International Journal of Computers and Informatics (IJCI)
Vol. (4), No. (3)

IJCI

المجلة الدولية للحاسبات والمعلوماتية
الإصدار (4)، العدد (3)

March 2025

Malware Detection for Android Systems using Neural Networks

Reem A. Kh. A. Almeshal
Training team member at the Public Authority for Applied Education and Training (PAAET) - Higher Institute for Administration Services, Kuwait
Ra.almeshal@paaet.edu.kw

Abstract

The proliferation of Android malware poses an ever-growing menace to billions of mobile users worldwide. Detection systems are updated constantly to address these threats. Nevertheless, a counteraction arises in the form of evasion attacks, where an opponent modifies malicious samples in a way that causes them to be incorrectly classified as benign. In this paper, the proposed method aimed to investigate the signs of malware on Android devices, and to develop a malware detection model for Android systems based on the Drebin and the MH-100K datasets. We used each of the LSTM, MLP, and RNNs for reducing and detecting the threats and malware to enhance security over the Android systems. Each algorithm works separately and calls the sub-algorithms in the feature selection (PCA, and CFS). We used several scenarios for testing the performance of each algorithm according to the number of attributes in both datasets and the number of epochs for each algorithm. The experiment results showed the preference of results for the MH-100K dataset compared to the Drebin dataset. On the other hand, the results showed that the accuracy for the LSTM algorithm reached (98.31%) and outperformed both the MLP and the RNN algorithms for malware detection for both datasets.

Keywords: Malware Detection, Android Systems, Deep Learning Methods, Drebin Dataset, MH-100K Dataset.

Introduction

Mobile malware can manifest in various guises [1], yet users may lack the knowledge to discern its presence [2]. Malware poses a significant risk to enterprise endpoints, and mobile administrators should possess knowledge on identifying and eliminating this menace on Android devices [3].

Mobile devices pose a substantial risk to enterprises, and organizations should not overlook their vulnerability to malicious attacks orchestrated by cybercriminals with the intention of data theft [4]. Mobile malware encompasses various manifestations, such as spyware [5], ransomware [6], and Trojan horses [7]. Additional strategies have arisen, including smishing (SMS phishing) [8], which involves cybercriminals sending a corrupted text message to a mobile device to deceive the user into installing malicious software onto the device [6].

These kinds of malware can inflict substantial damage by pilfering sensitive corporate and user data, disrupting operations, impairing hardware, or divulging confidential information. To mitigate these hazards, organizations must comprehend the perils associated with mobile malware and implement measures to safeguard their devices [9], [10].

Mobile malware prevention measures encompass the implementation of robust security protocols, including the enforcement of authentication and authorization requirements [11]. Additionally, security and encryption policies can be enforced through the utilization of mobile device management (MDM) systems [12]. Furthermore, mobile malware detection and antimalware tools can be employed to enhance security. Furthermore, organizations must provide comprehensive training to their users regarding the identification of potential threats and the appropriate actions to be taken in the event of encountering suspicious activity. Organizations can

safeguard themselves from the detrimental effects of malware on mobile devices by implementing these proactive measures [13].

In this study, the proposed method aimed to investigate the signs of malware on Android devices, and to develop a malware detection model for Android systems using each of the Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), and Recurrent neural networks (RNNs) for reducing and detecting the threats and malware to enhance security over the Android systems.

Problem Statement

The proliferation of Android malware poses an ever-growing danger to billions of mobile users worldwide. Continuous updates are made to detection systems to effectively address these threats. Nevertheless, a counteraction arises in the form of evasion attacks [14], wherein an opponent modifies malicious samples in such a way that these samples are incorrectly classified as benign.

The Android operating system does not pose an inherent security risk. Nevertheless, Android devices are vulnerable to malware due to several factors. Android being Open Source allows any developer to access the code and potentially create applications with malicious intentions [15]. Furthermore, Android's substantial worldwide market dominance renders it highly susceptible to potential security breaches.

An additional complexity of the Android ecosystem lies in the multitude of device manufacturers and carriers, each of whom holds a crucial responsibility in delivering software updates for their respective devices. This can lead to a fragmented ecosystem of devices operating on obsolete or unpatched iterations of the Android operating system [16].

Since 2012, the Android operating system has maintained its status as the most widely used platform for smartphones and tablets. This surge in Android malware has been a direct result of its increasing popularity in recent years. The complexity of Android

malware obfuscation and evasion techniques has greatly advanced, rendering numerous conventional malware detection methods outdated [17].

