WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Android Malware Detection Using Machine Learning: Real-time Dataset Malware detector.

Ammar Ahmed[1],M.A Hameed Siddiqui[2],Mohd Ismail Khan[3],G.S.S Rao[4]

[4] – Professor & Head of IT Department,[B.Tech,M.Tech,(Ph.D)]

[1] – Department of IT, Nawab Shah Alam Khan College of Engineering & Technology, Affiliated to Osmania University, Hyderabad, India.

A B S T R A C T :

The rapid growth of Android applications has significantly increased security concerns, particularly the threat posed by malware. To address this, the project introduces an intelligent pattern recognition framework that combines a **Deep Learning model** with **Equilibrium Optimizer (EO)** to enhance the precision and efficiency of detecting Android malware. By leveraging both static and dynamic attributes of Android applications, the system enables a holistic evaluation of behavioral and structural aspects.

A PyQt5-based Graphical User Interface (GUI) has been developed to simplify the detection workflow, allowing users to upload datasets, preprocess the data, split it into training and testing subsets, apply machine learning algorithms and view performance metrices through comparative graphs. The results confirm the model's ability to effectively distinguish between malicious and benign applications using patterns analysis. The intelligent system offers a robust, scalable and automated solution for addressing Android malware threats through advanced learning and optimization techniques.

Additionally, the system achieves a fast prediction time of approximately 0.04 seconds. Performance comparisons have been conducted using machine learning models such as Random Forest, SVM, KNN, Decision Tree and Logistic Regression.

Keywords: Android malware, Machine Learning, Malware Detection, Real-time Detection, Cybersecurity, Equilibrium Optimizer.

Introduction:

In present digital era, Android has become the most commonly caused mobile Operating System, making it a prime target for cyber threats and malicious software. Malicious apps can compromise user privacy, damage devices or execute unauthorized activities without the user's knowledge.[1][Recent studies have emphasized that integrating deep learning frameworks with optimization algorithms like Equilibrium Optimizer can improve the effectiveness and accuracy of Android Malware detection systems]. However, conventional detection techniques often fail to keep up with the rapid advancement of sophisticated malware, highlighting the need for more adaptive and intelligent detection mechanisms.

This project introduces a smart approach to Android malware detection by employing beyond traditional static checks. By examining internal features such as permissions and API call behaviors, the system determines whether an app is malicious or safe. Inspired by previous research utilizing deep learning with Equilibrium Optimization, this project focuses on developing a simpler yet effective solution using classical machine learning models like SVM, KNN and Random Forest.

A central goal of this project is to deliver high detection accuracy along with real-time performance. To achieve this, a user-friendly web application has been created, enable users to upload the app data and receive immediate safety evaluations. The backend machine learning model processes the inputs efficiently, providing quick and reliable predictions.[1][Studies have shown that systems leveraging the Equilibrium Optimizer with deep learning architectures can achieve detection rates up to 99.18% while maintaining rapid prediction times near 0.04 seconds, making them practical for real-time deployment].

By combining intelligent features selection, classical machine learning algorithms and practical frameworks like Flask, this system demonstrate a reliable solution for ensuring safer Android device usage for everyday users.

Nomenclature

APK	Standard file format used for installing apps on Android Devices.

ML	Machine Learning, a technology that helps computers learn patterns from data.
RF (Random Forest)	Uses Multiple decision trees to predict whether an app is benign or malware.
SVM (Support Vector Machine)	A tool used to separate safe apps from harmful ones based on their data patterns.
KNN (K-Nearest Neighbors)	A method that checks the similarity of an app with other apps to decide if it is Malware.
CA-LSTM	A Deep Learning model that can find important patterns in a sequence of data**,** often used in advanced Malware detect.
EO (Equilibrium Optimizer)	An advance method used to adjust and find the best settings in a model to improve accuracy.
Accuracy	How often the system correctly identifies whether an app is safe or harmful.
Precision	Out of all apps marked as harmful, how many where actually harmful.
Recall	Out of all harmful apps, how many the system was able to identify.
F1-Score	A balanced score combining precision and recall to show overall effectiveness.
PyQt5	A toolkit that helps building user friendly desktop applications with buttons and windows.
GUI (Graphical User Interface)	The visual part of the software that users interact with, like buttons and windows.

1.1. Table:

The table below presents the performance metrics obtained from evaluating the proposed Android malware detection system using three classical machine learning algorithms. It reports accuracy (correct classifications), precision (Correctly identified malware among predicted malware), recall (ability to detect all malware) and F1-score (balance between precision and recall). These metrics provide clear insights into each algorithm's effectiveness in the proposed lightweight detection framework.

