

AMaLSTM: ANDROID MALWARE DETECTION USING LSTM

Mehwish Aslam

Department of Computer Science, NFCIET,
Multan, Pakistan

Ahmad Naeem

Department of Computer Science, NFCIET,
Multan, Pakistan

Ahmad Sarfraz

Department of Computer Science, NFCIET,
Multan, Pakistan

Muhammad Kamran Abid*

Department of Computer Science, NFCIET,
Multan, Pakistan

Yasir Aziz

Department of Computer Engineering, BZU,
Multan, Pakistan

Naeem Aslam

Department of Computer Science, NFCIET,
Multan, Pakistan

Muhammad Fuzail

Department of Computer Science, NFCIET,
Multan, Pakistan

*Corresponding author: Muhammad Kamran Abid (kamran.abid@nfciet.edu.pk)

Article Info



This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license
<https://creativecommons.org/licenses/by/4.0>

Abstract

Android smartphone apps are becoming increasingly popular, but their security is a concern. Malware can cause damage to mobile devices and servers. Developing detection technologies to avoid attacks is crucial for protecting consumers' mobile devices, desktops, and servers. The goal of this study is to avoid malware attacks, which are addressed via static, dynamic and hybrid features. Combating such attacks requires effective malware detection tools. The framework restricts the deep learning architecture in order to find dependencies between APK-retrieved attributes. To examine the performance and robustness of our proposed system, we conducted a comprehensive experimental investigation that included machine learning and deep learning approaches. This study assesses the efficiency of LSTM for identifying Android malware appears in time-varying sequences of healthy and infected apps. To evaluate the AMaLSTM model, a set of malicious and benign Android applications, along with their package files containing features such as API calls, system call sequences, opcode sequences and permissions, are used. Nevertheless, the dataset has a balanced malware set for different types, but it does not cover the whole nature of Android malware and available tooling for producing new threats. Android malware classification accuracy is evaluated by means of deep learning models trained on the AMaLSTM framework. It works well on classification accuracy at low false positive and positive false negatives, thus being more favourable towards evasive virus tactics. The results demonstrate that the proposed approach exceeds earlier algorithms with detection accuracy: 98.4%, precision: 98.5%, recall: 97.2%, and F1 measure: 97.8%. Our future study is to apply LSTM network topologies to genuine Android malware samples, rather than static permissions or profiled program attributes.

Keywords:

Android malware detection, deep learning model, malware classifier, AMaLSTM

Introduction

A highly popular mobile device platform offering numerous conveniences like texting, video chat, socializing through social media, use of internet and multiple IoT apps includes Android. It as an open-source system, and the complexity of calls both system and API, make it attractive for viruses and make identification of malicious processes themselves challenging. This had led to open source platform being targeted by virus makers for instance its continuous attack force has shifted research towards the examination of risky android applications. By virtue of reasoning that software from the same family often takes similar actions, malware analysts are crucial in differentiating the malware families. Consequently, knowing the type of malware becomes even more important with this research revealing the existence of more bad software.[1]. Mobile phones are increasingly important for online shopping, instant messaging, and payments, but they also pose security risks. The open-source Android platform allows for easy creation of malware, which can send phony SMS, consume data, steal personal information, download malicious apps, and access remote control, threatening users' property and privacy [2].

Smartphone malware on Android platforms, including Banking Trojans and automatic messaging exploits, poses a significant security threat, necessitating efficient detection and protection of users' assets due to its widespread use [3]. Anti-malware programs like McAfee and Kingsoft safeguard users from malware by utilizing signature-based techniques, which match malicious programs in databases [4]. Mobile phones enable easy application design across industries, but also expose them to cyberattacks. Android-based applications saw the highest downloads in 2021, posing risks to data confidentiality, system availability, and integrity [5].

Malware, designed to steal data and gain root access, is increasing due to the internet's expansion and software industry. Anti malware systems is an important area of study, and there is vast literature on this subject [6]. The Android platform has been attacked due to its popularity and relative openness to other programming environments. In the last few years, Android malwares have become increasingly common which poses great danger to Android security. Attackers have attacked the Android platform due of its increasing popularity and openness. The prevalence of Android malware has skyrocketed in recent years, endangering Android security in the process. In this way, by proposing measures for the accurate identification of the presence of Android malware [7], it is possible to eradicate malware.

