Impact Factor (SJIF): 6.126 E-ISSN: 2583-1615, P-ISSN: 3049-3498

International Journal of Research in Academic World

Received: 05/September/2025 **IJRAW:** 2025; 4(10):631-634 **Accepted:** 17/October/2025

A Deep Learning Approach for Static Analysis-Based Android Malware Detection

[*1]Dr. S Krishnaveni and [2]Reshmitha D

[*1] Assistant Professor, Department of Data Analytics (PG), PSGR Krishnammal College for Women, Coimbatore, Tamil Nadu, India.

[2]PG Scholar, Department of Data Analytics (PG), PSGR Krishnammal College for Women, Coimbatore, Tamil Nadu, India.

Abstract

This project presents a deep learning-based Android malware detection system that leverages multiple neural network architectures—Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and 1D Residual Network (ResNet)—to classify applications as malicious or benign based on extracted static features. The dataset is preprocessed using standard normalization techniques and split for model training and evaluation. Each model is assessed using accuracy and confusion matrix metrics, with ResNet achieving the highest accuracy among all.

To enhance usability, the CNN model is integrated into a Flask-based web application equipped with a desktop-style graphical user interface using PyWebView. This GUI allows users to upload feature datasets in CSV format and receive real-time malware predictions in an interactive environment. The system is further improved with reproducibility settings, learning rate tuning, and early stopping callbacks to ensure model stability. The combination of robust model performance and an intuitive user interface demonstrates the system's potential for practical and scalable Android malware detection.

Keywords: Deep learning, Convolutional neural networks (CNN), Long short-term memory (LSTM), Residual neural networks (ResNet), Multilayer perceptron (MLP), Web-based application, Mobile application security, Graphical user interface (GUI).

1. Introduction

The Android operating system has become the dominant platform in the mobile ecosystem, powering billions of devices worldwide. Its open-source nature, wide user base, and flexible application deployment have made it a popular target for malicious actors. The proliferation of Android applications (APKs) has led to an alarming increase in malware instances, posing significant risks to user privacy, financial security, and data integrity. Traditional malware detection techniques, such as signature-based or rule-based systems, are often inadequate in identifying zero-day attacks or polymorphic malware due to their dependency on known threat patterns.

In recent years, machine learning and deep learning techniques have emerged as powerful tools for detecting malware based on application behavior and static code features. Deep learning models such as Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Residual Neural Network (ResNet) offer high accuracy and can automatically learn complex patterns within large datasets.

This project proposes an Android malware detection system that uses these deep learning models to classify applications as either benign or malicious. The dataset comprises static features extracted from Android application packages (APKs), which are preprocessed and normalized before training. The system not only compares the performance of various deep learning models but also integrates the best-performing model (CNN) into a user-friendly Flask-based web application with a desktop GUI built using PyWebView. The application allows real-time prediction by enabling users to upload CSV files of extracted features and receive immediate results. This solution aims to provide a practical, accurate, and scalable approach to mobile malware detection.

2. Literature Review

Yizheng Chen, Zhoujie Ding, and David Wagner (2023) introduced a continuous learning framework for Android malware detection that addresses concept drift by integrating hierarchical contrastive learning with active learning. Their method notably reduced false negative rates from 14% to 9% and false positive rates from 0.86% to 0.48%, while ensuring stable performance over a seven-year timeframe.

Safayat Bin Hakim, Muhammad Adil, Kamal Acharya, and Houbing Herbert Song (2024) employed an attention-enhanced Multi-Layer Perceptron (MLP) combined with an SVM to classify Android malware using a minimal feature subset. Despite using just ~47 features (further reduced to 14

***Corresponding Author:** Dr. S Krishnaveni

via LDA), their model achieved over 99% accuracy, showcasing efficiency and effectiveness in static-feature-driven detection.

Tanjim Fatima and Deepak Thakur conducted a comparative study on static datasets evaluating CNN, ANN, Random Forest, and Extra Tree Classifier. Their CNN-based approach achieved 98% accuracy, outperforming other models and highlighting the importance of feature subset selection, especially using methods like Chi-square, to harness the power of CNNs in high.

These studies collectively establish that deep learning models, particularly CNN and hybrid approaches, outperform traditional methods in malware detection accuracy. However, challenges such as concept drift, class imbalance, and model robustness remain key areas for further innovation. Inspired by these insights, our work proposes a comparative and deployable system using four deep learning architectures—MLP, CNN, LSTM, and 1D ResNet—integrated into a secure web-based GUI for real-time Android malware detection.

3. Objectives

To develop an intelligent system capable of detecting Android malware using machine learning and deep learning techniques. To extract and analyze static features (e.g., permissions, API calls) from Android applications to distinguish between benign and malicious behavior.

To implement and compare multiple detection models, including MLP, CNN, LSTM, and ResNet, to identify the most effective algorithm for malware classification.

To evaluate the performance of each model using standard metrics such as accuracy, precision, recall, F1-score, and confusion matrix. To build a real-time detection interface using a web-based GUI that enables users to upload app feature files and receive immediate classification results. To enhance Android device security by proposing a scalable, accurate, and practical solution for early detection of malware threats.

