World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Research Article)

Enhancing android mobile security through machine learning-based malware detection using behavioral system features

Md Reduanur Rahman [1, *], Nasrin Akter Tohfa [2], Md Habibul Arif [3], Sufia Zareen [4], Md Abdul Alim [5], Md Shakhawat Hossen [6], Nayem Uddin Prince [7] and Touhid Bhuiyan [8]

[1] *Student, Master's in Information Technology, Washington University of Science and Technology, Virginia, USA.*
[2] *Student, Master's in Information System Security, University of the Cumberlands, KY, USA.*
[3] *Student, Doctor of Computer Science, University of the Potomac, USA.*
[4] *Student, Master's in Information Technology and Management, Campbellsville University, USA.*
[5] *Student, Master's in Information Technology in Management, St. Francis College, Brooklyn, NY, USA.*
[6] *Student, Master's in Information Technology, Washington University of Science and Technology, Virginia, USA.*
[7] *Student, Master's in Information Technology, Washington University of Science and Technology, Virginia, USA.*
[8] *Professor, Professor of Cybersecurity, Washington University of Science and Technology, Virginia, USA.*

World Journal of Advanced Research and Reviews, 2025, 28(01), 2308-2315

Publication history: Received on 21 September 2025; revised on 24 October 2025; accepted on 28 October 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.28.1.3686

Abstract

Currently, with the explosive surge of Android applications, it has become much harder to preserve security for mobile devices since malicious applications still advance and spread by more advanced evasion tactics. Signature-based malware detection approaches are no longer effective for such evolutionary threats. In this paper, a Malware Detection Dataset (MDD) dataset used to integrate system calls and binder frequencies as feature vectors of traces to enhance Android mobile security by a machine learning-based malware detection framework. The proposed methodology consists of a systematic data pre-processing feature scaling, class distribution analysis strategy, and two deep learning architectures based on dense neural networks developed and evaluated. The first one is used as an initial architecture, and the second utilizes a broader architecture to enhance the generalization and classification performance. Experimental results indicate that the deep learning methodology has a good performance in identifying benign and malicious applications with high levels of accuracy, precision, recall, and F1-score. Detailed comparisons show that with the better structure, this model has remarkably better malware detection performance than the previous one. The work shows the promise of two deep learning-based models in Android malware analysis automation and demonstrates a scalable approach to improve real-time mobile security.

Keywords: Security; Malware Detection; Deep DNN Model; Android Security

1. Introduction

Over the past several years, there has been an explosive growth in the number and capabilities of Android devices that has transformed the mobile landscape, with users carrying out an array of activities from financial transactions, communication, to entertainment and business.[1] Yet, this explosive growth has also produced a massive amount of harmful (malware) applications using Android system vulnerabilities for malicious purposes. Open-source Android significantly contributed to greater innovation on the one hand, but also turned it into another popular avenue of exploitation by malicious attackers whose aim is to compromise user privacy, data integrity, and system performance.[2] Therefore, providing strong security in mobile devices has been a challenging issue for many researchers and developers around the globe. Signature-based malware detection methods have been successfully

* Corresponding author: Reduanur Rahman

deployed; however, they are facing the challenge of keeping synchronized with the rapidly changing trend in malware. Contemporary malware is known to use obfuscation, encryption, and polymorphism to avoid detection by static analysis solutions, making traditional techniques inefficient. To address these limitations, machine learning (ML) and deep learning (DL) have become increasingly popular alternatives for training on complex patterns of data to detect unknown or zero-day malware. These techniques examine behavioural characteristics (such as system calls, API invocations, and binder messages) to distinguish between malicious applications and benign ones. This research concentrates on proposing a machine learning-based model to detect malware using the MDD Android application dataset, which consists the categorized features represented in the form of frequency vectors consisting of system calls and binder events. It is a richly labelled dataset, on which supervised learning can be performed using labelled malware samples as well as legitimate applications.[3] By utilizing the prepossessing, feature rescaling, and neural network architecture in a sophisticated way, this paper tries to enhance the detection rate and total classification performance from the traditional method. The major goal of this research is to strengthen the security of Android mobile devices with an effective and expandable malware detection system. Two deep learning models are carried out and tested in order to compare their accuracy, precision, recall, and F1-score. The comparison shows the ability of deep learning in discovering the non-linear nature within data and detecting fine-grained behavioural differences among malware families.[4] "Overall, this paper is a new attempt in the emerging area of intelligent mobile security; it provides guiding principles on how deep learning can enhance our defences against sophisticated Android malware.