Based on the approaches the Android malware has been detected by using different methods which include signature-based [8]-[9] static analysis [10] [11] and dynamic analysis [12] [13].. Static analysis for detecting Android malware is a behavior-based technique. It examines how apps functional without even running the apps on a mobile device or an emulator. This approach is the favourite because of its ease and low implementation complexity and time as compared with other methods. Static features are analyzed by employing a reverse engineering toolkit, which includes APKTool [14] JADX [15] Dex2Jar [16] Android Studio [17] and Andro guard [18]. In contrast, dynamic analysis involves running a program on an emulator or device to imitate real-world behavior. This allows monitoring and recording of an application's runtime operations, including Operating system activity, network connectivity, and any actions that resemble any form of harmful activity. Some studies [19]-[20] utilized a hybrid analysis technique, developing strategies for developing malware detection models with reference to static and dynamic data. Android malware is continuously changing, and attackers use clever ways to avoid detection. To successfully address this expanding threat, improved real-time detection systems are required.

Several scholars, including [22] Machine learning methods include SVMs, Naïve Bayesian networks, RFs, multilayer perceptron's, and decision trees have been utilized to detect malware in Android applications, as reviewed in several studies [23]-[24] on Machine learning-based malware detection models use data from Android applications for training, testing, and validation. Researchers are using deep learning techniques, a sub-field of machine learning within AI, to combat harmful behavior by using neural networks with multiple parameters and layers. The study uses LSTM architecture to accurately identify malware on Android devices, improving model Interpretability, user trust, and integrating it with existing

security infrastructure. Static or dynamic or both types of techniques are used by researchers for studying mobile malware, where features are derived from the manifest files, code execution, and simulation which indeed yield excellent results when compared to the entire study based on individual techniques. Using this foundation, our method improves upon the prior technique as it achieves 99.94% of the effectiveness for identifying malware in recent datasets. [21].

The primary contributions of our research work are outlined as follows:

- (1) We suggested an LSTM-based malware detection model to detect constantly emerging malware instances. Our technique is not just highly accurate for existing malware, but also for freshly discovered malware.
- (2) For the first time, we retrain a malware detection model utilizing long-short term memory, allowing it to battle newly created malware with unclear features.
- (3) Our method's accuracy has improved significantly over the prior method, which obtained 99.94% accuracy in malware detection on current data sets.

Related Work

Researchers have utilized machine learning and deep learning approaches to create malware and intrusion detection systems for Android because of its widely used platform and open source code [25]-[26]. The majority of them focused on mobile malware datasets, but few used desktop ones.

A. Alzubaidi et al. (2021) in [27] discuss the growing threat of mobile malware on Android platforms and the current methods of detection, including machine learning (ML), deep learning (DL), and ensemble learning. They found ML-based techniques are most effective in detecting malware, while DL-based methods like CNN and RNN are effective in detecting fake apps.

Vinayakumar et al. [28] utilized machine learning and deep learning methods for detecting malware using the Ember malware dataset. Additionally, when using this approach, might not be easy to deal with obfuscated code and zero day malware. Both static and dynamic analysis of Android applications and its features extracted from static code and dynamic behaviour were implemented using an LSTM architecture. The LSTM model produced the highest accuracy (98.9%) after 200 epochs of training [29]. Amer and El-Sappagh [19] presented a behavioral predicted model of Android malware that incorporates static and dynamic characteristics. The deep learning LSTM model is utilized to organize API and system call sequences. Furthermore, ensemble machine-learning approaches are employed to classify Android permissions.

Alkahtani and Aldhyani [30] propose a comparative analysis of malware detection on Android is done utilizing a variety of machine and deep learning methods. Depending on the outcome of the assessment reset, SVM, LSTM, and CNN-LSTM approaches are more successful at detecting malware that targets the Android platform.

Ban et al. [31] utilized convolutional neural networks for Android virus detection. The researchers utilized a malware dataset with 28,179 records from 2018 to 2020 to identify the most prevalent malware actions. The experiments showed 98% accuracy with a F1-score of 0.82.