This study aimed to examine the indicators of malware on Android devices and create a malware detection model for Android systems using Neural Networks. The objective is to mitigate and identify threats and malware, thereby enhancing security on Android systems.

Android Malware

When it comes to identifying malware on an Android device, there are multiple indicators that users and IT professionals should carefully observe. Occasionally, a performance issue, such as sluggishness, can be more than a mere annoyance and is caused by a malware infection. Malicious software frequently operates surreptitiously on a device, covertly utilizing data without the user's knowledge [18]. If an Android phone experiences a sudden surge in data usage or an abnormal depletion of its battery, it may have been infected with malware [19].

Malicious applications frequently establish themselves on mobile devices unbeknownst to users [20]. If users detect any newly installed applications on their mobile devices that they did not personally download, these applications may contain malicious code. An illustrative instance involves the surge of counterfeit ChatGPT applications inundating app stores, masquerading as Trojan horses, thereby infiltrating devices with malware and potentially pilfering files, text messages, call records, and other data [21].

Adware is a software application designed to exhibit undesirable advertisements on a device, usually in the form of intrusive pop-up windows or banners [22]. This not only annoys and decreases the efficiency of end users, but also depletes device resources, resulting in slowdowns. Furthermore, these pop-up advertisements can illicitly acquire personal information. If end users begin to encounter advertisements for

products and services unrelated to their search history, or encounter unfamiliar prompts requesting personal information, their device may have been infected with malware [23].

If a device experiences a sudden decrease in performance, it may be indicative of a malware infection. Certain forms of mobile malware are specifically engineered to execute actions that deplete device resources, such as CPU and memory, resulting in device slowdown and, in certain instances, rendering it unresponsive [24]. By being cognizant of these indicators, users can promptly and precisely detect malware on their Android devices. If any of these indicators are detected, it is imperative to promptly address the issue by eliminating the malicious software and fortifying the device against potential future risks.

Related Works

Using a recently released dataset known as CICMalDroid2017, which is maintained by the Cyber Security Institute of Canada, the authors of the study [6] employed an ANN-based system for detecting Android malware. The results of the experiments demonstrated that by dividing an IP address into four numbers, a high level of accuracy (98.1%) can be achieved. The authors of the [17] study introduced DL-Droid, a deep learning system designed to identify malicious Android applications by employing dynamic analysis with dynamic input generation. A total of more than (30,000) applications, including both benign and malware, were tested on real devices. The experimental results demonstrated that the DL-Droid is capable of achieving a detection rate of up to (97.8%).

Using static analysis, the authors of the study [25] proposed Android malware detection using DT, SVM, K-Nearest Neighbor (KNN), and naive Bayes (NB). They recommended and checked for malicious nodes using over (10,000) Android applications. The results showed that the KNN demonstrated the best prediction rate

at (93%) for malware. In contrast, MaMaDroid, an Android malware detection system that examines the application's control flow graph, was suggested by the authors of the study [26]. Adaboost, Decision Tree, and 5-NN are among the ML models they've adopted and used. The findings demonstrated a detection rate of over (90%) against all assaults.

The authors of the study [27] suggested GUIDED RETRAINING, a technique based on supervised representation learning, to improve malware detector performance by dividing samples into "easy" and "difficult" categories. There is a high rate of mistakes in the classifier's predictions when dealing with challenging samples because the probabilities are low. Next, we used the GUIDED RETRAINING method to enhance the classification of the challenging samples. Since the error rate on the "easy" samples is low by design, the base malware detector is used to make the final predictions for that subset. They show that GUIDED RETRAINING can decrease malware detector prediction errors by up to (40.41) percent, and they validate their method on four Android malware detection approaches using more than (265,000) malicious and benign apps.

In [28], the authors proposed DeepAMD to defend against real-world Android malware using deep ANN. The DeepAMD outperformed other methods in detecting and identifying malware attacks on both Static and Dynamic layers, with (93.4%) accuracy for malware classification, (92.5%) for malware category classification, and (90%) for malware family classification. DeepAMD has the highest malware category and family classification accuracy on the Dynamic layer at (80.3%) and (59%), respectively.

