33795, 2018.

[18] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson Education, 2020.

[19] R. RamaDevi and M. Abualkibash, "Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets - a review paper," Int. J. Comput. Sci. Inf. Technol., vol. 11, no. 3, pp. 65-80, 2019, doi: 10.5121/ijcsit.2019.11306.

[20] S. A. Salihu, S. O. Quadri, and O. C. Abikoye, "Performance evaluation of selected machine learning techniques for malware detection in Android devices," ILJCSIT, vol. 3, no. 1, 2020.