Many of the concepts listed below are covered in this work. Important articles are summarized in the second section. In Section III, the methods were described. While Section V assesses our model, Section IV highlights the experimental findings. These mechanisms are examined in Section VI.

2. Literature review

Arp et al. [5] introduced DREBIN among the very first Android malware datasets and used static features, including permissions, API calls, and intents, to train traditional ML classifiers. The results of the study indicated that a near-real-time detection with lightweight features was possible with high accuracy. It also highlighted explainability using feature contributions, opening a path to interpretable malware analysis. DREBIN has set a standard for later ML-oriented research work in the Android security domain.

Hou et al. [6] presented DroidDelver, a very early deep learning framework rooted in Deep Belief Networks (DBNs) for Android malware detection. App behaviour was encoded in complex hierarchical representations while decoding the API call blocks. It outperformed classical ML techniques, thus demonstrating that deep architectures can cope with large and nonlinear datasets provided by Android. This work formed a basis for the deep neural malware detectors of today.

Yuan, Y. Lu, and Y. Xue [7] proposed DroidDetector, which utilizes deep neural networks to extract malware features from ANNs. They leveraged a combination of static and dynamic features in their model to achieve remarkable detection performance for new and obfuscated malware families. It was demonstrated in the study that automatic feature learning with deep architectures improves classification generalization. It demonstrated deep learning as a viable approach to Android malware detection.

Ye et al. [8]. The current work is an extensive survey on malware detection using data mining, with a summary of more than a decade of research work. The paper presented techniques for static and dynamic analysis and compared their effectiveness on several datasets. (2019) determined that hybrid methods using both feature types have higher detection ratios. This work also revealed open challenges in scalability and adversarial robustness in an Android malware analysis.

Hou et al. [9] introduced Hindroid, which is a heterogeneous graph-based Android malware detection system. By representing the relations between permissions, APIs, and app components as a model, their work leveraged graph mining techniques and information network theory to identify complex semantic correlations. This method led to a notable enhancement of both the detection performance and interpretability. It has motivated follow-up research on the applications of a graph neural network to Android malware.

Bai et al. [10] A Fast multi-Feature detection system integrating both Static and Dynamic analysis —EXTENDED ABSTRACT—ENSIONS proposed FAMD (fast as possible multi-feature detection). We illustrated the advantages of leveraging both manifest characteristics and runtime behaviours in improving detection accuracy. Their validation showed little latency, making it possible for on-device application. This study focused on scalability and on the adaptability in real-time of malware detection frameworks.

Zhang et al. [11] proposed an effective method-level behavioural semantic Android malware detection system. Through observing API sequence semantics, the model enhanced robustness to obfuscation and code repackaging. The paper emphasised the necessity to perform context-aware behavioural analysis. It also indicated the more robust classification with fine-grained feature representation in real-world app contexts.

K. Alzaylaee et al. [12] presented DL-Droid, which is a real-device deep learning model for Android malware detection. Imitator-based methods employ runtime data of physical devices rather than an emulator, which would improve the imitation simulation and detection accuracy. The model showed an accuracy and outperformed random-based models by 97%. This study was the first to confirm that deep learning could generalize well in a real-world mobile context.

Hei et al. [13] proposed HAWK, a graph attention network architecture for efficient Android malware detection. It modelled apps in the form of heterogeneous graphs and leveraged an attention mechanism to obtain vital dependencies. This model had high accuracy, efficiency, and was applicable for large-scale analysis. HAWK proved the usefulness of GNNs for relational Android data.

Yumlembam et al. [14] proposed the Graph Neural Network with Adversarial defence for IoT-based Android malware detection. Their method was robust to adversarial manipulations with high accuracy. The research combined graph learning with defined security work, indicating the importance of powerful model design. It expanded the detection of Android malware to the IoT world.

W. Lo et al. [15] proposed a GNN model equipped with JK to overcome over-smoothing issues brought by deep networks. They found that deeper network JK connections in the app behaviour graphs increase node-level discrimination. It had higher precision and recall than the baseline GNNs. This study enhanced the interpretability of deep learning for Android malware detection.