Proposed Method

For the Malware Detection, we used three neural networks, LSTM, MLP, and RNN for reducing and detecting the threats and malware to enhance security over the Android systems using each of the Drebin and the MH-100K datasets. Figure 1 shows the general framework for the proposed method:

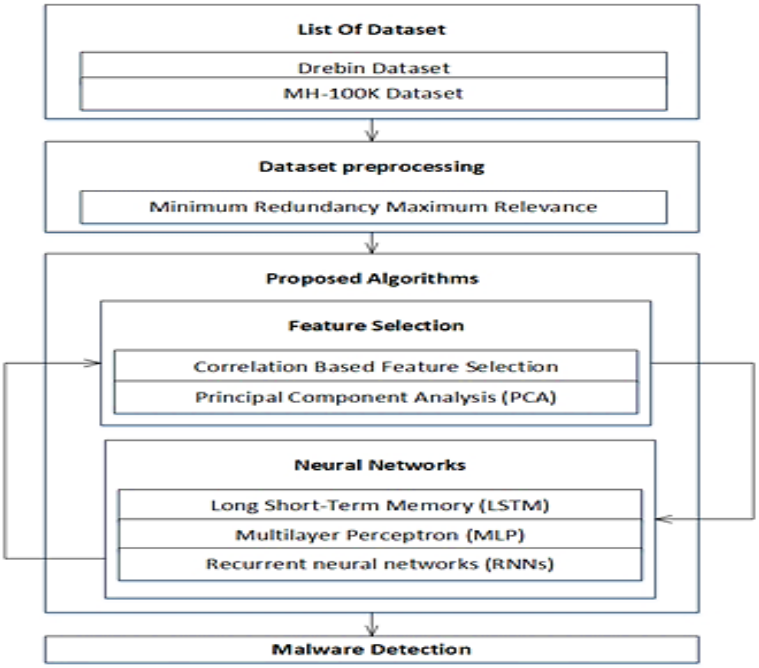

Fig. (1): General framework

A. Dataset:

In this study, we used two datasets to test the performance of the proposed method. The first dataset is the Drebin dataset [18], which contains feature vectors with (215), attributes extracted from a total of (15,036) applications. This includes (5,560) malware apps from the Drebin project and (9,476) benign apps.

The second dataset is the MH-100K dataset [3], which is a comprehensive compilation of Android malware data, consisting of (101,975) samples. The main CSV file contains important metadata, such as the SHA256 hash (APK's signature), file name, package name, Android's official compilation API, (166) permissions, (24,417) API calls, and (250) intents. Table 1 presents a description of the MH100K metadata data is provided:

The cryptographic hash "SHA256" is represented as a hexadecimal string. The term "NAME" pertains to the designation of the application. The term "PACKAGE" serves as a unique identifier for each application. The "TARGET_API" represents the API level that the application has been tested against, while the "MIN_API" specifies the minimum Android API level that is necessary for the program to work.

TABLE (I): List of metadata attributes and descriptions

Feature Name	Data Type	Attributes
SHA256	String	SHA256 cryptographic hash
NAME	String	Package Name
PACKAGE	String	Unique Application ID
MIN_API	Numeric	Minimum API Level Required to Run
TARGET_API	Numeric	API Level That the Application Targets

B. Data Pre-processing:

Dataset pre-processing involved the utilization of the Minimum Redundancy Maximum Relevance (MRMR) Algorithm. This algorithm was employed to identify an optimal set of features that are both mutually and maximally dissimilar, thereby effectively representing the response variable. The MRMR can process multivariate temporal data without the need for prior data flattening.

C. Feature Selection and Neural Networks:

The proposed method used three different neural networks (LSTM, MLP, and RNN). Each algorithm works separately and calls the sub-algorithms in the feature selection (Principal Component Analysis (PCA), and the Correlation Based Feature Selection

(CFS)). The CFS aims to create new subsets of features highly that contain a high correlated features with a specific class (Malware, Not Malware), and uncorrelated to each other (attributes) from both datasets. The PCA in the proposed method was used to identify all uncorrelated features from both datasets.

