ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 14 (2024) No. 8, pp. 299-309
DOI:10.33168/JSMS.2024.0819

Deep Learning-Based Malware Detection via Meta-Information and Manifest Analysis in Android Systems

Mohammad Rasmi Al-Mousa[1], Abdullah Alqammaz[1], Mohammed Abdulrazzaq Rajab[2], Ala Alshiakh[1], Ayman Ghaben[1]

[1]Department of Cybersecurity, Zarqa University, Zarqa, Jordan,
[2]University Headquarter; University of Anbar, Ramadi, Anbar, Iraq
{mmousa, a.qammaz, ashaikh}@zu.edu.jo, m.a.rejab.d@uoanbar.edu.iq, aghaben@zu.edu.jo

Abstract. Day by day, the number of users of electronic devices is increasing around the world, as electronic- based systems have become a part of human life. The Android system is one of the largest and most popular systems used in different devices such as mobile devices. The ease of using Android system applications and their availability is main reason behind the widespread a vast number of applications. For example, gaming, social networking, and other sensitive applications such as banking application systems. However, as android is an open-source system, it becomes a valuable opportunity to increase malware and malicious applications to achieve cybercrime such as privacy violation, theft, extortion, and also other crimes that have become a major challenge for security information specialist and a important threat to the daily human life. Regardless of many developed protection systems and methods for detecting malware, the development and methods used by cybercriminals make some of these traditional methods useless in protecting devices and information. Hence, researchers have resorted to the use of artificial intelligence-based techniques to develop different protection systems against malwares. This study proposes deep learning- based technique for malware detection based on the meta information of applications; accuracy results for Various Deep Learning Techniques (RNNs, LSTMs, CNNs, and Bi-LSTMs) are 88.3, 88.3, 88.4 and 88.2 respectively.

Keywords: malware detection, android, manifest, permission, deep learning, meta-information

1. Introduction

The use of smart phones has spread greatly since the beginning of their launch in the market, and the percentage of sales began to increase annually by a very large percentage, and statistics indicate that the percentage of sales of smartphones increased during the ninth, tenth, and eleventh months of 2011 to more than 111% compared with what was sold in 2010 during the first, second and third months, in 2010- 54 million devices and in 2011- 115 million devices within the mentioned periods (Sahs & Khan, 2012). With the passage of days, mobile devices have become an essential element in life and in most areas such as communication, electronic payment, banking, learning, and etc. The Android system is the most popular and most widely used among the smart phone systems. Since the launch of this system in 2007, millions of people have started using it, and to this day, the number of devices used for this system has reached three billion active devices, according to Google reports. In 2020, the number of smart phone users reached 2.8 billion users, and the Android system users exceeded 2.02 billion users, i.e. 72.26% of the total number of smart phone users (Sihag et al., 2021). Millions of different applications work on the Android system in various fields, as the number of applications on the official store of Google is 2.8 million applications. In addition to external stores and according to Google statistics in 2015, more than 65 billion downloads are performed from the official store only.

The open-source environment is one of the most important factors that made the Android system very popular, especially for developers, its wide popularity and uses make it a focus of attention for illegal developers who are working to exploit these factors to carry out their cybercrimes in various ways (Mohammad, Fuad, & Hourani, 2016). Therefore, annually millions of cases of electronic attacks are recorded that cause violation of privacy and data loss as well as theft and financial losses (Alzaylaee, Yerima, & Sezer, 2020; Shamshirsaz, Asghari, & Marvasti,2022). At the beginning of 2017, more than 750,000 applications infected with malware were discovered and the number is increasing, and in 2020, about 24,000 malware applications are blocked from smart phone applications, most of which are within the Android system (Abuthawabeh, & Mahmoud , 2020; Sihag et al., 2021). The Kaspersky, one of the leading companies specializing in electronic systems protection technologies, announced that in 2020 only, more than 5.5 million malicious packages were discovered within the Android system and that the percentage of increase over what was discovered of malware packages for the year 2019 is about 60%. On the other hand, McAfee reported that the percentage of malware samples that were discovered in 2019 had increased by 25% from what was announced in 2018 (Liu, Tantithamthavorn, Li, & Liu, 2022). Most of the current Android application markets are insecure, with the exception of Google's official store. Cybercriminals exploit these markets to carry out their cybercrime.