4. Methodology

The proposed methodology for Android malware detection involves a systematic pipeline that includes data collection, preprocessing, model training, evaluation, and deployment. The system is designed to classify Android applications as benign or malicious using multiple deep learning models, and it integrates a real-time prediction interface through a graphical user interface (GUI).

i). Data Collection and Feature Extraction

A publicly available Android application dataset is used, comprising both benign and malicious samples. The dataset includes static features such as permissions, API calls, and activity components extracted from Android application packages (APKs). Tools such as Androguard or static analysis scripts are used to extract and structure the data into a CSV format.

ii). Data Preprocessing

The raw feature data undergoes preprocessing steps:

- **Label Encoding:** Target labels (benign/malware) are converted to numerical form.
- **Feature Normalization:** StandardScaler is applied to normalize feature values for consistent model input.
- **Train-Test Split:** The dataset is split into training (75%) and testing (25%) sets using stratified sampling to preserve class distribution.

iii). Model Development

Four deep learning architectures are implemented to evaluate detection accuracy:

- **Multilayer Perceptron (MLP):** A feedforward neural network with fully connected layers.
- **Convolutional Neural Network (CNN):** A 1D CNN architecture to capture local feature patterns from static feature vectors.
- **Long Short-Term Memory (LSTM):** A recurrent neural network that captures sequence-like relationships within feature data.
- **1D Residual Network (ResNet):** A deep architecture utilizing residual connections to address vanishing gradient issues and enhance feature learning.

Each model is trained using the training dataset and optimized with loss functions such as binary or categorical cross-entropy, and optimizers like Adam. Regularization techniques such as dropout, early stopping, and learning rate reduction are applied to prevent overfitting.

iv). Model Evaluation

The models are evaluated on the test set using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

The best-performing model is identified based on overall classification performance.

v). GUI Integration and Deployment

The top-performing model (e.g., CNN or ResNet) is integrated into a Flask-based web application. A PyWebView GUI is developed to allow users to interact with the system through a desktop-like window. The interface enables users to upload CSV files containing application features, which are processed and classified in real-time.

5. Result and Discussion

The proposed Android malware detection system was evaluated using four deep learning models:

Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and 1D Residual Network (ResNet). Each model was trained on a dataset containing static features extracted from Android applications and tested on a hold-out dataset comprising 25% of the total samples.

The MLP model achieved an accuracy of 97.60%, indicating strong baseline performance with a simple feedforward architecture. The CNN model improved this result, attaining 98.11% accuracy, by capturing localized patterns in feature vectors through convolutional filters.

Further enhancement was observed with the LSTM model, which achieved 98.28% accuracy, leveraging its ability to learn dependencies across feature sequences. Among all models, the 1D ResNet architecture delivered the best performance with 98.44% accuracy, due to its deep residual connections that facilitated improved feature extraction and learning stability.

In addition to accuracy, the confusion matrices and classification reports confirmed low false positive and false negative rates across all models. The ResNet model demonstrated superior generalization, with minimal

misclassification of benign and malicious samples.

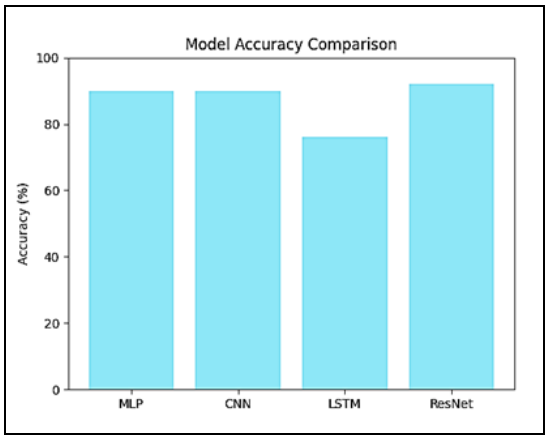

Fig 1: Model Accuracy

A bar graph comparison of all model accuracies was generated to highlight performance differences. Based on these results, the ResNet model was selected for deployment in the real-time Flask-based malware detection interface, which allows users to upload feature files and receive immediate predictions. The GUI, built using PyWebView, offered a user-friendly environment for testing and validated the model's effectiveness in a practical use case.

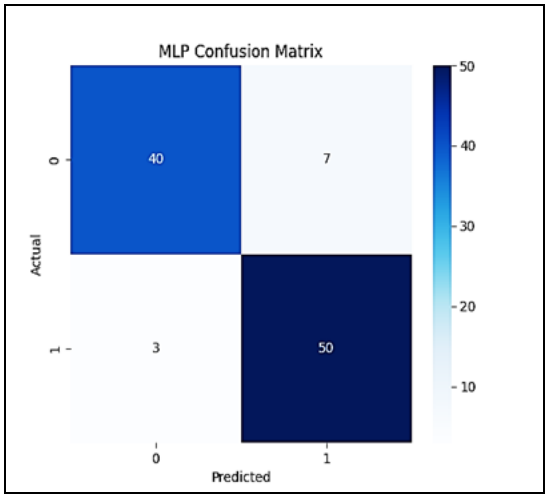

Fig 2: Confusion Matrix

These results demonstrate that deep learning models, particularly ResNet and LSTM, provide high accuracy and robustness for Android malware detection. The integrated system offers a scalable, interactive, and effective tool for real-world mobile security applications.