The Multilayer Perceptron (MLP) is a type of neural network that is fully connected and consists of multiple layers [29]. The MLP algorithm is an extension and subclass of the feed-forward neural network [30]. The structure comprises three distinct layers: the input layer, the output layer, and the hidden layer. The input layer operates by receiving the input signal that is intended for processing. The output layer is tasked with performing crucial functions, such as making predictions and carrying out classifications [31]. The Multilayer Perceptron (MLP) utilizes hidden layers, located between the input and output layers, as its main computational mechanism. In a multilayer perceptron (MLP), the data exhibits a comparable pattern to a feed-forward network, wherein it propagates in a unidirectional manner from the input layer to the output layer. The neurons in the Multilayer Perceptron (MLP) are trained using the backpropagation learning algorithm. Multi-layer perceptrons (MLPs) can accurately approximate any continuous function and effectively solve problems that cannot be solved using linear methods [32]. The main uses of Multilayer Perceptron (MLP) are pattern classification, recognition, prediction, and approximation.

The recurrent neural network (RNN) possesses the capability to execute a consistent operation for every element in a sequence, where the output is impacted by prior computations. Recurrent Neural Networks (RNNs) are characterized by the presence of bidirectional information flow. The input for the next time step is obtained from the output of the Recurrent Neural Network (RNN) and is subsequently reused. A feedforward neural network consists of an input layer, one or more hidden layers, and an output layer. The output of a network node is generated by applying a weight matrix to its inputs and then applying an activation function to the resulting values. A

backpropagation algorithm is used to train the network. The process entails computing the gradients for each weight in the neural network and subsequently adjusting each weight to attain the desired output of the network. Recurrent Neural Networks (RNNs) have backward connections, where the output of a specific layer is sent back to the same layer or a previous layer within the network. Recurrent Neural Networks (RNNs) utilize values calculated in a previous time step to impact calculations in the current time step, thereby maintaining an internal state. This state serves as a type of temporary memory within the network. Recurrent Neural Networks (RNNs) are commonly employed as a computational framework for examining time series and sequential data [33].

LSTM is a distinct variant of recurrent neural network (RNN). In the context of recurrent neural networks (RNNs), the output produced by the RNN in the previous time step is employed as the input for the current time step. The model being referred to is the Long Short-Term Memory (LSTM) model. As the gap length increases, the efficiency of the Recurrent Neural Network (RNN) decreases. The LSTM model has an inherent ability to efficiently retain information for long periods. This technology is utilized to examine, predict, and classify data that is arranged in chronological order [34].

D. Experiments:

The experiments in the proposed method depend on the two main scenarios, where the first scenario depends on a number of the attributes in both datasets (Drebin, and the MH-100K), while the second scenario depends on the number of epochs (5, 10, 20, 25, 30, 35, 40, 45, and 50) for each algorithm (LSTM, MLP, and RNN). Each scenario worked on each of the datasets (Drebin, and the MH-100K) as shown in Table 2.

International Journal of Computers and Informatics (IJCI)
Vol. (4), No. (3)
IJCI
March 2025
المجلة الدولية للحاسبات والمعلوماتية
الإصدار (4)، العدد (3)

TABLE (2): Experiments scenarios

Method	Dataset	Attributes	Epochs
LSTM	Drebin	100	5, 10, 20
		150	25, 30, 35
		215	40, 45, 50
	MH-100K	100	5, 10, 20
		150	25, 30, 35
		250	40, 45, 50
MLP	Drebin	100	5, 10, 20
		150	25, 30, 35
		215	40, 45, 50
	MH-100K	100	5, 10, 20
		150	25, 30, 35
		250	40, 45, 50
RNN	Drebin	100	5, 10, 20
		150	25, 30, 35
		215	40, 45, 50
	MH-100K	100	5, 10, 20
		150	25, 30, 35
		250	40, 45, 50

For the results evaluation, we used the confusion matrix for each scenario. The confusion matrix was utilized to assess the accuracy of the classifiers, and the F1-Score was evaluated. Accuracy was measured using either Precision or Recall, while F1-Score was specifically used for imbalanced data. Below are the formulas for the performance evaluation metrics used.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

International Journal of Computers and Informatics, London Vol (4), No (3), 2025
https://doi.org/10.59992/IJCI.2025.v4n3p2 E-ISSN 2976-9361

$$\text{F Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Where the TP: Predicted Positive, Actual Positive; TN: Predicted Negative, Actual Negative; FP: Predicted Positive, Actual Negative; and FN: Predicted Negative, Actual Positive.

Results and Discussion

According to the experiment scenarios, this section presents the results for the proposed algorithms (MLP, RNN, and LSTM) for Malware Detection using each of the Drebin and the MH-100K datasets.