Although there are tools to detect malicious applications, they are considered ineffective, the concerns of those interested and specialist in the field of information security have risen from this rapid growth and the methods of malicious applications, which have become a permanent source of concern for users of Android systems. For these reasons, researchers and analysts resort to new methods to protect these systems and develop new tools to detect malware applications. Indeed, there are three categories on the basis of which malware detection techniques can be classified, static detection, dynamic detection, and mixed detection (Zhang, Tan, Yang, & Li,, 2021). The basis for traditional protection methods is to rely on identifying multiple patterns, based on signature through matching, which usually with low accuracy and long time in discovery is the most important problems of such methods (Lu et al., 2020). Some researchers and data security specialists have started working on automatic detection solutions through the use of artificial intelligence, and this has been widely applied in detecting malware, and these technologies have proven their quality compared to traditional methods that cannot detect already unknown malware. Special methodologies and methods that can be applied through machine learning models and deep learning models to detect malicious programs by analyzing and understanding the behaviors of these programs (Alzaylaee et al., 2020; Ma et al., 2020; Khalifeh et al., 2022).

An android mobile system is vulnerable to be attacked by illegal hackers to exploit the weakness

points of its an open source by infected it with malicious programs which is a significant gap which is discovered by many organizations; this study works to develop DL method to detect them effectively.

This study makes a significant contribution to the advancement of malware detection by introducing predictive models based on deep learning algorithms. It delves into the clarification of fundamental concepts related to detecting malware through Deep Learning (DL) techniques. The research provides a succinct overview of applied deep learning models detecting malwares. Additionally, a concise summary is presented for the dataset employed in the experiment. Furthermore, the study conducts a thorough review of pertinent concept and analysis literature review related in the field. The study unfolds in the following structure; Section 2 delves into an important concept and analysis Literature Review, while Section 3 explains the proposed methodology. Moving forward, Section 4 outlines the experimental setup and results. Section 5 delves into a detailed discussion of the results, while Section 6 outlines the conclusion and avenues for future work.

2. Literature Review: Concepts and Analysis

In the pursuit of detecting malicious software, the foundational approach involves the analysis of malware programs and applications. It is noteworthy that each analysis method possesses inherent advantages and disadvantages. The primary classifications of malware analysis are static analysis, dynamic analysis, and mixed analysis (hybrid). Static Analysis, which occurs without the execution of the application and involves a comprehensive examination of the code's internal structure. This strategy works specifically well with conventional techniques and out-of-date malicious software. Nevertheless, when it comes up against contemporary malware methods which pose problems such code obfuscation and confusion, its efficacy wanes (Kouliaridis & Kambourakis, 2021). In contrast, dynamic analysis occurs during the operation of the program and offers insights into the code's execution inside a controlled setting. Time cost is a significant disadvantage, but dynamic analysis outperforms static analysis in that it reveals information that static analysis might miss (Kouliaridis, Potha, & Kambourakis, 2020). Hybrid analysis, or mixed analysis, integrates both static and dynamic analyses in order to achieve a cost-efficiency balance. By utilizing the advantages of each technique, this strategy aims to provide a more exhaustive and sophisticated understanding of the software under analysis. Table 1 illustrates the available alternatives for feature extraction based on the type of analysis.

Table 1. Analysis types and feature extraction options.

Analysis Type	Feature Extraction Method	Features Extracted
Static	Manifest analysis Code analysis	Package name, Permissions, Intents, Activities, Services, Providers API calls, Information flows, Taint tracking, Opcodes, Native code, Cleartext analysis
Dynamic	Network traffic analysis Code instrumentation System calls analysis System resources analysis User interaction analysis	URLs, IPs, Network Protocols, Certificates, Non-encrypted data, Java classes, intents, networks traffic, System calls, CPU, Memory, and Battery usage, Process reports, Network usage, Button, Icons, Actions/Events

In the realm of Android application analysis, the Android Manifest stands as a pivotal file encompassing crucial information about the application. This file includes details such as the

application's version, permissions, icon, broadcast receivers, activities, and its overall purpose (Kumar, Mishra, Panda, & Shukla, 2021). Permissions, a vital aspect defined by application developers, grant access to specific private elements on the user's device with user consent. This feature also enables interaction with various activities on the user's device, covering elements like location, camera, microphone, photo repository, and contacts (Li et al., 2017). Figure 1 illustrates an example of permissions presented to the user during the installation of an Android application (Peiravian & Zhu, 2013).