Smamarwar et al. [32] proposed "wrapping feature selection" (WFS) approach. A few of them were able to employ techniques such as random forest, decision trees and SVM classifiers. The classifiers are built with the optimal number of attributes originating from the CIC-InvesAndMal2019 malware dataset. Trials demonstrated the SVM, RF, and DT models attained accuracy rates of 82.33%, 91.32%, and 91.8%, respectively.

Alshahrani et al. [33] created DDefender, malware detection system. This solution detects fraudulent Android apps using static and dynamic analysis as well as deep neural networks. Dynamic analysis extracts features including system data, network traffic, system calls, and permissions, while static analysis extracts key application components. The DDefender was trained and evaluated against 2.1k harmful and 2.1k benign applications. DDefender's evaluation findings indicate 95% accuracy.

Lê et al. [34] present a detecting approach that uses machine learning that uses permissions and API attributes. Nevertheless, this method excludes installing the framework on Android smartphones.

Islam et al. [35] proposed a machine learning ensemble model for detecting Android malware employs dynamic feature analysis. However, the study did not ensure malware real-time detection is used. Bhat et al. [13] to detect malware, dynamic behavior analysis was employed in conjunction using system-centric characteristics and a stacking ensemble method.

Methodology

Overview of Proposed Approach

This paper describes the Long Short-Term Memory (LSTM) network concept for identifying Android malware more effectively. It uses raw data from sources like API call sequences and network traffic patterns, preprocessing for model training, encoding categorical features, normalizing data, and handling missing or incomplete values.

Due to feature selection, the LSTM network is made more accurate and efficient by reducing the idling components. Statistical analysis and machine learning techniques are employed to train model on informative data points. Adequate dataset partitioning offers a sound model for the discovery of malware in the real world.

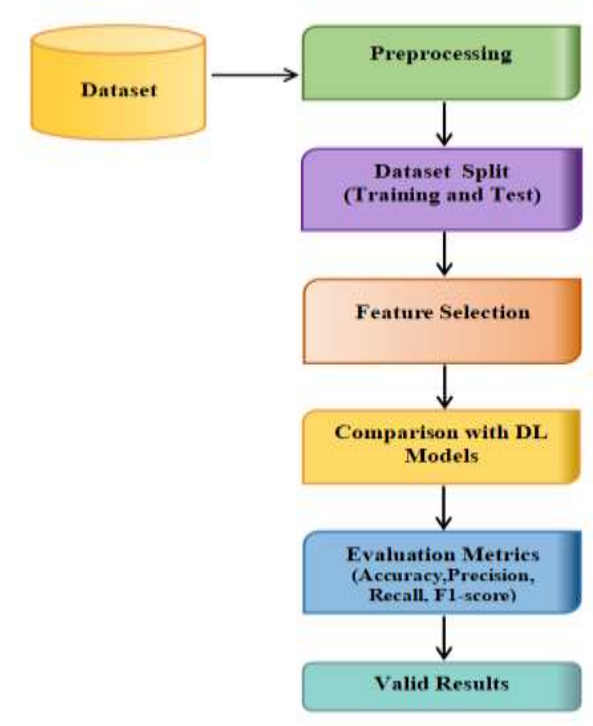


Figure 1: Proposed Framework

Dataset Description:

The Drebin Dataset is used in the current analysis and it is obtained from Kaggle which is an online platform where people share machine learning projects and data. The dataset may be accessed at the following link: [/Kaggle/input/Drebin-dataset/drebin.csv](https://www.kaggle.com/input/Drebin-dataset/drebin.csv)

It contains elaborated explanation of all the technical terms and references used regarding application development in Android as well as management of the system and its interaction with the hardware device.. Key components are services and binders for process communication, and Android binder and system level of operations for cross-process communication. Permissions and Telephony include sending, reading, and receiving SMS messages, accessing phone state, managing user accounts, and providing access to telephony services. Intent actions include broadcasting when the device has finished booting, sending data, and managing package changes.