A. MLP RESULTS:

Table 3, and Table 4 show the Recall, Precision, and F-Measure results for the MLP algorithm based on the number of features (100, 150, and 215), and using each of the Drebin and the MH-100K datasets respectively:

TABLE (3): MLP ALGORITHM RESULTS WITH THE DREBIN DATASET

Method	Dataset	Attributes	Recall	Precision	F1
MLP	Drebin	100	0.9520	0.9406	0.9462
MLP	Drebin	150	0.9558	0.9401	0.9479
MLP	Drebin	All	0.9545	0.9487	0.9516

TABLE (4): MLP algorithm results with the MH-100K dataset

Method	Dataset	Attributes	Recall	Precision	F1
MLP	MH-100K	100	0.9647	0.9716	0.9682
MLP	MH-100K	150	0.9643	0.9738	0.9690
MLP	MH-100K	All	0.9697	0.9732	0.9715

Rate of F-Measure for each of the Drebin and the MH-100K datasets, where the F-Measure with the Drebin dataset reached (94.62%) using (100) attributes, and reached (96.82%) with the MH-100K dataset using the same number of attributes. While the F-Measure with the Drebin dataset reached (94.79%) using (150) attributes, and reached (96.9%) with the MH-100K dataset using the same number of attributes.

Finally, the F-Measure with the Drebin dataset reached (95.16%) using (ALL) attributes, and reached (97.15%) with the MH-100K dataset using the same number of attributes.

We can notice also the preferences of the MH-100K dataset compared to the Drebin dataset for detecting the malware. Figure 2 shows the overall accuracy of the MLP algorithm for detecting the malware in both datasets:

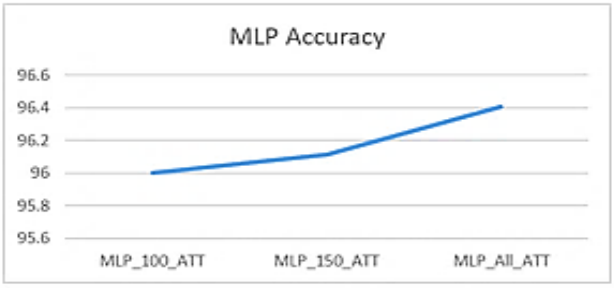

Fig. (2): MLP Accuracy

Finally, we can notice the performance of the MLP algorithm for detecting the malware increased with the increase of the attribute numbers in both datasets, where the accuracy reached (96%) (96.11%) (96.4%) using the (100, 150, and ALL) of attributes respectively.

B. RNN Results:

Table 5, and Table 6 show the Recall, Precision, and F-Measure results for the RNN algorithm based on the number of features (100, 150, and 215), and using each of the Drebin and the MH-100K datasets respectively:

TABLE (5): RNN Algorithm Results with the Drebin Dataset

Method	Dataset	Attributes	Recall	Precision	F1
RNN	Drebin	100	0.9571	0.9792	0.9680
RNN	Drebin	150	0.9586	0.9818	0.9701
RNN	Drebin	All	0.9648	0.9848	0.9747

International Journal of Computers and Informatics (IJCI)
Vol. (4), No. (3)
IJCI
March 2025
المجلة الدولية للحاسبات والمعلوماتية
الإصدار (4)، العدد (3)

TABLE (6): RNN algorithm results with the MH-100K dataset

Method	Dataset	Attributes	Recall	Precision	F1
RNN	MH-100K	100	0.9881	0.9752	0.9816
RNN	MH-100K	150	0.9896	0.9761	0.9828
RNN	MH-100K	All	0.9913	0.9796	0.9854

According to the RNN results, we can notice the increasing rate of F-Measure for each of the Drebin and the MH-100K datasets, where the F-Measure with the Drebin dataset reached (96.8%) using (100) attributes, and reached (98.16%) with the MH-100K dataset using the same number of attributes. While the F-Measure with the Drebin dataset reached (97.01%) using (150) attributes, and reached (98.28%) with the MH-100K dataset using the same number of attributes. Finally, the F-Measure with the Drebin dataset reached (97.47%) using (ALL) attributes, and reached (98.54%) with the MH-100K dataset using the same number of attributes.