Another critical component is Meta-Information, or metadata, which is visible to users during the download of Android applications. This information, detailing aspects like file size, requested permissions, and application ratings, is authored by the developer through the manifest. Notably, hackers exploit this feature to deceive users by portraying malware applications as benign. For instance, attackers may limit the permissions shown to the user, such as accessing contacts only, while concealing additional permissions like camera and photo storage access. Figure 2 provides a comparative analysis of permissions requested by malicious and benign applications, revealing that malicious applications typically demand more permissions, except for certain permissions that align closely between the two, which are generally not targeted by attackers (Peiravian & Zhu, 2013).

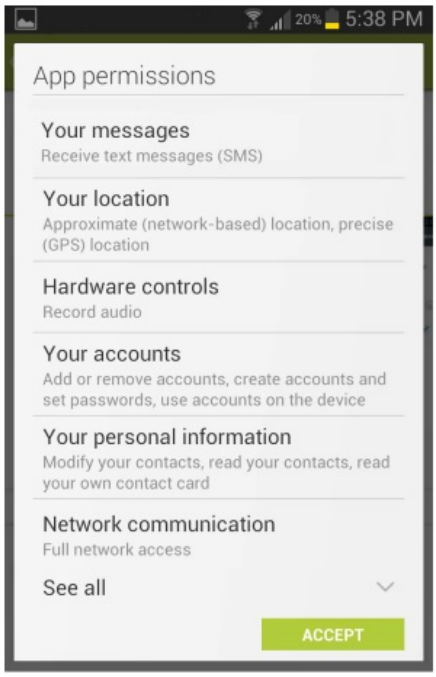

Fig.1: An example of the permissions that appear to the user when installing an Android application.

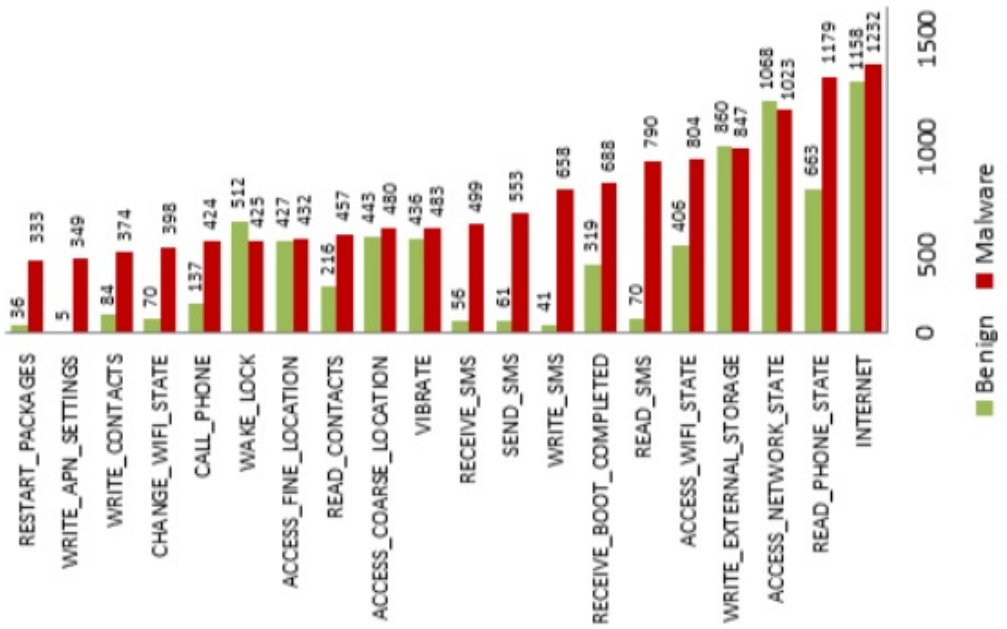

Fig.2: Comparison of malware applications and benign applications in some of the most important permissions.

3. Methodology

Data Preparation

The dataset utilized in this study integrates metainformation from both benign and malware Android samples, obtained by downloading 12,360 applications from the Aptoide market, a prominent platform for Android applications. Subsequently, an integrity assessment using VirusTotal, equipped with over 50 anti-virus engines, identified 4,799 applications with malicious programs. After refining the dataset to exclude applications lacking descriptions, it comprised 3,418 malicious applications and 8,058 benign applications. Each application is linked to a file containing crucial descriptive information and a classification indicating benign or malicious status. The dataset is accessible for further exploration and research at https://www.kaggle.com/datasets/saurabhshahane/android-malware-dataset.

