Migration Letters

Volume: 21, No: S14 (2024), pp. 647-660

ISSN: 1741-8984 (Print) ISSN: 1741-8992 (Online)

www.migrationletters.com

Hybrid Permission-Based Android Malware Detection Using Deep Learning-Enhanced Cnns And Xgboost

Atif Raza Zaidi[1], Tahir Abbas*[1], Sadaqat Ali Ramay[1], Iftikhar Ul Islam[2] and Muhammad Irfan[3]

Submission Date: 15/04/2024 Publication Date: 05/09/2024

Abstract:
Android, with a global mobile operating system market share of 71.17%, has become a primary target for malware attacks, leading to significant social issues such as privacy violations, financial losses, and psychological stress. This study provides insights into the global impact of Android malware, including country-specific attack statistics. While traditional machine learning algorithms have been extensively used for malware detection, their limitations in addressing the evolving complexity of Android malware emphasize the need for deep learning approaches. This research discusses permission-based detection methods and explores alternative models, evaluating their performance across diverse datasets. To address these challenges, a hybrid model is proposed, combining XGBoost for feature enhancement with Convolutional Neural Networks (CNNs) [1]for hierarchical learning. Implemented within a K-Fold cross-validation framework, the model achieves exceptional results, including an average accuracy of 94.23%, precision of 95.75%, recall of 92.41%, F1 score of 93.98%, and ROC AUC of 97.59%. A comparative analysis highlights the model's superiority over traditional machine learning algorithms such as Logistic Regression, Random Forest, Naive Bayes, and KNN in all key performance metrics. The findings demonstrate the potential of integrating feature enrichment with deep learning to develop robust and scalable solutions for Android malware detection.

Keywords: *Detection of Android Malware, Deep Learning, CNN, XGBoost*

1.0 Introduction:

In today’s world mobile devices are widely popular because of their portability, which allows people to stay connected and perform tasks effortlessly from any location unlike desktops and laptops that're more stationary, in nature. According to the recent data available (Statcounter 2024 - Desktop vs Mobile vs Tablet Market Share Worldwide) suggests that mobile devices are leading the global market with a share of 61.63% while desktop devices hold a lower share of 36.52% while tablets make up a smaller portion at 1.85%. Android leads the market share

[1]Department of Computer Science, TIMES Institute, Multan, 60000, Pakistan.

[2]NFC Institute of Engineering & Technology, Multan, 60000, Pakistan.

[3]National College of Business Administration and Economics, Multan campus, 60000, Pakistan.

*Corresponding Email: drtahirabbas@t.edu.pk

among Mobile Operating Systems worldwide with a substantial share of 71.17% (Statcounter 2024 - Mobile Operating System Market Share Worldwide) clearly illustrating its prominence in the industry. With the rise in Android phone users comes a growing concern about malware that poses a threat to their devices security due to the platforms popularity and open source structure making it vulnerable to attacks. Like computers, mobile devices also enable users to personalize and download applications. The rise, in Android app creation has surged, marked as one of the growing tech sectors. Nevertheless, this swift expansion brings along heightened security vulnerabilities that could impact both the applications and their users. It is important to note that in the quarter of 2023 there was an increase in the number of malicious software adware and riskware attacks, on mobile devices.

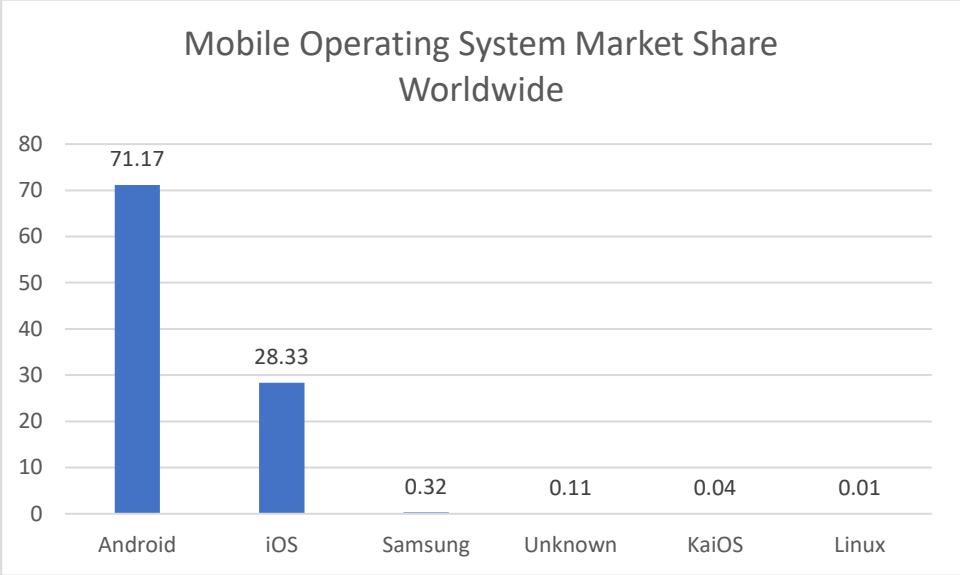

Figure: 1 Android Market Share in the world.

In 2024 according to Kaspersky Security Network (Kaspersky 2024 - IT threat evolution in Q1 2024. Mobile statistics)data analysis results showed that approximately 10 million security breaches were prevented on mobile devices due to malicious software such as malware and adware among others being blocked proactively by their systems. The prevalent issue was found to be adware accounting for 46% of all identified threats. Furthermore, a total of around 389000 software installations were uncovered with a focus, on mobile banking applications and ransomware.