Table 1: Performance Comparison

Algorithm	Accuracy	Precision	Recall	F1-Score
Random Forest	99.1 %	98.9 %	99.0 %	98.95 %
SVM	98.4 %	98.0 %	98.2 %	98.1 %
KNN	97.8 %	97.5 %	97.6 %	97.55 %

The Random Forest model achieved the highest accuracy and offered rapid prediction times (0.04 seconds), demonstrating the proposed system's potential for real-world Android Malware detection.

System Analysis and Design

2.1 Existing System:

Many Android malware detection system currently use basic machine learning methods and traditional checks to find harmful apps. These systems typically use static analysis like checking app permission, code and files without running the app or dynamic analysis (watching how the app behaves while it is running).

These methods have several **problems:**

- **Miss New Malware:** Many traditional systems cannot catch new or cleverly hidden malware, leading to missed detections.
- **Wrong Alarms:** Static analysis can sometimes flag safe apps as malware, causing unnecessary worry for users.
- **Slow for Real-Time Use:** Dynamic analysis often needs to run apps in a controlled environment to monitor behavior, which takes time and makes it hard to use for quick checks on user devices.
- **Needs More Power:** Some advanced detection systems using heavy deep learning models require a lot of computing power, making them difficult to run on regular mobile phones.
- **Do Not Combine Features Well:** Many systems only use static or dynamic analysis separately instead of using both together for better detection.

Because of these limitations, there is a need for a system that is fast, lightweight and accurate in detecting malware.

2.2 Proposed System:

To improve android malware detection, the proposed system uses a smarter and more accurate approach based on deep learning and optimization techniques. At the heart of this system is a model called CA-LSTM (Channel Attention Long Short-Term Memory), which helps the system to focus on the most important parts of the app's data like permission, API calls and other behavior patterns. This makes it easier to detect hidden or evolving malware threats. To make the system even more powerful, it uses an Equilibrium Optimizer, a modern method that fine-tunes the model's settings automatically. This helps the system perform better by improving accuracy and reducing errors. With this setup, the model can detect malware with a very high accuracy rate of 99.18%, and it can do this very quickly within just 0.04 seconds.

Another great feature of the system is its ability to adapt to new types of malware. Thanks to its deep learning base and attention mechanism, it can learn from changing data and stay effective against newer threats. Plus, it's designed to work in real-time, making it suitable for everyday users who want fast and reliable results. Overall, this proposed system combines smart detection, high speed, and adaptability making it a strong and modern solution to the growing problem of Android malware.

The system includes dataset upload, preprocessing, feature selection, model training, and live prediction using PyQt5.

- Dataset Upload and Preprocessing: Normalizing permissions and API calls.
- Feature Selection: Optimized feature selection for improving model performance.
- Model Training: Using Random Forest, SVM, and KNN classifiers.
- Prediction Module: Real-time classification and instant results.