We can notice also the preferences of the MH-100K dataset compared to the Drebin dataset for detecting the malware. Figure 3 shows the overall accuracy of the RNN algorithm for detecting the malware in both datasets:

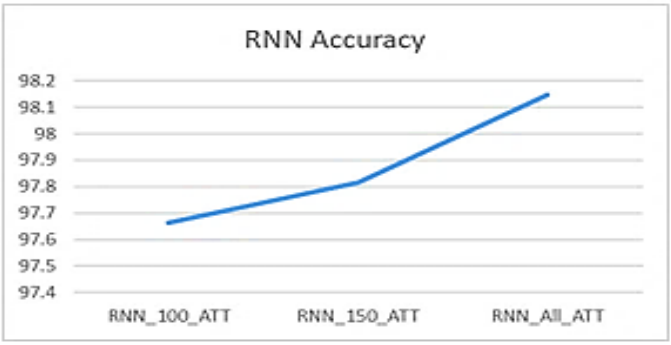

Fig. (3): RNN Accuracy

We can notice the performance of the RNN algorithm for detecting the malware increased with the increase of the attribute numbers in both datasets, where the

accuracy reached (97.66%) (97.81%) and (98.14%) using the (100, 150, and ALL) of attributes respectively.

C. LSTM Results:

Table 7, and Table 8 show the Recall, Precision, and F-Measure results for the LSTM algorithm based on the number of features (100, 150, and 215), and using each of the Drebin and the MH-100K datasets respectively:

TABLE (7): LSTM algorithm results with the Drebin dataset

Method	Dataset	Attributes	Recall	Precision	F1
LSTM	Drebin	100	0.9573	0.9808	0.9689
LSTM	Drebin	150	0.9651	0.9830	0.9739
LSTM	Drebin	All	0.9669	0.9874	0.9770

TABLE (8): LSTM algorithm results with the Drebin dataset

Method	Dataset	Attributes	Recall	Precision	F1
LSTM	MH-100K	100	0.9890	0.9753	0.9821
LSTM	MH-100K	150	0.9902	0.9797	0.9849
LSTM	MH-100K	All	0.9928	0.9808	0.9867

According to the LSTM results, we can notice the increasing rate of F-Measure for each of the Drebin and the MH-100K datasets, where the F-Measure with the Drebin dataset reached (96.89%) using (100) attributes, and reached (98.21%) with the MH-100K dataset using the same number of attributes. While the F-Measure with the Drebin dataset reached (97.39%) using (150) attributes, and reached (98.49%) with the MH-100K dataset using the same number of attributes. Finally, the F-Measure with the Drebin dataset reached (97.7%) using (ALL) attributes, and reached (98.67%) with the MH-100K dataset using the same number of attributes.

By similarity with the MLP and the RNN results, we can notice also the preferences of the MH-100K dataset compared to the Drebin dataset for detecting the malware. Figure 4 shows the overall accuracy of the LSTM algorithm for detecting the malware in both datasets:

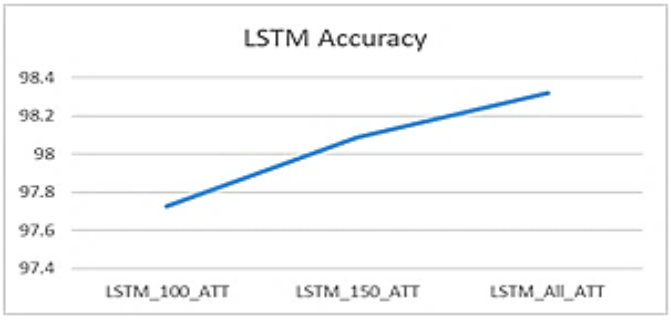

Fig. (4): LSTM Accuracy

Finally, we can notice the performance of the LSTM algorithm for detecting the malware increased with the increase of the attribute numbers in both datasets, where the accuracy reached (97.72%) (98.09%) (98.31%) using the (100, 150, and ALL) of attributes respectively.

Conclusion

This study aimed to investigate the signs of malware on Android devices and to develop a malware detection model for Android systems based on the Drebin and the MH-100K datasets. We used each of the Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), and Recurrent neural networks (RNNs) for reducing and detecting the threats and malware to enhance security over the Android systems. We used several scenarios for testing the performance of each algorithm.

The experiment results showed the preference of results for the MH-100K dataset compared to the Drebin dataset. On the other hand, the results showed that the LSTM algorithm outperforms both the MLP and the RNN algorithms for malware detection for both datasets. For future work, we are looking to use more datasets with other algorithms and investigate the most effective methods and datasets that increase malware detection.