But as more and more people use Android, there are chances of getting harmful software threats called malware. Malware has spread rapidly on Android due, to its use and the open nature of its development platform. This presents a security concern. (McAfee. McAfee mobile threat report (2021).). Android malware collects sensitive data (e.g., call logs, location), leading to privacy breaches that erode trust in mobile applications and services (Azad et al., 2020). Malware engages in fraudulent activities like stealing banking credentials or unauthorized transactions, causing economic stress and costly recovery efforts for victims (Laguerre, 2020). Persistent malware threats cause anxiety, stress, and distrust in mobile ecosystems, impacting users' digital participation and mental well-being (Woodward et al., 2020). Malware targeting critical sectors like defense poses national security risks, leading to geopolitical tensions and diminished public confidence in government systems (Falowo et al., 2024). The prevalence of malware undermines confidence in app stores and technology providers, reducing the adoption of beneficial digital solutions like e-government and m-health platforms (Adnyana et al., 2024).

Malware targeting educational or workplace apps disrupts learning and productivity, causing lost opportunities, stress, and frustration in remote settings.

Android viruses have a significant impact on at risk groups and worsen digital disparities and social problems. In developed areas, people with less access to safe gadgets and greater reliance on third-party app stores face a higher risk of malware threats, perpetuating the digital divide. (Razaghpanah et al., 2018). Older individuals with tech knowledge are often targeted by cyber threats and experience financial losses and stress while losing faith, in online platforms. Children and adolescents are also being approached through gaming and educational applications which may result in breaches of data privacy and exposure to content; this leaves guardians to deal with the aftermath. Such differences in demographics highlight the pressing requirement, for cybersecurity measures that're inclusive and awareness campaigns. According to Statista (Statista 2024 - Percentage of mobile users fallen victim 3rd quarter 2022), during the third-quarter of 2022, over 81% of all mobile users in Iran faced mobile malware attacks, making it the most affected country. Yemen ranked second with a 19% malware encounter rate among mobile users, followed by Saudi Arabia, where approximately 13% of mobile users were targeted.

Conventional malware detection methods are inherently difficult and uncertain (Farhat & Rammouz, 2021). Android malware detection using these techniques has developed as a result of the widespread application of deep learning and machine learning techniques in recent years (Iadarola et al., 2020), which has greatly improved malware detection systems' accuracy (Ni et al., 2018; Saracino et al., 2016). In response to the explosive increase of Android malware, a great deal of research has been conducted on techniques for deep learning-based Android malware detection (Levie et al., 2018). The surge, in Android malware has prompted research into the use of learning for detecting such threats (Levie et al., 2018). Researchers have explored a range of methods and study results involving machine and deep learning models (Ham & Choi, 2013; Mahindru & Sangal, 2021).

Our findings from the tests demonstrate that our innovative model outperforms machine learning techniques in terms of accuracy and performance measures. The next section outlines a Review of Literature in detail where we delve into strategies with a specific emphasis on studies related to detecting Android malware based on permissions and their constraints. In the section titled Proposed Model section we will delve into the specifics of our approach by outlining the structure and methodology in depth. This is followed by the Proposed Model section, where we describe the architecture and methodology of our approach in detail.

2.0 Literature Review:

The work of (Akbar et al., 2022) presents PerDRaML which is a permissions-based malware detection system for mobile devices. It focuses on Android due to its vulnerability from open-source policies, unofficial app stores, and lax app verification. It has used dataset of 10,000 apps. The system extracts feature like permissions and smali sizes to classify apps as malicious or benign with machine learning models like SVM and Random Forest. Achieving accuracies from this work are 86.25%-89.96%. It outperforms existing methods by optimizing features and improving metrics like precision and sensitivity. The paper (Şahin et al., 2023) introduces an Detection of Android Malware system which uses machine learning and linear regression-based feature selection to improve real-time detection. When reducing unnecessary features it enhanced training efficiency and classification performance. When it was tested on a dataset of 2,000 apps (1000were malicious & 1000 were benign), the system applied algorithms of Machine Learning like KNN, Naive Bayes, and Random Forest. It demonstrated improved performance through effective feature selection. The paper (Hein & Myo, 2018) presented a system for Android malware detection by analyzing the Manifest File and focusing on

Permission-Based Features. In this method of detection it has used a Score-based Approach to reduce feature dimensions achieving comparable performance to other methods while maintaining lightweight efficiency. By grouping applications according to the risks associated with permissions granted to them in the study examines algorithms to pinpoint dangerous permissions and enhance the accuracy of detecting malware. The findings, from experiments underscore the efficiency of classifiers that are based on permissions, in achieving detection rates underscoring the importance of exploring pertinent characteristics to enhance the detection of malware. The study (Arslan et al., 2019) examines mobile app security and privacy by analyzing spare permissions requested by apps for suspicious activities. It uses a combination of static and code analysis. It evaluates permission requests against actual usage to calculate risk levels. Machine learning algorithms were used with k-nearest neighbors which achieved the best classification accuracy of 91.95%. This integrated approach highlighted the effectiveness of combining static analysis with machine learning for app security. The paper (Amer, 2021) proposes a permission-based approach for Android malware analysis utilizing permission combinations declared in the manifest file of Android devices. The objective is to differentiate between malicious and benign apps by collecting commonly requested permission combinations by malware and benign apps. An ensemble model was developed which outperformed individual classifiers in terms of accuracy. The research (Kapoor et al., 2019) addresses the growing threat of Android malware, highlighting the need for effective detection methods due to the platform's open-source nature. Using a dataset of 4000 Android apps, permissions were extracted from manifest files and processed into a CSV format. Machine learning algorithms were applied to classify apps as malicious or benign, with training and testing conducted on the dataset to evaluate detection accuracy. The study (Lubuva et al., 2019) reviewed 56 research papers on static malware detection for Android apps, emphasizing its pre-installation advantage over dynamic analysis, which risks exposing devices to malicious apps. Key aspects included permission misuse, heuristic models, and reverse engineering using tools like JD-GUI and APKTOOL to analyze .apk files and extract critical data such as AndroidManifest.xml. Although static analysis is still important for spotting and countering threats effectively the assessment pointed out gaps in dealing with changing malware threats and suggested more research to improve the precision and effectiveness of detection approaches. The use of static analysis methods is vital, for protecting Android users. The paper (Ghasempour et al., 2020) proposes a multi-level permission extraction framework for Android malware detection, addressing challenges in feature extraction and classification due to increasing malware complexity. Analyzing Android APK files through static analysis techniques enhances detection accuracy by utilizing permissions as factors, in the process. The classification is carried out using Support Vector Machine (SVM) and Decision Tree algorithms which have shown promising outcomes in maintaining accuracy levels even when dealing with extensive datasets.