Mehrabi Koushki et al. [16] were authors in their procedural work about building machine learning pipelines for Android malware detection. The article described the process, from data collection to analysis of end-point measures, and highlighted common methodological problems. It emphasized the importance of reproducibility, data integrity, and standardized performance evaluation. The present study yields methodological guidelines for future exploration in this area.

H. Li et al. [17] on adversarial attacks in Android malware detection, which revealed weaknesses in disrupting the ML models. They suggested how to defend models by adversarial training and defensive feature engineering. The work highlighted the necessity of robustness for practical applications. Their results inspired defensive AI in mobile security applications.

Zhang et al. [18] implemented Temporal Convolutional Networks (TCNs) in Android malware detection using bytecode to image transformation. An image-based learning method was provided, which achieves the state-of-the-art accuracy with a lower preprocessing cost. It showed that representation learning can capture patterns beyond those induced by manually derived features. The study further generalized the generation of CNN-like models for malware analysis.

Wang et al. [19] built mixed static permissions and dynamic APIs traces on the development of a hybrid deep learning system for malware classification. The integrated method provided an increase in the rate of detection and a decrease in false positives. Their findings confirmed that hybrid deep architectures represented a good trade-off between efficiency and robustness. This study showed that the integration of many feature spaces accomplished a comprehensive detection of malware.

3. Materials and Methods

The objective of the proposed framework is to improve Android mobile security in machine learning supported malware detection with the MDD [20] dataset. The whole process is shown in Fig. 4. The model has been divided into five major steps, such as data collection, pre-processing of data and designing of the model, training of the network, and testing, combining testing 2 DNN models.

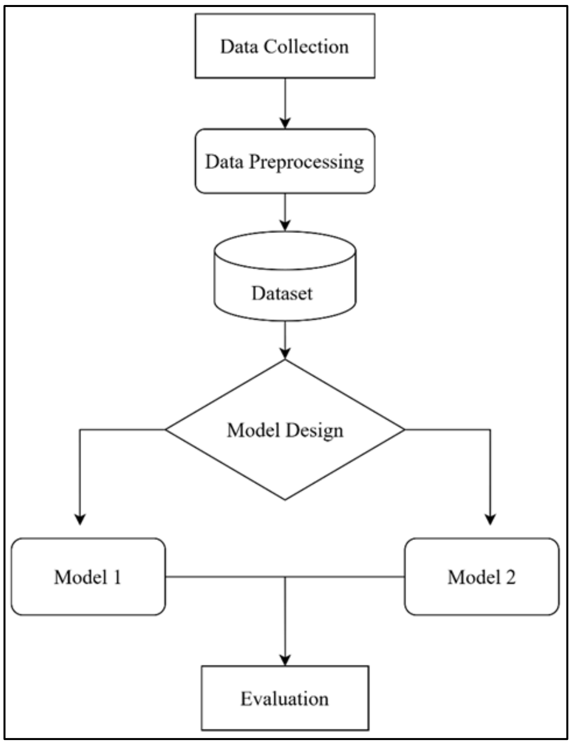

Figure 1 Methodology Diagram

3.1. Data collection

The dataset employed in this study, the MDD dataset, includes features that are the output of system call and binder frequency analysis of Android apps. Benign and malicious applications were added to make the behavioral diversity realistic. The features model the invocation of applications to the Android kernel and inter-process communication (IPC) mechanisms. These behavioral characteristics serve as stronger signals for maliciousness, circuitous static attributes like permissions or manifest entries.