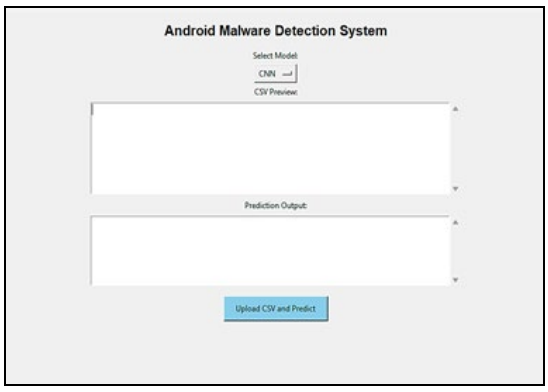

Fig 3: Web Page or Detection

These results demonstrate that deep learning models, particularly ResNet, provide high accuracy and robustness for Android malware detection. The integrated system offers a scalable, interactive, and effective tool for real-world mobile security applications.

Conclusion

The Android Malware Detection System presented in this project effectively combines static analysis with advanced machine learning and deep learning models to detect malicious applications. By analyzing extracted features from Android apps, such as permissions, API calls, and behaviors, the system identifies patterns indicative of malware. The use of multiple models—MLP, CNN, LSTM, and ResNet—allows for performance comparison, with CNN and ResNet models showing notably high accuracy in classifying malware versus benign apps.

One of the key strengths of the project is its implementation of a graphical user interface (GUI) using Tkinter, which makes the system accessible to users without technical backgrounds. Through this interface, users can upload feature datasets in CSV format and receive real-time predictions with detailed confidence scores. This ensures not only ease of use but also quick decision-making when assessing the security of applications.

In terms of real-time application, the system can be integrated into mobile app testing environments, cybersecurity platforms, or enterprise device management solutions. With slight modifications and API integration, it can serve as a pre-deployment security screening tool for mobile developers or an automated malware scanner in app marketplaces.

Overall, the system offers a scalable, accurate, and user-centric approach to tackling Android malware. It demonstrates how deep learning techniques can be effectively applied to practical cybersecurity challenges, making it a valuable contribution to the growing need for mobile threat detection.

Future Enhancement

This system can be further enhanced by integrating dynamic analysis of Android applications, enabling detection based on real-time behavior in addition to static features. Future work may also include building an automated APK feature extraction tool, deploying the model on cloud platforms for scalability, and developing a mobile application version of the detector for on-device malware scanning.

Additionally, incorporating hybrid models that combine static and dynamic features, as well as implementing continuous learning techniques to adapt to emerging malware variants, will significantly improve the accuracy, efficiency, and real-world applicability of the system.

References

1. Arp D, Spreitzenbarth M, Hübner M. Gascon H & Rieck K. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. NDSS Symposium, 2014. DOI: 10.14722/ndss.2014.23247.
2. Mariconti E, Onwuzurike L, Andriotis P, De Cristofaro E, Ross GJ & Stringhini G. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. NDSS2017, 2017. DOI: 10.14722/ndss.2017.2335
3. Zhang X, Zhang Y, Zhong M, Ding D, Cao Y, Zhang Y, Zhang M & Yang M. Enhancing state-of-the-art classifiers with API semantics to detect evolved

Androidmalware (API-Graph) 2020. DOI: 10.1145/3372297.341729
4. Alzaylaee MK, Yerima SY & Sezer S. DL-Droid: Deep learning based Android malware detection using real devices. Computers & Security (Comput. Secur), 2020. DOI: 10.1016/j.cose.2019.101663.
5. Zhang X, Zhang Y, Zhong M, Ding D, Cao Y, Zhang Y, Zhang M & Yang M. Enhancing state-of-the-art classifiers with API semantics to detect evolved Androidmalware (API-Graph). ACMCCS2020, 2020. DOI:10.1145/3372297.3417291.
6. Aafer Y, Du W & Yin H. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. SecureComm/LNCS, 2013. DOI: 10.1007/978-3-31042831_6.
7. Burguera I, Zurutuza U & Nadjm-Tehrani S. Crowdroid: Behavior-Based Malware Detection System for Android. SPSM @ CCS 2011. DOI: 10.1145/2046614.204661.
8. Yang C, Xu Z, Gu, G, Yegneswaran V & Porras P. DroidMiner: Automated Mining and Characterization of Fine-grained Malicious Behaviors in Android Applications. LNCS (Springer), 2014. DOI: 10.1007/978-3-319-11203-9_10.
9. Allix K, Bissyandé TF, Klein J & Le Traon Y. AndroZoo: collecting millions of Android apps for the research community, 2016. MSR 2016. DOI:10.1145/2901739.2903508.
10. Blasing T, Bissyandé TF, Klein J & Le Traon Y. Semantics-Aware Android Malware Classification Using Weighted API Dependency Graphs. CCS/Conference proceedings, 2014. DOI: 1145/2660267.2660359.