Though the use of permission-based techniques has been extensively studied for detecting Android malware the literature has also delved into other methods. In Table 1 we will find an overview of research that has examined approaches showcasing their analysis methods, models, for learning, data sets used and the outcomes achieved in terms of performance.

Table 1. Literature Review of Previous work

Related work	**Year**	**Analysis Method**	**Learning Model**	**Dataset**	**Performance Results**
(Lashkari et al., 2018)	2018	Dynamic Analysis	RF, KNN, DT	CICAndMal2017 (Android Malware Dataset (CIC-AndMal2017))	Precision: 85%

(Taheri et al., 2019)	2019	Static & Dynamic Analysis	RF	CICAndMal2017, InvesAndMal2019 (Android Malware Dataset (CIC-AndMal2017); Android Malware Dataset (CIC-InvesAndMal2019))	Accuracy: 83.3%
(Noorbehbahani et al., 2019)	2019	Dynamic Analysis	DT, RF, KNN, SVM, NB	CICAndMal2017 (Android Malware Dataset (CIC-AndMal2017))	Accuracy: 82.80%
(Mahindru & Sangal, 2019)	2019	Static Analysis	DBN	Google Play (Google Play Store.)	Accuracy: 94%
(Imtiaz et al., 2021)	2020	Static & Dynamic Analysis	ANN	AMD	Static Layer: 93.4%, Dynamic Layer: 80.3%
(Zhu et al., 2020)	2021	Static Analysis	MLP, SVM	SEDMDroid (Zhu et al., 2020)	Accuracy: 89.07%
(Kim et al., 2022)	2022	Static Analysis	Lightweight CNN	Google Play (Google Play Store.), Virus Share (VirusShare), AMD	Accuracy: 91.27%
(Yilmaz et al., 2022)	2022	Static Analysis	NB, SVM	AMD	Accuracy: 92.4%

3.0 Proposed Methodology:

3.1 Dataset

The dataset (Lopez & Cadavid, 2016) used in this study comprises 199 malware samples and 199 benign samples, totaling 398 records, each with 331 features. This study utilizes a permission-based dataset, leveraging permissions extracted from Android applications as key features for malware detection. The focus on permissions provides an interpretable and effective approach for distinguishing malicious apps from benign ones. The Random Forest method was used to pinpoint the 20 vital features in the analysis process; among them were permissions like READ_PHONE_STATE and READ_SMS which held the highest ranking positions in terms of importance level. Feature importance metrics offer insights into how permissions play a crucial role, in detecting malware effectively.

Table 2. Top 20 Dataset Features by Importance

Sno	Feature	Weight
1	android. permission . READ_PHONE_STATE	0.167714
2	android. permission . READ_SMS	0.10376
3	android. permission . WRITE_SMS	0.07496
4	android. permission . ACCESS_WIFI_STATE	0.073909
5	android. permission . CHANGE_WIFI_STATE	0.053314
6	android. permission . ACCESS_NETWORK_STATE	0.048563

7	android. permission . INTERNET	0.045845
8	android. permission . INSTALL_PACKAGES	0.030443
9	android. permission . SEND_SMS	0.030433
10	android. permission . WRITE_EXTERNAL_STORAGE	0.024783
11	android. permission . RECEIVE_BOOT_COMPLETED	0.024695
12	android. permission . GET_TASKS	0.022108
13	android. permission . ACCESS_COARSE_LOCATION	0.018934
14	android. permission . RECEIVE_SMS	0.018304
15	android. permission . WRITE_APN_SETTINGS	0.016528
16	android. permission . READ_CONTACTS	0.016086
17	android. permission . WRITE_CONTACTS	0.016083
18	android. permission . WAKE_LOCK	0.01544
19	android. permission . CALL_PHONE	0.015111
20	android. permission . RESTART_PACKAGES	0.014794

The top 20 features extracted from the permission-based dataset, ranked by their importance, are summarized in Table 2. A visual representation of these features and their corresponding importance scores is displayed in the bar graph in Fig. 2:

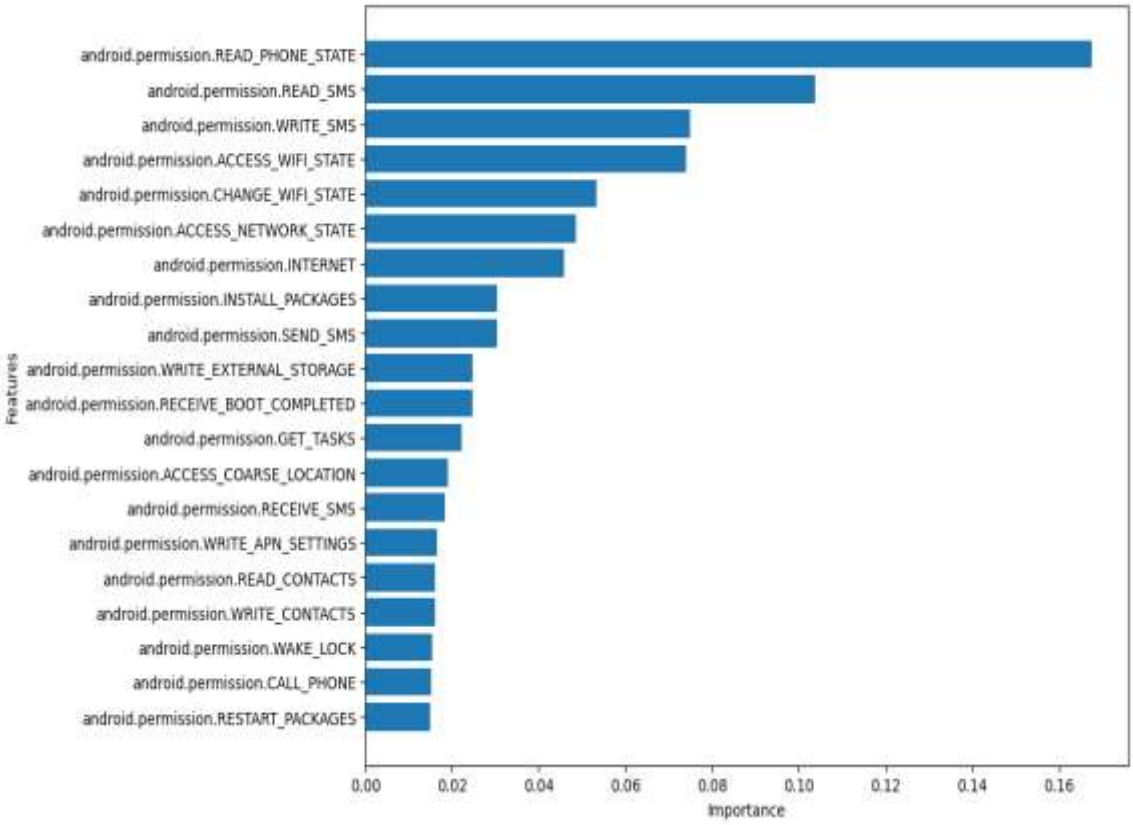

Figure: 2 Visual representation of top 20 Dataset Features by Importance

3.2 Proposed Hybrid Model Framework

Model combines the use of XGBoost and a Convolutional Neural Network (CNN) within a cross-validation framework to ensure a robust evaluation of a predictive model trained on a dataset. Features are scaled using a StandardScaler to normalize the data, enhancing the performance and stability of both the XGBoost and CNN models. XGBoost is then trained on the scaled features to generate probability outputs, which are concatenated back with the original features to serve as enriched input for the CNN.

Algorithm 1 Hybrid Approach: XGBoost + CNN for Android Malware Detection

1. Load the Android permission dataset: $D = \{X, y\}$, where X represents features and y represents labels.
2. Scale the dataset features using StandardScaler:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma},$$

where μ is the mean and σ is the standard deviation of X.

3. Train an XGBoost model on the scaled features and obtain probability outputs:

$$P_{\text{XGB}} = \text{XGBoost}(X_{\text{scaled}}),$$

where $P_{\text{XGB}} \in [0, 1]$ represents the predicted probabilities of the malware classification.

4. Concatenate the original scaled features with the XGBoost probability outputs:

$$X_{\text{enhanced}} = [X_{\text{scaled}}, P_{\text{XGB}}].$$

5. Perform K-fold cross-validation (split X_{enhanced} into $K = 6$ parts):
6. **for** each fold k in K-fold cross-validation **do**
7. Train a Convolutional Neural Network (CNN) model on the training fold:
   - Define the CNN model with convolutional layers, max-pooling, and dense layers.
   - Compile the model with performance metrics: accuracy (A) and AUC (AUC).
   - Train the CNN on the training data $X_{\text{train}}^{(k)}$ with corresponding labels $y_{\text{train}}^{(k)}$.
8. Evaluate the CNN model on the corresponding validation fold:
   - Predict malware classifications for the validation set:

$$\hat{y}_{\text{val}}^{(k)} = \text{CNN}(X_{\text{val}}^{(k)}).$$

   - Compute performance metrics for fold k:

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Samples}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{False Positives (FP)}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{False Negatives (FN)}}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

9. Collect and store performance metrics for the current fold.
10. **end for**
11. Compute the average performance metrics across all folds:

$$\bar{M} = \frac{1}{K} \sum_{k=1}^{K} M^{(k)},$$

where $M^{(k)}$ represents the performance metric (accuracy, precision, recall, F1 score, or ROC AUC) for fold k.

Figure: 3 Algorithm of proposed model

The enriched input undergoes K-fold cross-validation, where it is split into six parts to ensure the model is not biased towards any particular segment of the data. Each fold is used once as a validation set while the rest serve as training data. This approach mitigates overfitting and ensures that the model's performance is consistent across different subsets of the dataset.

The CNN architecture, designed for sequence data, consists of convolutional layers followed by max pooling and dense layers with dropout for regularization. The model is built with precision and area under the curve (AUC), as measures. It is trained using stopping to avoid overfitting by considering validation loss as a factor. After the training process it concludes with predictions being generated for the validation set by assessing performance indicators such, as accuracy rate and ROC AUC alongside evaluating precision, recall, F measure and AUC of the operating characteristic (ROC). These metrics are averaged across all subsections to deliver an assessment of how the model performs.

3.3 System Specifications

The implementation and the performance evaluation of the proposed model was carried on the system specifications whose details are appended below in a table. The corresponding results are presented and analyzed in the next section.