This dataset, comprising 138 features and 11,476 records, underwent a comprehensive examination, revealing 8,058 "benign" and 3,418 "malware" records. The data integrity check showed missing or unknown data at a rate below 2% of the total dataset. To ensure uniformity and eliminate biases introduced by individual record names representing applications, the names were omitted. Missing values were substituted with (0). As mentioned previously in Literature Review section, the cost and time are the main disadvantages of existing strategies which are used to detect the illegal codes in an android system; so this study justify the usage the following features which are contained text data and symbols and converting them into a digital format using the LabelEncoder library to enhance uniformity and computational tractability of them to measure the rate of enhancement in the cost and time of the detection method through applying it on the dataset. The ensuing Table 2 showcases the successful re-encoding of samples of features with text data and symbols, contributing to a standardized and analytically accessible dataset.

Table 2: Features Encoding.

	value	
Feature	Original data (before encoding)	The values (after encoding)
MD5	7de8fa31cb8c1145f49707a1feb61c45	5728
version	1.3	456
Min_Screen	small	3
Supported_CPU	armeabi-v7a	32
Signature	CE:26:15:7A:4C:7D:D6 :B4:06:34:3C:AB:CD:9A:FE:F...	3867
Organization	Studio C	2214
Developer	NaN	2223
Locality	NaN	2
Country	NaN	0
State	NaN	0
description	Animal Link is a puzzle game for everyone! Esp...	636

Detection Models

In the domain of deep learning methods, Recurrent Neural Networks (RNNs) play a pivotal role with their unique loops facilitating data storage within the network. These networks do exceptionally well in inferring conclusions for the future from previous reasoning and are notable for their capacity to sequence vectors, which improves their ability to handle increasingly complex tasks (HEIKAL et al., 2018). One subclass of RNNs known for its enhanced memory capacity is the Long Short-Term Memory (LSTM) network, which is especially good at learning from temporally dispersed events. In order to select the storage of information, LSTMs use gates, which allows for learning over time based on the importance of the data (HEIKAL et al., 2018; SETYANTO et al., 2022).

A further important participant is the Convolutional Neural Network (CNN), which is designed primarily to handle structured input arrays, especially pixel-based data such as photographs. CNNs are now essential for tasks like text categorization through natural language processing, image classification, and computer vision since they use convolution rather than standard matrix multiplication (HEIKAL et al., 2018). Combining the strengths of CNNs and LSTMs, the CNN-LSTM model proves versatile, excelling in sequence detections and finding application in visual time series detection problems, as well as generating text descriptions for picture sequences (SETYANTO et al., 2022).

Introducing a sequential processing technique, Bidirectional Long Short-Term Memory (Bi-LSTM) incorporates two LSTMs that accept input in opposite directions, significantly enhancing information availability. This bidirectional approach optimizes contextual understanding, exemplified by an awareness of words that precede and follow a particular word in a sentence (BERGLUND et al., 2015). This study performed deep learning techniques developed in python programming language includes RNN, LSTM, CNN-LSTM, Bi-LSTM. Figure 3 illustrates the methodology workflow of the performed models in this study.

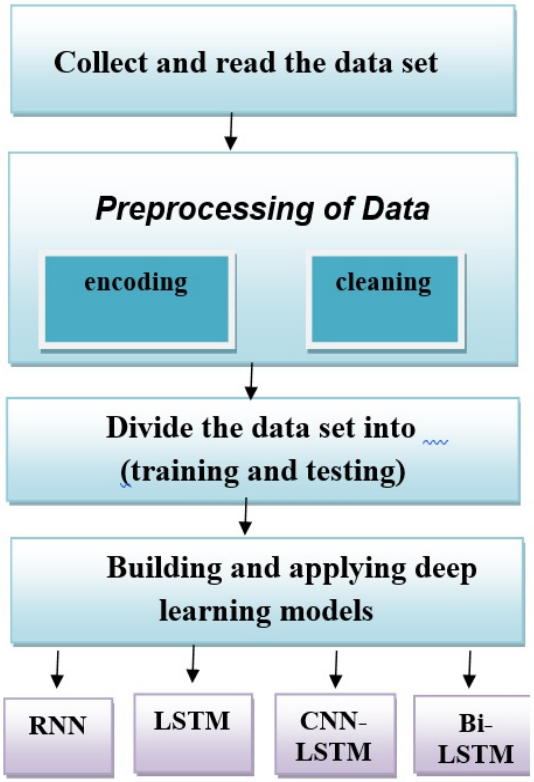

Fig.3: Methodology workflow.