References

1. B. Tahtaci and B. Canbay, "Android Malware Detection Using Machine Learning," Proc. - 2020 Innov. Intell. Syst. Appl. Conf. ASYU 2020, 2020, doi: 10.1109/ASYU50717.2020.9259834.
2. H. Papadopoulos, N. Georgiou, C. Eliades, and A. Konstantinidis, "Android malware detection with unbiased confidence guarantees," Neurocomputing, vol. 280, pp. 3–12, 2018, doi: 10.1016/j.neucom.2017.08.072.
3. H. Bragança, V. Rocha, L. Barcellos, E. Souto, D. Kreutz, and E. Feitosa, "Android malware detection with MH-100K: An innovative dataset for advanced research," Data Br., vol. 51, 2023, doi: 10.1016/j.dib.2023.109750.
4. M. Altaİy, İ. Yildiz, and B. Uçan, "MALWARE DETECTION USING DEEP LEARNING ALGORITHMS," vol. 7, no. 1, pp. 11–26, 2023.
5. A. Hota and P. Irolla, "Deep Neural Networks for Android Malware Detection," Int. Conf. Inf. Syst. Secure. Priv., no. Icissp, pp. 657–663, 2019, doi 10.5220/0007617606570663.
6. E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "Neural Network Based Android Malware Detection with Different IP Coding Methods," HORA 2021 - 3rd Int. Congr. Human-Computer Interact. Optim. Robot. Appl. Proc., no. November 2020, 2021, doi: 10.1109/HORA52670.2021.9461302.
7. H. Rathore, S. K. Sahay, P. Nikam, and M. Sewak, "Robust Android Malware Detection System Against Adversarial Attacks Using Q-Learning," Inf. Syst. Front., vol. 23, no. 4, pp. 867–882, 2021, doi: 10.1007/s10796-020-10083-8.
8. R. Kumar, W. Wang, J. Kumar, Zakria, T. Yang, and W. Ali, "Collective Intelligence: Decentralized Learning for Android Malware Detection in IoT with Blockchain," no. i, pp. 1–15, 2021, [Online]. Available: http://arxiv.org/abs/2102.13376.
9. T. Sun, W. Pian, N. Daoudi, K. Allix, T. F. Bissyandé, and J. Klein, "LaFiCMIL: Rethinking Large File Classification from the Perspective of Correlated Multiple Instance Learning," no. Mil, 2023, [Online]. Available: http://arxiv.org/abs/2308.01413.
10. H. Rathore, S. K. Sahay, S. Thukral, and M. Sewak, "Detection of Malicious Android Applications: Classical Machine Learning vs. Deep Neural Network Integrated with

Clustering," Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, vol. 355, pp. 109–128, 2021, doi: 10.1007/978-3-030-68737-3_7.

11. F. Taher, O. Al Fandi, M. Al Kfairy, H. Al Hamadi, and S. Alrabaee, "A Proposed Artificial Intelligence Model for Android-Malware Detection," Informatics, vol. 10, no. 3, 2023, doi: 10.3390/informatics10030067.
12. H. Rathore, S. K. Sahay, R. Rajvanshi, and M. Sewak, "Identification of Significant Permissions for Efficient Android Malware Detection," Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, vol. 355, pp. 33–52, 2021, doi: 10.1007/978-3-030-68737-3_3.
13. A. Muzaffar, H. R. Hassen, H. Zantout, and M. A. Lones, "A Comprehensive Investigation of Feature and Model Importance in Android Malware Detection," pp. 1–18, 2023, [Online]. Available: http://arxiv.org/abs/2301.12778.
14. H. Bostani, Z. Zhao, Z. Liu, and V. Moonsamy, "Level Up with RealAEs: Leveraging Domain Constraints in Feature Space to Strengthen Robustness of Android Malware Detection," no. Ml, 2022, [Online]. Available: http://arxiv.org/abs/2205.15128.
15. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "Graph Neural Network-based Android Malware Classification with Jumping Knowledge," 5th IEEE Conf. Dependable Secur. Comput. DSC 2022 SECSOC 2022 Work. PASS4IoT 2022 Work. SICSA Int. Pap. Compet. Cybersecurity, no. June 2022, doi: 10.1109/DSC54232.2022.9888878.
16. B. Molina-coronado, A. Ruggia, U. Mori, A. Merlo, A. Mendiburu, and J. Miguel-alonso, "Light up that Droid ! On the Effectiveness of Static Analysis Features against App Obfuscation for Android Malware Detection," pp. 1–16.
17. M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," Comput. Secur., vol. 89, 2020, doi 10.1016/j.cose.2019.101663.
18. Daniel Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," J. Jpn. Stud., vol. 36, no. 1, pp. 165–169, 2014, doi: 10.1353/jjs.0.0130.
19. M. S. Saleem, J. Mišić, and V. B. Mišić, "Android Malware Detection using Feature Ranking of Permissions," 2022, [Online]. Available: http://arxiv.org/abs/2201.08468.