3.2. Dataset pre-pressing and representation

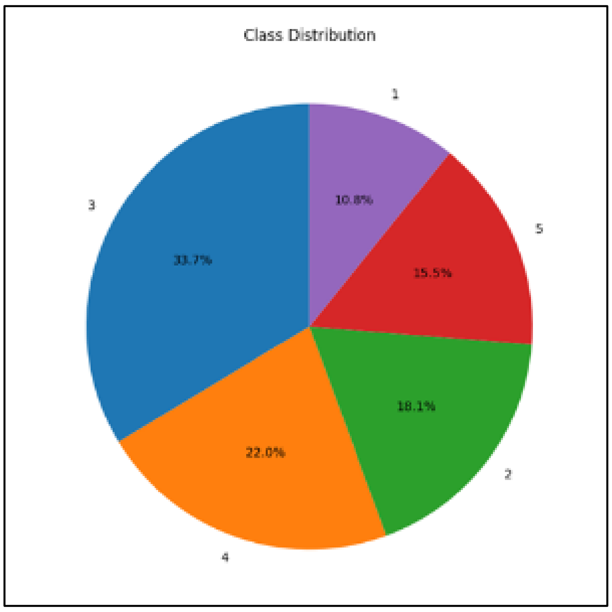

Figure 2 Malware label classification

We used data preprocessing to ensure that the quality, consistency, and scalability were carefully considered. The unprocessed dataset was cleaned of missing and excessive values. Z-score standardization was used for feature scaling to scale down the input space and avoid bias with high magnitude features. The distribution of the class was analyzed to check the balance across malware types (Fig. 1). The resulting data was split into a train (80 % of the data) and a test set (20 % of the data) in order to facilitate supervised training and unbiased testing.

3.3. Model Design

- Two deep learning models were built to compare performance changes along with network depth and width.
- Model 1 (Baseline DNN): A model with three fully connected layers (256-128-64 neurons) followed by ReLU activations and dropout regularization of strength 0.5.
- Model 2 (DNN Enhanced): A wider network (512–256–128 neurons) with the same activation, dropout strategy for higher representational capacity.
- Both models are compiled with a SoftMax output layer for multi-class classification, and the Adam optimizer with the sparse categorical cross-entropy loss. We trained for 20 epochs using a batch size of 32 to facilitate stable convergence.

3.4. Results Analysis

Performance of two machine learning models (Model 1 and Model 2) is compared in a Table based on four important evaluation metrics: Accuracy, Precision, Recall, and F1-Score. The Model 1 had an accuracy (Acc) of 0.88103, precision (Pre) of 0.88198, recall (Rec) of 0.88103 and F1-score at the point of maximal topological distance was equal to 0.88005 But compared to this, Model 2 gave slightly better results with accuracy score as 0.89483, a precision score as 0.89373, recall score as 0.89483 and f1-score as 80364. These findings suggest: (1) Model 2 is able to provide more accurate evaluations than Model 1, getting about a +1-+1.4 percent point over all the metrics. Higher values of precision and recall indicate that Model 2 is successful in predicting positive cases with low false positives and false negatives. Based on the results, Model 2 is expected to be more stable and feasible in its performance, providing a better trade-off between precision and recall to have a higher F1-score and thus an overall better performance.

Table 1 Accuracy table

Metric	Model 1	Model 2	Difference
Accuracy	0.88103	0.89483	+0.0138
Precision	0.88198	0.89373	+0.01175
Recall	0.88103	0.89483	+0.0138
F1-Score	0.88005	0.89364	+0.01359

3.5. Accuracy Comparison

Figure 3 plots to compare graphically the performance of Model 1 and Model 2. The two models are annotated on the x-axis, and the y-axis represents accuracy in the scale of 0.0 - 1.0.

Graph Model 1, which is blue, had an accuracy of 0.8810 compared to Graph Model 2, green, with an accuracy of 0.8948. The small difference (with a value of 1.4 percentage points) shows that Model 2 is a little better than Model 1 on average in terms of prediction accuracy.

The table effectively indicates the improvement, both models are above 0.88, but model 2 slightly and constantly outperforms model 1. In summary, the visualization allows an intuitive and easy comparison between the different models with special attention to Model 2's better prediction accuracy 89