4. Experimental Results

This section presents the results of undertaken experiments in this study. Building deep learning models involved in used specific parameters such as extreme features, inline vector length, and number of neurons, number of layer periods, batch size, and LSMT neurons, iterative dropout, test rate, and number of filters and length of filter and activation function. On the other side, the experimental evaluation of various deep learning models for Android malware detection yielded insightful performance metrics, as presented in the Table 3 and Figure 4 below.

The accompanying table contains a comprehensive list of evaluation metrics that are utilized to evaluate each model's performance in detecting Android malware. These performance indicators, which included accuracy, precision, recall, F-score, and area under the curve (AUC), were like benchmarks for us to compare the relative strengths of each model. By combining all of these variables, this study was able to gain a richer understanding of the benefits and drawbacks of each model, including RNNs, LSTMs, CNNs, and Bi-LSTMs. The table highlights the significance of each statistic in obtaining a comprehensive understanding of the performance of these varied models by demonstrating how well each model could distinguish between good and bad apps.

Table 3: Comparative Accuracy Results for Various Deep Learning Techniques

Model	Accuracy%	Precision	Recall	F-score	AUC
RNN	88.3	0.876	0.727	0.794	0.859
LSTM	88.3	0.968	0.644	0.773	0.863
CNN	88.4	0.928	0.679	0.784	0.913
Bi-LSTM	88.2	0.996	0.644	0.773	0.876

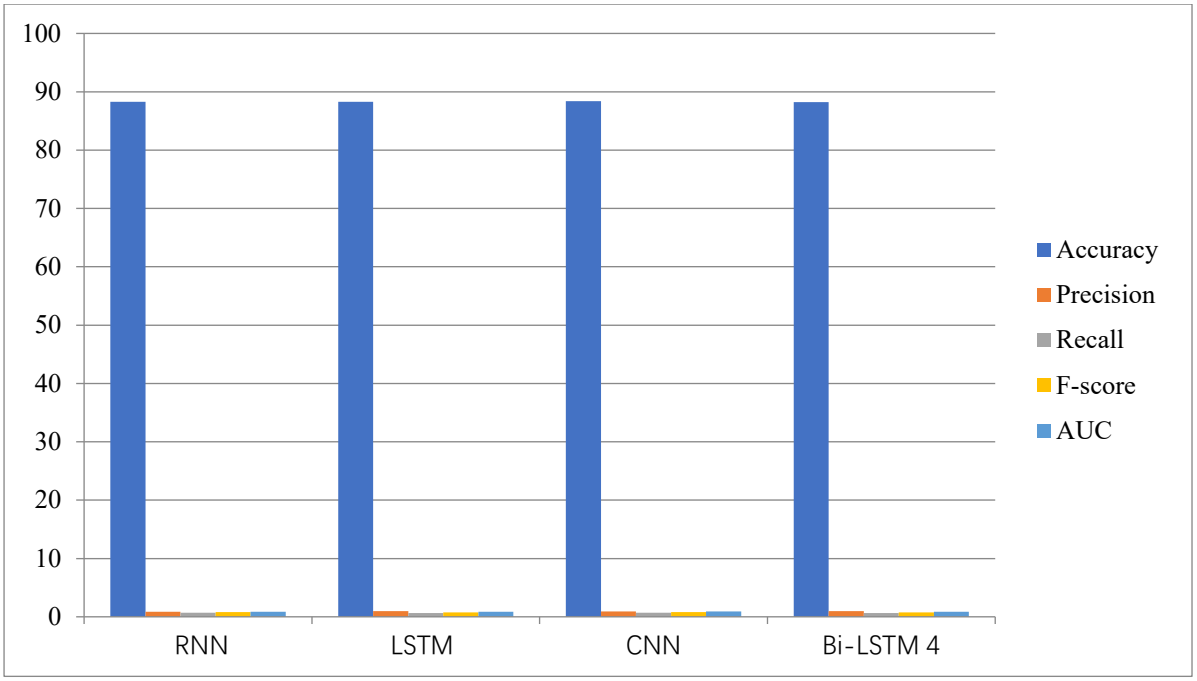

Fig.4: Comparative Accuracy Results for Various Deep Learning Techniques.