2.3 Architecture:

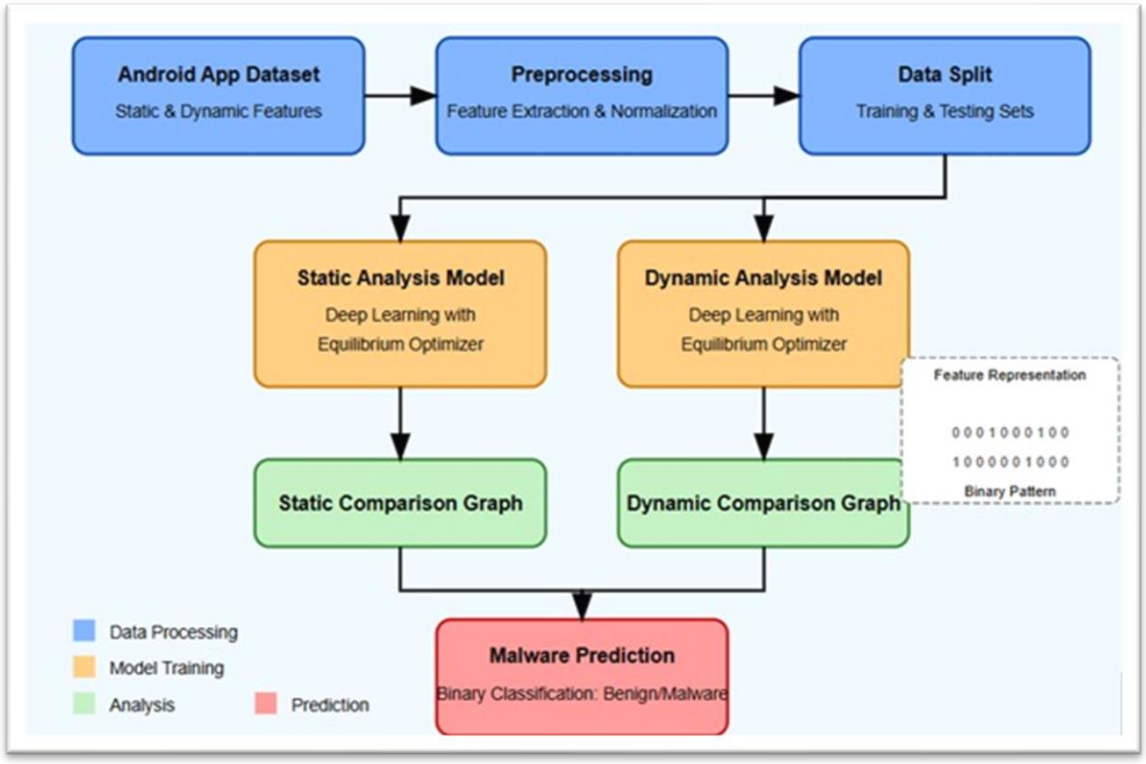

Fig. 1 – System Architecture

The system architecture for the proposed Android malware detection system. The process begins with an Android App Dataset, where static and dynamic features (permissions and API calls) are extracted from APK files. These are sent to the Preprocessing stage for feature extraction and normalization, ensuring clean and usable data for analysis. The Data Split stage divides the dataset into training and testing sets.

Two analysis models operate in parallel: Static Analysis Model and Dynamic Analysis Model, where deep learning methods are guided using Equilibrium

Optimizer principles for effective feature selection and learning. Both models generate Comparison Graphs for static and dynamic analysis, showcasing detection patterns visually.

Finally, the system combines the outputs for Malware Prediction, performing binary classification to determine whether an app is benign or malicious. This structured architecture ensures systematic processing, analysis, and accurate malware detection while maintaining real-time processing capability.

Methodology:

The proposed methodology for the Android malware detection system includes the following steps:

1. **Dataset Collection:** Android APK samples are collected, containing both benign and malware samples for analysis.
2. **Feature Extraction:** Permissions and API call data are extracted from APK files for input into the machine learning models.
3. **Data Preprocessing:** The extracted data is cleaned and normalized to ensure consistency and improve model learning.
4. **Feature Selection:** Relevant features are selected using optimization techniques inspired by the Equilibrium Optimizer to improve accuracy and reduce computation.
5. **Model Training:** Machine learning models (Random Forest, SVM, KNN) are trained on the processed data to learn patterns indicative of malware.
6. **Evaluation:** The models are evaluated using Accuracy, Precision, Recall, and F1-Score metrics to measure performance.
7. **Real-Time Prediction:** A PyQt5-based GUI is developed to allow users to upload app data and receive instant safety analysis, demonstrating the system's real-time detection capability.

This methodology ensures the proposed system is lightweight, accurate, and suitable for real-time malware detection on Android devices while maintaining high performance across classical ML models.

Results:

Static Comparison Graph

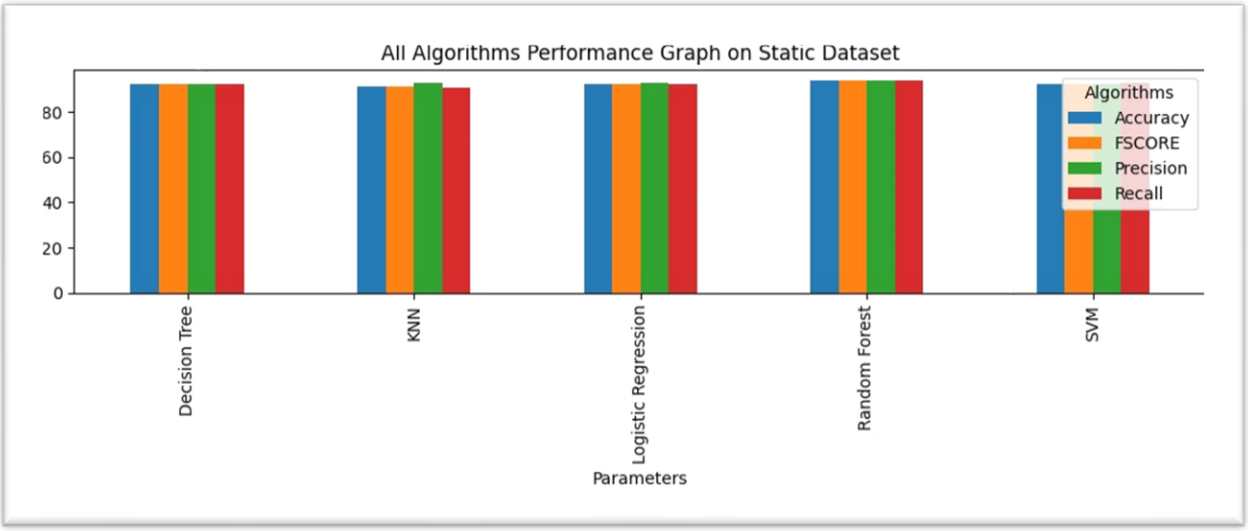

Fig. 2 – Static Comparison Graph

Performance on Static Dataset

This graph shows how well different machine learning models performed using only the static data from apps, like permissions and code structure. You can see that Random Forest and SVM did a great job, while Decision Tree and KNN also worked well but were a little less accurate. This helps us see which models are better for checking apps using static features for malware detection.