Table 3. System Specifications

Processor	Core i7 (11th Gen)
RAM	40GB DDR-4 (8GB + 32GB)
Storage	512GB NVME SSD PCIE 4x4
GPU	2GB MX 450 GDDR6
Operating System	Windows 10 64-Bit
Python Version	Python 3.7 Environment (via Anaconda Jupyter Notebook)
Libraries	TensorFlow, matplotlib, and XGBoost

4.0 Results & Analysis:

4.1 Model Performance Metrics

The table 4 below describes the average values of performance metrics such as accuracy, precision, recall, F1 score, and ROC AUC, which are being calculated across all folds

Table 4. Performance Metrics

Metric	**Value**
Average Accuracy	0.9423337856173678
Average Precision	0.9575471224178119
Average Recall	0.9241073687539533
Average F1 Score	0.9397528224434577
Average ROC AUC	0.9759126598138798

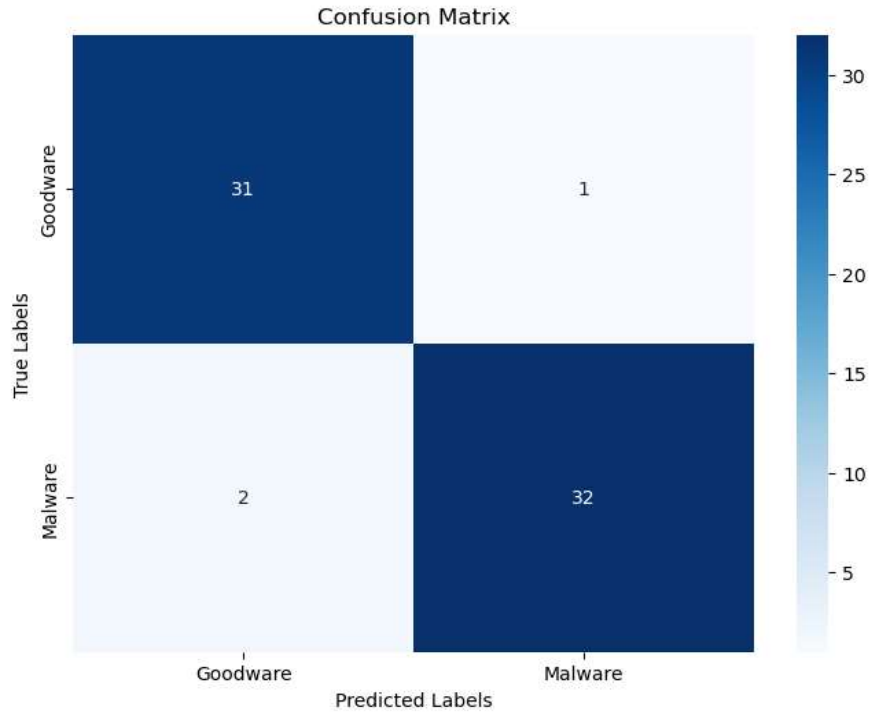

Figure: 4 Confusion Matrix of the proposed model

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{32 + 31}{32 + 31 + 1 + 2} = \frac{63}{66} \approx 0.9545$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{32}{32 + 1} = \frac{32}{33} \approx 0.9697$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{32}{32 + 2} = \frac{32}{34} \approx 0.9412$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.9697 \cdot 0.9412}{0.9697 + 0.9412} \approx 0.9553$$

Where:

True Positives (TP) = 32 (it has correctly predicted malware cases)

True Negatives (TN) = 31 (it has correctly predicted benign cases)

False Positives (FP) = 1 (Benign which were misclassified as malware)

False Negatives (FN) = 2 (Malware which were misclassified as benign)

The training and validation accuracy for the final fold of the K-Fold Cross-Validation process are illustrated in the figure below. It demonstrates the accuracy results, for the final fold round of the K Fold Cross Validation method used in training and evaluating the models performance over time in the fold.

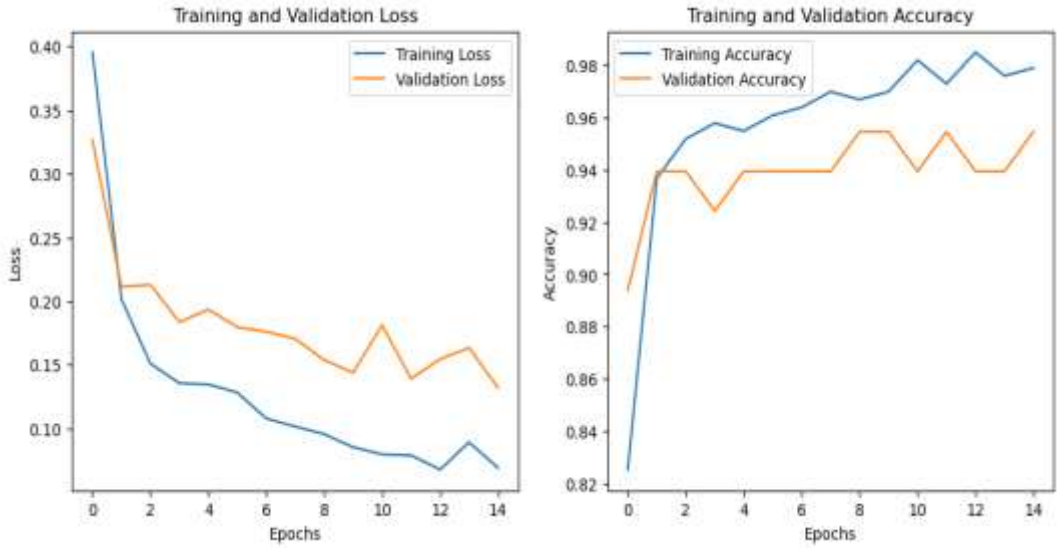

Figure: 5 Training and validation accuracy for the final fold of the K-Fold Cross-Validation

4.2 Comparative Analysis of proposed methodology with Machine Learning algorithms

The table below presents the performance metrics of various machine learning algorithms evaluated on the same dataset where our proposed model has been trained. These metrics provide a comparative analysis of the effectiveness of different algorithms in terms of accuracy, precision, recall, F1 score, and ROC AUC.