Therefore, when it comes to identifying Android spyware, every model that was tested performed well. Nevertheless, the LSTM outperformed the RNN in detecting all malware with recall at 72.7%, but the RNN excelled in recognizing true positives with precision at 96.8%. Both models had an accuracy of 88.3%. With 88.4% accuracy, the CNN trailed closely, striking a balance between recall (67.9%) and precision (92.8%). Lastly, the Bi-LSTM caught almost all malware with remarkable precision at 99.6% and a slightly lower accuracy at 88.2%. These findings demonstrate the usefulness of deep learning models, each with unique advantages, for Android security. Furthermore, all models demonstrated strong AUC values, indicating their capacity to distinguish between safe and dangerous apps.

Remarkably, both the RNN and LSTM models have accuracy levels of 88.3%; the RNN model has excellent precision at 87.6%, while the LSTM model has remarkable precision at 96.8%. Recall, on the other hand, shows a trade-off, with RNN showing a higher value with 72.7% than LSTM 64.4%. The F-score reflects this trade-off, with LSTM scoring 77.3% and RNN scoring 79.4%. With RNN at 85.9% and LSTM at 86.3%, the models' AUC values, which show how well they can distinguish between benign and malicious applications, are likewise similar.

5. Discussion

The results displayed in Table 3 offer significant understanding of the LSTM model's performance, especially when compared to well-known conventional machine learning techniques. Even with concentrated attempts to fine-tune feature selection and optimize parameters, the various deep learning models' peak accuracy is astounding. Still, it makes one consider the particular difficulties presented by the dataset and the complexities of deep learning techniques. To achieve increased accuracy, a number of iterations were carried out to adjust various factors, such as the number of layers and neurons in each layer. Unfortunately, these careful changes did not produce results that were better than those shown in Table 3. This begs the question of how well deep learning approaches can be tailored to the particular intricacies of the data. at hand.

On the other side, the investigation of manipulating datasets added still another level of intricacy. Trials included deliberate deletion and insertion of particular columns together with encoding re-representation. Even though the goal of these projects was to improve the dataset's informative value,

several of the trials produced worse than ideal outcomes. This is explained by the inherent nature of the dataset, which included columns with missing critical data that contributed to the features that the models were trained on.

6. Conclusion and Future Work

In conclusion, this study on the effectiveness of deep learning models for Android malware detection has produced some interesting results. The outcomes of the experiments demonstrate the strong capabilities of the RNN, LSTM, CNN, and Bi-LSTM models in classifying applications as malicious or benign. The results show that each model demonstrates distinct capabilities in terms of precision, recall, and overall accuracy, which are mirrored in the thorough evaluation measures.

The discussion highlights the difficulties of fine-tuning deep learning models—in particular, the LSTM model, to the dataset's complexities. Even with intensive attempts to manipulate datasets and modify parameters, a major breakthrough in exceeding the accuracy criteria set by conventional machine learning approaches is still elusive. This raises important questions about how well-suited deep learning techniques are to the particular subtleties of the situation under investigation. Future research endeavors will further explore the domain of deep learning technology. The goal is to improve features by employing sophisticated methods and feature-specific algorithms. The main goal is to fully utilize deep learning techniques for Android malware detection, which will need a careful reevaluation of feature engineering strategies. Furthermore, this study suggests investigating more effective methods for multimodal parameter tweaking by utilizing advanced strategies like evolutionary algorithms.

Moreover, the proposed method has opened paths for future research into the detection of illegal programs inside android mobile system.

- Improve the security policies and detection mechanisms to detect malware attacks by using DL on different the android mobile system layers.
- The promising results of the proposed method to detect malware on the dataset which is used in this study can be generalized to be applied to detect different types of attacks against in the android mobile system.
- Lastly, combining the proposed detection method with network management and security systems will add value to the applications of android mobile system.