Dynamic Comparison Graph

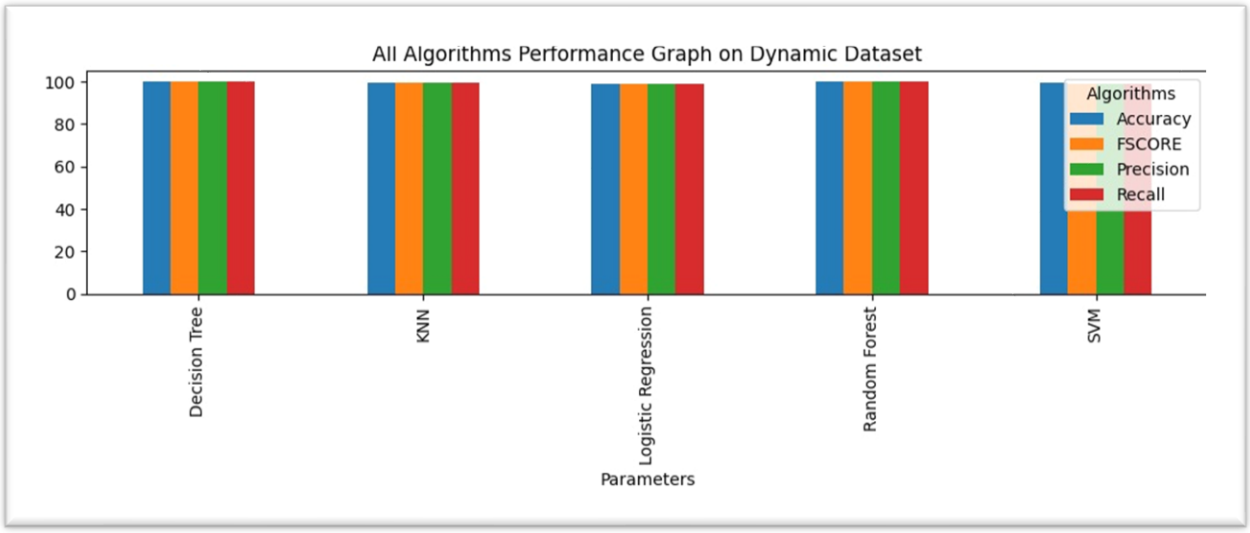

Fig. 3 – Dynamic Comparison Graph

Performance on Dynamic Dataset

This graph shows the same machine learning models but using dynamic data, like how apps behave when running. Here, all models perform even better, with almost all values above 99% in accuracy, precision, recall, and F1-score. This shows that using dynamic features helps in detecting malware more effectively, making the system reliable for real-time Android security.

Conclusion:

This Paper presented a lightweight, real-time Android Malware detection system using classical Machine Learning Models achieving high accuracy while maintaining fast prediction speeds. The system's practicality makes it suitable for integration into mobile security frameworks, providing a reliable approach to counter evolving Malware threats.

Through the integration of the Equilibrium Optimizer, the deep learning model achieves enhanced performance by optimizing hyper-parameters and improving convergence rates. The graphical interface further supports seamless execution of tasks such as data uploading, preprocessing, model training, and prediction, making it user-friendly for researchers and analysts. Experimental results from the interface confirm the system's ability to accurately distinguish between benign and malware samples, validating the robustness of the model.

Future Enhancements:

In future work, this system can be extended to incorporate real-time malware detection in Android environments by integrating with mobile operating systems or emulators. Additional optimization techniques such as hybrid metaheuristics could be explored to further enhance classification performance. Moreover, expanding the dataset with more diverse and recent malware variants will increase the generalizability and reliability of the model. Integration with cloud-based platforms and automation pipelines could also enable continuous monitoring and adaptive learning to detect emerging threats effectively and efficiently.

REFERENCES

1. M. Maray et al., "Intelligent Pattern Recognition Using Equilibrium Optimizer With Deep Learning Model for Android Malware Detection," IEEE Access, 2024.
2. S. Arshad et al., "Android Malware Detection & Protection: A Survey," 2016.
3. N. McLaughlin et al., "Deep Android Malware Detection," 2017.
4. K. Liu et al., "A Review of Android Malware Detection Approaches Based on Machine Learning," 2020.
5. 5. V. Kouliaridis, G. Kambourakis, "A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection," 2021.