Table 5. Performance Metrics of Machine Learning on same dataset

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Logistic Regression	0.908333	0.910178	0.912587	0.908276	0.96
Random Forest	0.891667	0.893493	0.895804	0.891599	0.958042
Naive Bayes	0.775	0.79	0.783916	0.774609	0.903916
KNN	0.866667	0.872727	0.872727	0.866667	0.924336

We present a comparison chart illustrating the performance of the machine learning algorithms alongside our proposed model. The chart provides a metric-wise comparison, offering insights into how the proposed model outperforms or aligns with other algorithms in terms of key performance indicators.

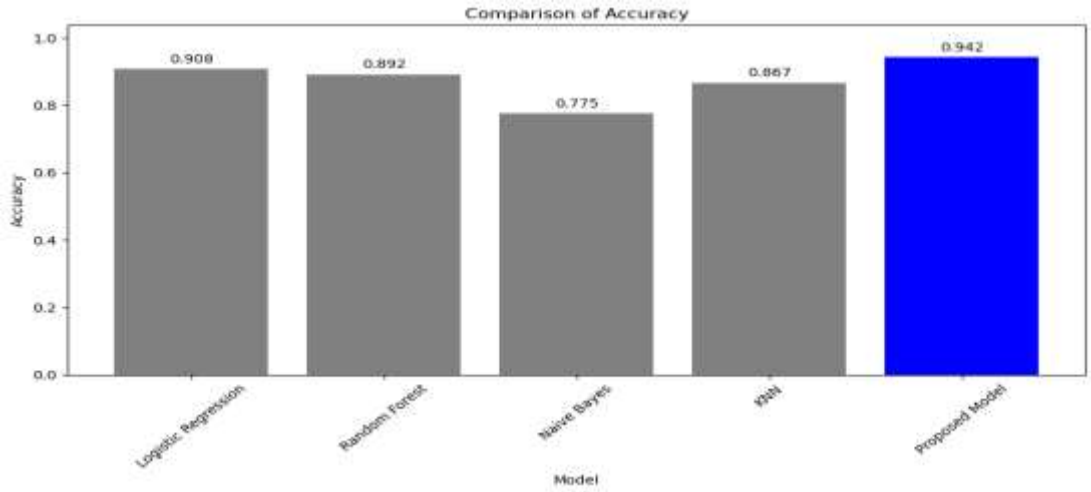

Figure: 6 Comparison of Accuracy Metric

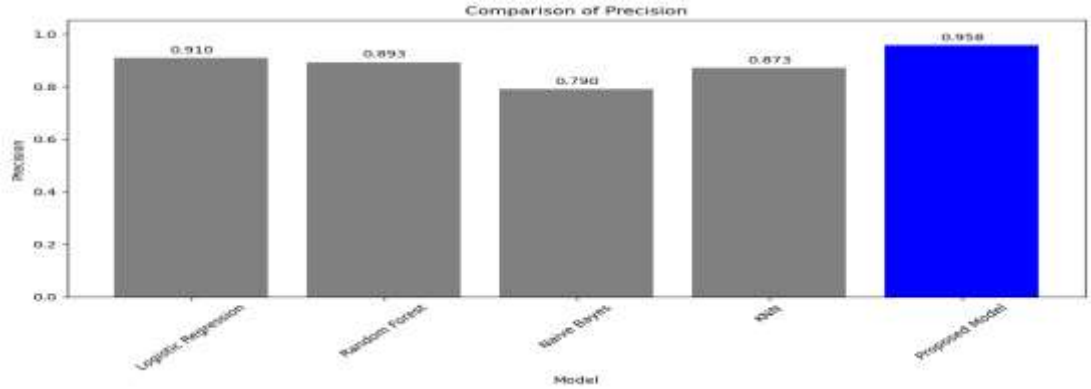

Figure: 7 Comparison of Precision Metric

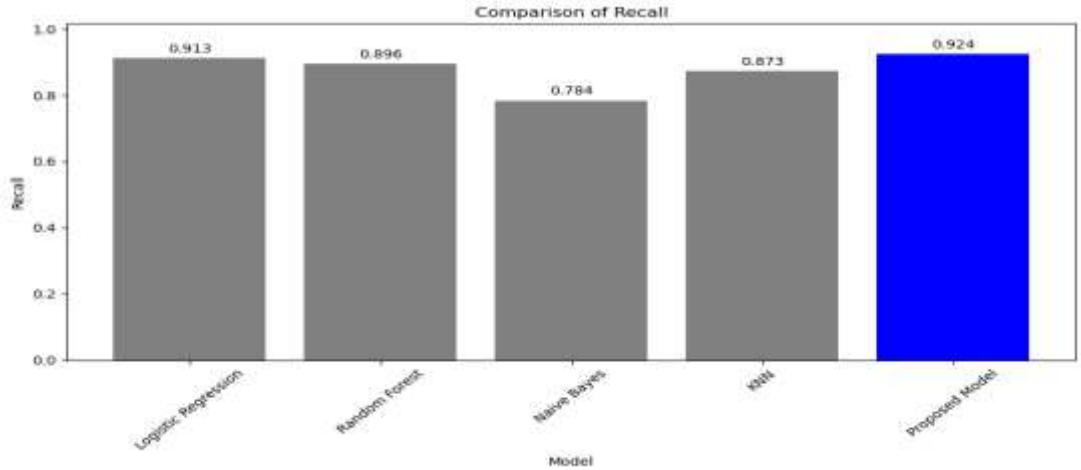

Figure: 8 Comparison of Recall Metric

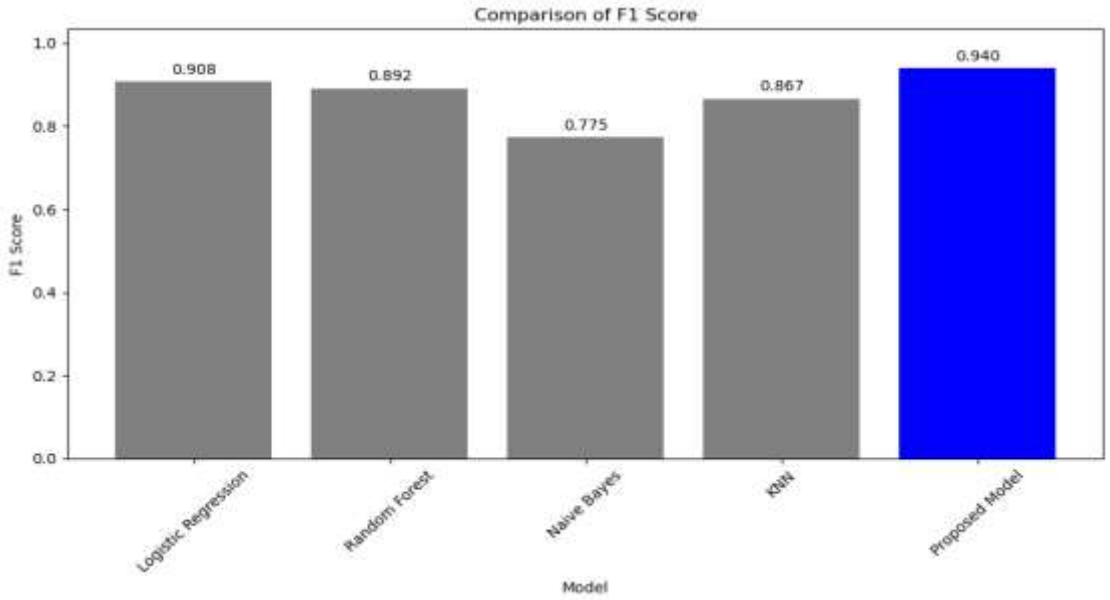

Figure: 9 Comparison of F1 Metric

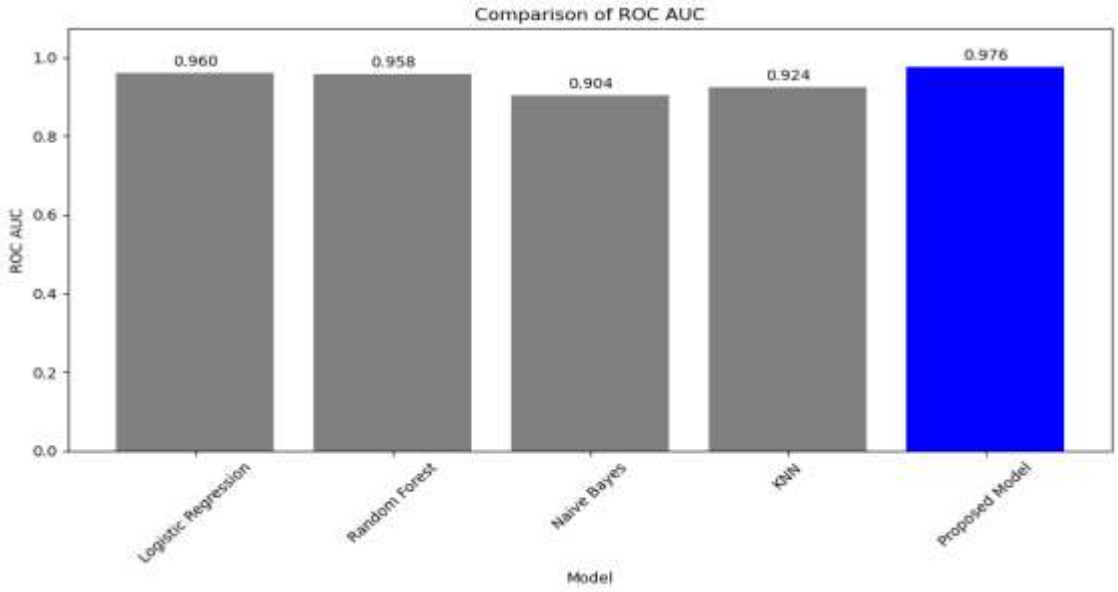

Figure: 10 Comparison of ROC AUC Metric

5.0 Conclusion:

The study findings show how effective the new hybrid approach is, for detecting Android malware by combining XGBoost with a Convolutional Neural Network (CNN). The model achieved an accuracy of 94% too. It also had precision at 95% recall at 92% an F1 score of about 94% and an ROC AUC of nearly 98%. Consistently outperforming machine learning methods like Logistic Regression and Random Forest, on the dataset. The combination method utilized the power of XGBoost to enhance features and CNN for extracting features effectively resulting in detection accuracy and adaptability overall.

This research highlights the significance of combining feature improvement and sophisticated deep learning structures for identifying malware. Although the suggested approach provides an dependable solution future research could investigate datasets, various types of features (such, as dynamic or behavioral) and optimization techniques to improve its effectiveness and scalability even further.