The findings confirm overall malware discrimination potential across models but performance maximization remains challenging. However, purpose-built mobile security demands holistic preparedness spanning detection, response and recovery across devices, applications and networks. As multi-point attack vectors rapidly evolve in sophistication, fluid defense mobilization through collective coordination is imperative. Though promising, exclusive reliance on even advanced artificial intelligences remains precarious without embedding prudence promoting cyber-hygiene as a social virtue. Technology stewardship rooted in ethical governance can power protective prosperity.

In summary, even though our research shows that deep learning models perform well, more research and development are required to achieve better accuracy and flexibility. The next research will address the changing issues in this dynamic sector and advance the state-of-the-art in Android malware detection; the objective from the proposed contribution of this study which is mentioned in this study is achieved.

References

Abuthawabeh, M., & Mahmoud, K. W. (2020). Enhanced android malware detection and family classification, using conversation-level network traffic features. *Int. Arab J. Inf. Technol.*, *17*(4A), 607-614.

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, 101663.Nieto, A., Rios, R., & Lopez, J. (2018). IoT-forensics meets privacy: towards cooperative digital investigations. Sensors, 18(2), 492.

BERGLUND, Mathias, et al. Bidirectional recurrent neural networks as generative models. Advances in neural information processing systems, 2015, 28.

HEIKAL, Maha; TORKI, Marwan; EL-MAKKY, Nagwa. Sentiment analysis of Arabic tweets using deep learning. Procedia Computer Science, 2018, 142: 114-122.

Khalifeh, A. F., AlQammaz, A. Y., Abualigah, L., Khasawneh, A. M., & Darabkh, K. A. (2022). A machine learning-based weather prediction model and its application on smart irrigation. *Journal of Intelligent & Fuzzy Systems*, *43*(2), 1835-1842.

Kouliaridis, V., & Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. Information, 12(5), 185.

Kouliaridis, V., Potha, N., & Kambourakis, G. (2020, November). Improving android malware detection through dimensionality reduction techniques. In International Conference on Machine Learning for Networking (pp. 57-72). Springer, Cham.

Kumar, S., Mishra, D., Panda, B., & Shukla, S. K. (2021, December). DeepDetect: A Practical On-device Android Malware Detector. In 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) (pp. 40-51). IEEE.

Li, L., Gao, J., Hurier, M., Kong, P. F., Bissyandé, T. F., Bartel, A., ... & Le Traon, Y. A. (2017). Collecting millions of Android apps and their metadata for the research community. *arXiv preprint arXiv:1709.05281*.

Liu, Y., Tantithamthavorn, C., Li, L., & Liu, Y. (2022). Explainable AI for Android Malware Detection: Towards Understanding Why the Models Perform So Well?. arXiv preprint arXiv:2209.00812.

Lu, T., Du, Y., Ouyang, L., Chen, Q., & Wang, X. (2020). Android malware detection based on a hybrid deep learning model. Security and Communication Networks, 2020.

Ma, Z., Ge, H., Wang, Z., Liu, Y., & Liu, X. (2020). Droidetec: Android malware detection and malicious code localization through deep learning. arXiv preprint arXiv:2002.03594.

Mohammad, H., Fuad, A., & Hourani, M. A. (2016). Using Mobile Technologies for Enhancing Student Academic Experience: University of Jordan Case Study. *International Journal of Interactive Mobile Technologies*, *10*(1).

Peiravian, N., & Zhu, X. (2013, November). Machine learning for android malware detection using permission and api calls. In 2013 IEEE 25th international conference on tools with artificial intelligence (pp. 300-305). IEEE.

Sahs, J., & Khan, L. (2012, August). A machine learning approach to android malware detection. In 2012 European Intelligence and Security Informatics Conference (pp. 141-147). IEEE.

SETYANTO, Arief, et al. Arabic Language Opinion Mining Based on Long Short-Term Memory(LSTM). Applied Sciences, 2022, 12.9: 4140.

Shamshirsaz, B., Asghari, S. A., & Marvasti, M. B. (2022). An Improved Process Supervision and Control Method for Malware Detection. *INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY*, *19*(4), 652-659.

Sihag, V., Vardhan, M., Singh, P., Choudhary, G., & Son, S. (2021). De-LADY: Deep learning based Android malware detection using Dynamic features. J. Internet Serv. Inf. Secur., 11(2), 34-45.

Zhang, N., Tan, Y. A., Yang, C., & Li, Y. (2021). Deep learning feature exploration for android malware detection. Applied Soft Computing, 102, 107069.