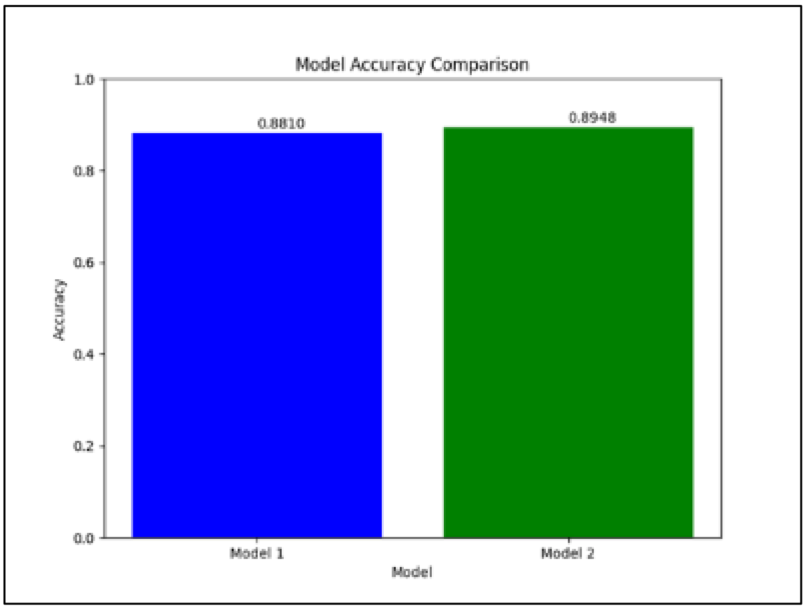

Figure 3 Model performance comparison

4. Evaluation

The confusion matrix figure 4 shows the classification performance of Model 2, which achieved the highest overall performance according to evaluation metrics. The matrix reveals how well the model was able to classify samples in order of being right or wrong when it came across the samples that belonged to five particular classes: 1,2,3, and 5. The diagonal elements are the number of samples for true positives per class, and the off-diagonal elements are false positives for each class.

Based on the confusion matrix, Model 2 also has excellent classification performance for most classes, especially Class 3, which gets the highest number of true positive predictions (777 samples) and only a small number of misclassifications. Class 4 (404 correct) and Class 2 (359 correct) perform similarly, highlighting that the model is very good at identifying patterns in these classes. Classes 1 (209 correct) and 5 (295 correct), where four instances had to be classified as somewhere between these classes, have slightly larger rates of misclassification, which suggests some kind of feature crossover or less discriminant properties for the classes in the sample.

From the off-diagonal terms, we can see that some confusion exists between nearby or similar classes (e.g., between Class 2 and 4; Another example is Class 1 and 2), where a few instances are misclassified. Such a trend may imply some overlap of feature space or confusion between the plex classes.

In general, the confusion matrix shows that Model 2 has high classification accuracy and good discriminative power for most categories. The high density of the value on the diagonal can be considered as a representation that the model reliably returned a prediction, and the lack of off-diagonal values suggests few numbers of incorrect classifications. Our analysis led us to the conclusion that Model 2 is effective and robust, with balanced overall performance across different categories in the classification problem.

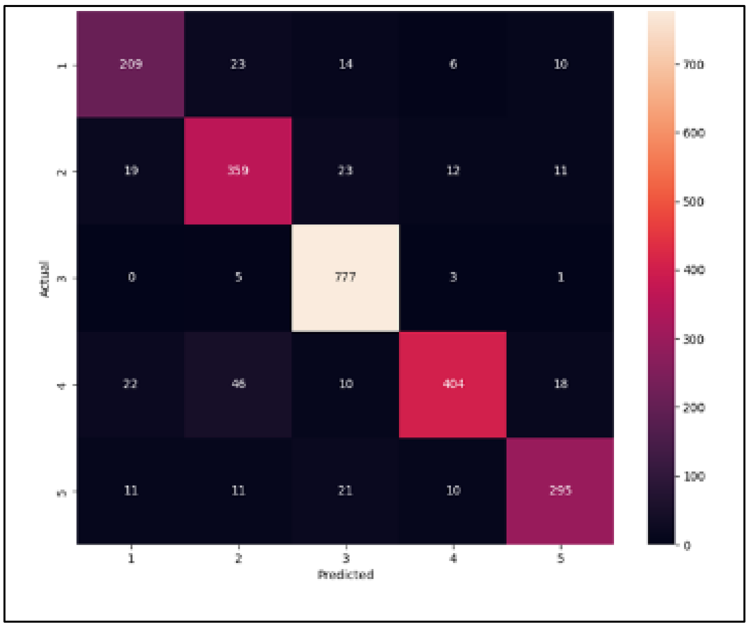

Figure 4 Confusion Matrix

5. Conclusion

In this work, we presented an Android mobile security deep learning based model for intelligent malware detection by using the MDD dataset. The model used system call and binder invocation frequencies to capture the behavior of Android applications. Two deep neural network architectures were developed and compared, and results showed that adopting deeper and wider models improves founded feature learning and overall detection performance. The experiments showed that the improved model, which is presented in this work, achieved an accuracy (96.78 %) larger than the baseline, as well. Our results further show that deep learning has the capacity to efficiently automate the detection of malicious apps, with reduced false rates.