20. Z. Yang, F. Deng, and L. Han, "Flexible Android Malware Detection Model based on Generative Adversarial Networks with Code Tensor," Proc. - 2022 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2022, pp. 19–28, 2022, doi: 10.1109/CyberC55534.2022.00015.

21. M. Al-Hawawreh, A. Aljuhani, and Y. Jararweh, "Chatgpt for cybersecurity: practical applications, challenges, and future directions," Cluster Comput., vol. 26, no. 6, pp. 3421–3436, 2023, doi: 10.1007/s10586-023-04124-5.

22. S. Suresh, F. Di Troia, K. Potika, and M. Stamp, "An analysis of Android adware," J. Comput. Virol. Hacking Tech., vol. 15, no. 3, pp. 147–160, 2019, doi: 10.1007/s11416-018-0328-8.

23. M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, and R. Maskeliunas, Android Malware Detection: A Survey, vol. 942. Springer International Publishing, 2018.

24. M. A. Omer et al., "Efficiency of Malware Detection in Android System: A Survey," Asian J. Res. Comput. Sci., no. April, pp. 59–69, 2021, doi: 10.9734/ajrcos/2021/v7i430189.

25. H. Babbar, S. Rani, D. K. Sah, S. A. AlQahtani, and A. Kashif Bashir, "Detection of Android Malware in the Internet of Things through the K-Nearest Neighbor Algorithm," Sensors, vol. 23, no. 16, pp. 1–17, 2023, doi: 10.3390/s23167256.

26. H. Berger, C. Hajaj, E. Mariconti, and A. Dvir, "MaMaDroid2.0 -- The Holes of Control Flow Graphs," 2022, [Online]. Available: http://arxiv.org/abs/2202.13922.

27. N. Daoudi, K. Allix, T. F. Bissyandé, and J. Klein, "A two-steps approach to improve the performance of Android malware detectors," 2022, [Online]. Available: http://arxiv.org/abs/2205.08265.

28. S. I. Imtiaz, S. ur Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network," Futur. Gener. Comput. Syst., vol. 115, pp. 844–856, 2021, doi: 10.1016/j.future.2020.10.008.

29. Janke, J., Castelli, M., & Popovič, A. (2019). Analysis of the proficiency of fully connected neural networks in the process of classifying digital images: Benchmark of different classification algorithms on high-level image features from convolutional layers. Expert Systems with Applications, 135, 12–38. https://doi.org/10.1016/j.eswa.2019.05.058.

30. Gumbarević, S., Milovanović, B., Gaši, M., & Bagarić, M. (2020). Application of Multilayer Perceptron Method on Heat Flow Meter Results for Reducing the Measurement Time. 29. https://doi.org/10.3390/ecsa-7-08272.

31. Al-Saif, A. M., Abdel-Sattar, M., Aboukarima, A. M., & Eshra, D. H. (2021). Application of a multilayer perceptron artificial neural network for identification of peach cultivars based on physical characteristics. PeerJ, 9, 1–22. https://doi.org/10.7717/peerj.11529.

32. Saha, S., Paul, G. C., Pradhan, B., Abdul Maulud, K. N., & Alamri, A. M. (2021). Integrating multilayer perceptron neural nets with hybrid ensemble classifiers for deforestation probability assessment in Eastern India. Geomatics, Natural Hazards and Risk, 12(1), 29–62. https://doi.org/10.1080/19475705.2020.1860139.

33. Ibrahim, M., & Elhafiz, R. (2023). Modeling an intrusion detection using recurrent neural networks. Journal of Engineering Research, 11(1), 100013. https://doi.org/10.1016/j.jer.2023.100013.

34. Rahman, S. M., Pawar, S., San, O., Rasheed, A., & Iliescu, T. (2019). Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. Physical Review E, 100(5). https://doi.org/10.1103/PhysRevE.100.053306.