References

1. Adnyana, I. G. A., Nugraha, P. G. S. C., & Nugroho, B. R. A. (2024). Reverse Engineering for Static Analysis of Android Malware in Instant Messaging Apps. Journal of Computer Networks, Architecture and High Performance Computing, 6(3), 1460-1469.
2. Akbar, F., Hussain, M., Mumtaz, R., Riaz, Q., Wahab, A. W. A., & Jung, K.-H. (2022). Permissions-based detection of android malware using machine learning. Symmetry, 14(4), 718.
3. Amer, E. (2021). Permission-based approach for android malware analysis through ensemble-based voting model. 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC),
4. Android Malware Dataset (CIC-AndMal2017). https://www.unb.ca/cic/datasets/andmal2017.html
5. Android Malware Dataset (CIC-InvesAndMal2019). https://www.unb.ca/cic/datasets/invesandmal2019.html
6. Arslan, R. S., Doğru, İ. A., & Barişçi, N. (2019). Permission-based malware detection system for android using machine learning techniques. International journal of software engineering and knowledge engineering, 29(01), 43-61.
7. Azad, M. A., Arshad, J., Akmal, S. M. A., Riaz, F., Abdullah, S., Imran, M., & Ahmad, F. (2020). A first look at privacy analysis of COVID-19 contact-tracing mobile applications. IEEE internet of things journal, 8(21), 15796-15806.
8. Falowo, O. I., Ozer, M., Li, C., & Abdo, J. B. (2024). Evolving Malware & DDoS Attacks: Decadal Longitudinal Study. IEEE Access.
9. Farhat, H., & Rammouz, V. (2021). Malware classification using transfer learning. arXiv preprint arXiv:2107.13743.
10. Ghasempour, A., Sani, N. F. M., & Ovye, J. A. (2020). Permission extraction framework for android malware detection. International Journal of Advanced Computer Science and Applications, 11(11).
11. Google Play Store. . https://play.google.com/store/apps
12. Ham, H.-S., & Choi, M.-J. (2013). Analysis of android malware detection performance using machine learning classifiers. 2013 international conference on ICT Convergence (ICTC),
13. Hein, C., & Myo, K. M. (2018). Permission-based feature selection for android malware detection and analysis. International Journal of Computer Applications, 181(19), 29-39.
14. Iadarola, G., Martinelli, F., Mercaldo, F., & Santone, A. (2020). Call graph and model checking for fine-grained android malicious behaviour detection. Applied Sciences, 10(22), 7975.
15. Imtiaz, S. I., ur Rehman, S., Javed, A. R., Jalil, Z., Liu, X., & Alnumay, W. S. (2021). DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network. Future Generation computer systems, 115, 844-856. https://doi.org/https://doi.org/10.1016/j.future.2020.10.008
16. Kapoor, A., Kushwaha, H., & Gandotra, E. (2019). Permission based android malicious application detection using machine learning. 2019 International Conference on Signal Processing and Communication (ICSC),
17. Kaspersky 2024 - IT threat evolution in Q1 2024. Mobile statistics. Retrieved 01 Jul 2024 from https://securelist.com/it-threat-evolution-q1-2024-mobile-statistics/112750/
18. Kim, J., Ban, Y., Ko, E., Cho, H., & Yi, J. H. (2022). MAPAS: a practical deep learning-based android malware detection system. International Journal of Information Security, 21(4), 725-738. https://doi.org/https://doi.org/10.1007/s10207-022-00579-6
19. Laguerre, C. (2020). Social Engineering Strategies within the Financial Industry through Online Banking Utica College].
20. Lashkari, A. H., Kadir, A. F. A., Taheri, L., & Ghorbani, A. A. (2018). Toward developing a systematic approach to generate benchmark android malware datasets and classification. 2018 International Carnahan conference on security technology (ICCST),

21. Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 67(1), 97-109.
22. Lopez, C. C. U., & Cadavid, A. N. (2016). Machine learning classifiers for android malware analysis. 2016 IEEE Colombian Conference on Communications and Computing (COLCOM),
23. Lubuva, H., Huang, Q., & Msonde, G. C. (2019). A review of static malware detection for Android apps permission based on deep learning. International Journal of Computer Networks and Applications, 6(5), 80-91.
24. Mahindru, A., & Sangal, A. (2019). Deepdroid: feature selection approach to detect android malware using deep learning. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS),
25. Mahindru, A., & Sangal, A. L. (2021). MLDroid—framework for Android malware detection using machine learning techniques. Neural Computing and Applications, 33(10), 5183-5240.
26. McAfee. McAfee mobile threat report (2021). https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf
27. Ni, S., Qian, Q., & Zhang, R. (2018). Malware identification using visualization images and deep learning. Computers & Security, 77, 871-885.
28. Noorbehbahani, F., Rasouli, F., & Saberi, M. (2019). Analysis of machine learning techniques for ransomware detection. 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC),
29. Razaghpanah, A., Nithyanand, R., Vallina-Rodriguez, N., Sundaresan, S., Allman, M., Kreibich, C., & Gill, P. (2018). Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. The 25th annual network and distributed system security symposium (NDSS 2018),
30. Şahin, D. Ö., Kural, O. E., Akleylek, S., & Kılıç, E. (2023). A novel permission-based Android malware detection system using feature selection based on linear regression. Neural Computing and Applications, 1-16.
31. Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). Madam: Effective and efficient behavior-based android malware detection and prevention. IEEE Transactions on Dependable and Secure Computing, 15(1), 83-97.
32. Statcounter 2024 - Desktop vs Mobile vs Tablet Market Share Worldwide. https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet
33. Statcounter 2024 - Mobile Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide
34. Statista 2024 - Percentage of mobile users fallen victim 3rd quarter 2022. https://www.statista.com/statistics/325201/countries-share-of-malicious-attacks/
35. Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019). Extensible android malware detection and family classification using network-flows and API-calls. 2019 International Carnahan Conference on Security Technology (ICCST),
36. VirusShare. https://virusshare.com/
37. Woodward, K., Kanjo, E., Brown, D. J., McGinnity, T. M., Inkster, B., Macintyre, D. J., & Tsanas, A. (2020). Beyond mobile apps: a survey of technologies for mental well-being. IEEE Transactions on Affective Computing, 13(3), 1216-1235.
38. Yilmaz, A. B., Taspinar, Y. S., & Koklu, M. (2022). Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms. International Journal of Intelligent Systems and Applications in Engineering, 10(2), 269-274. https://doi.org/https://www.ijisae.org/index.php/IJISAE/article/view/2010
39. Zhu, H., Li, Y., Li, R., Li, J., You, Z., & Song, H. (2020). SEDMDroid: An enhanced stacking ensemble framework for Android malware detection. IEEE Transactions on Network Science and Engineering, 8(2), 984-994. https://doi.org/doi.org/10.1109/TNSE.2020.2996379