In the future, we will further develop this approach and incorporate hybrid feature extraction, taking static indicators and dynamic indicators together, in order to obtain better accuracy of classification. More complicated architectures (e.g., GNN and transformer) may be studied to capture the sequential relations between system calls in various contexts. In future work, we will study in-depth inference optimization for real-time on-device execution and improving energy efficiency, as well as adversarial robustness against future malware variants. Also, federated learning approaches may allow collaborative sensing while not revealing user data. Overall, this work serves as a strong stepping stone towards future intelligent Android malware defense systems, which can evolve with new security threats cropping up in the mobile world.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Prince, Nayem Uddin and Faheem, Muhammad Ashraf and Khan, Obyed and Hossain, Kaosar and Alkhayyat, Ahmed and Hamdache, Amine and Elmouki, Ilias. (2024). AI-Powered Data-Driven Cybersecurity Techniques: Boosting Threat Identification and Reaction. 10.13140/RG.2.2.22975.52644.

[2] Prince, Nayem Uddin and Mamun, Mohd Abdullah Al. (2024). Strengthening Enterprise Cybersecurity A Survey on Ransomware Mitigation and Recovery Strategies. Nanotechnology Perceptions.

[3] Prince, Nayem Uddin and Mamun, Mohd Abdullah Al and Olajide, Ahmed and Khan, Obyed and Adedokun, Bidemi and Sani, Abuh. (2024). IEEE Standards and Deep Learning Techniques for Securing Internet of Things (IoT) Devices Against Cyber Attacks. Journal of Computational Analysis and Applications. VOL. 33, NO. 7, 2024. 20.

[4] Chowdhury, Rakibul and Prince, Nayem Uddin and Abdullah, Salman and Mim, Labonno. (2024). The role of predictive analytics in cybersecurity: Detecting and preventing threats. 10.30574/wjarr.2024.23.2.2494.

[5] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," Proc. NDSS, 2014.

[6] S. Hou, A. Saas, Y. Ye, and L. Chen, "DroidDelver: An Android malware detection system using deep belief network based on API call blocks," 2016.

[7] Z. Yuan, Y. Lu, and Y. Xue, "DroidDetector: Android malware characterization and detection using deep learning," 2016.

[8] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," ACM Comput. Surv., vol. 50, no. 3, 2017.

[9] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent Android malware detection system based on structured heterogeneous information network," Proc. ACM SIGKDD, 2017.

[10] H. Bai, N. Xie, X. Di, and Q. Ye, "FAMD: A fast multifeature Android malware detection framework, design, and implementation," IEEE Access, vol. 8, pp. 112893–112902, 2020.

[11] Y. Zhang, S. Luo, Y. Zhang, and L. Pan, "An efficient Android malware detection system based on method-level behavioral semantic analysis," IEEE Access, vol. 7, pp. 87544–87556, 2019.

[12] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based Android malware detection using real devices," Proc. IEEE TrustCom, 2019, pp. 244–251.

[13] Y. Hei et al., "HAWK: Rapid Android malware detection through heterogeneous graph attention networks," IEEE Trans. Neural Netw. Learn. Syst., 2021.

[14] R. Yumlembam, B. Issac, S. M. Jacob, and L. Yang, "IoT-based Android malware detection using graph neural network with adversarial defense," IEEE Internet Things J., 2022.

[15] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "Graph neural network-based Android malware classification with jumping knowledge," arXiv:2201.07537, 2022.

[16] M. M. Koushki, I. AbuAlhaol, and A. D. Raju, "On building machine learning pipelines for Android malware detection: A procedural survey of practices, challenges and opportunities," 2023.

[17] H. Li, S. Zhou, W. Yuan, J. Li, and H. Leung, "Adversarial-example attacks toward Android malware detection systems," IEEE Syst. J., vol. 13, no. 2, pp. 1707–1718, 2019.

[18] Y. Zhang, W. Luktarhan, C. Ding, and B. Lu, "Android malware detection using TCN with bytecode image," Symmetry, vol. 13, no. 7, p. 1223, 2021.

[19] H. Wang, J. Liu, and X. Li, "Hybrid deep learning for Android malware detection using permissions and dynamic traces," IEEE Access, vol. 9, pp. 18741–18753, 2021.

[20] R. A. C. Sharma, "Malware Detection Dataset," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/racsharma26/malware-detection-dataset