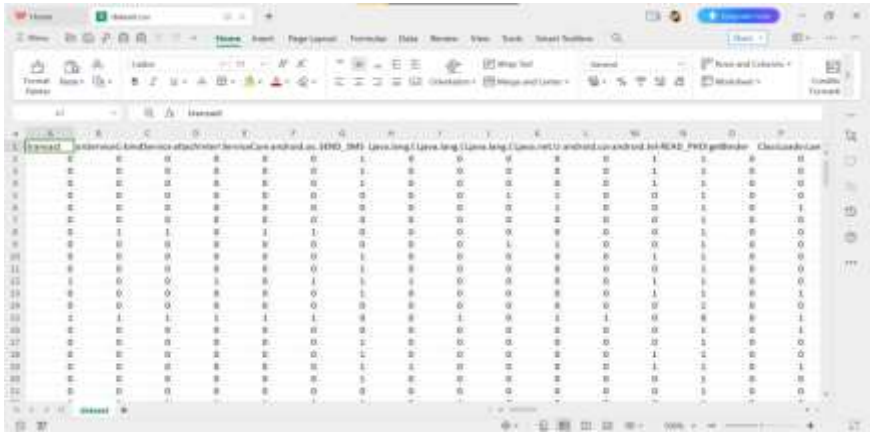
The image shows a screenshot of a spreadsheet application, likely Microsoft Excel, displaying the Drebin dataset. The spreadsheet has a grid of data with columns and rows. The first column contains numerical values, possibly representing sample IDs. The subsequent columns contain various features extracted from Android applications, such as permissions, API levels, and other metadata. The data is organized in a structured manner, with rows representing individual samples and columns representing different features. The spreadsheet interface includes a menu bar at the top with options like File, Home, Insert, Page Layout, Formulas, Data, Review, View, Tools, and Send Feedback. The status bar at the bottom shows the current selection and some statistics.

Figure 2: Drebin Dataset

Various Models: This stage represents the core of the deep learning model. It likely involves a Convolutional Neural Network (CNN), an Artificial Neural Networks (ANN), Radial Basis Function (RBF) Networks and AdaBoost.

CNN: CNN architecture is effective in sentence-level classification, sentiment analysis, and question categorization, using convolution layers for high-level features and pooling layers for feature reduction. It detects Android malware using fully linked layers and Soft max layers for distinct families.

Artificial Neural Networks (ANNs) are fully connected models that can learn complex patterns. Although versatile, they may not capture the temporal dependencies necessary for accurate viral detection.

Radial Basis Function (RBF) Networks: A type of ANN that uses radial basis functions for activation. While effective for pattern recognition, these methods rely heavily on kernel selection and parameters and struggle with large datasets required in malware detection.

AdaBoost is a potent ensemble learning technique to construct a strong classifier by using some weak classifiers. All training samples are first assigned the same weights, and then weak classifiers are iteratively trained according to changing weights in order to put more focus on misclassified samples. The core idea is to focus on difficult data to benefit later classifier’s performance.

The classifier’s error (ϵ) at each iteration is calculated and its weight (α) is given by:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Sample weights are then updated:

$$w_i = w_i \times e^{-\alpha y_i h(x_i)}$$

It is a weighted sum of weak classifiers whose final strong classifier:

$$H(\epsilon) = \text{sign} \left(\sum \alpha_t h_t(x) \right)$$

Long Short-Term Memory (LSTM) Network: RNN has been used in identifying Android malware while LSTM is appropriate for analyzing and predicting time series. The LSTM model presents a memory cell structure with three gates: It has input modulation and input connection along with modulators of individual output and connection to a modulation that may be referred as the input gate, forget gate and the output gate. In fact, the LSTM structure is made up of memory chunks.

These gates and states are defined as below:

Input gate:

Input gate:

$$it = \sigma(Wx_{it} + Wh_{it} - 1 + b_i)$$

Forget gate:

$$ft = \sigma(Wx_{ft} + Wh_{ft} - 1 + b_f)$$

Output gate:

$$ot = \sigma(Wx_{ot} + Who_{ht} - 1 + b_o)$$

Input Transform:

$$c_imt = \tanh(Wxc_{xt} + Whc_{ht} - 1 + b_{cim})$$

State update:

$$it = f_i \otimes ct - 1 + it \otimes c_imt$$

$$ht = ot \otimes \tanh(ct)$$

where x_t represents the input feature vector at time step t , W and b represent the weights and bias, respectively, and σ is the twisted activation function; \otimes is the element-wise product, and the hidden layer output and memory cell are indicated by ct and ht .

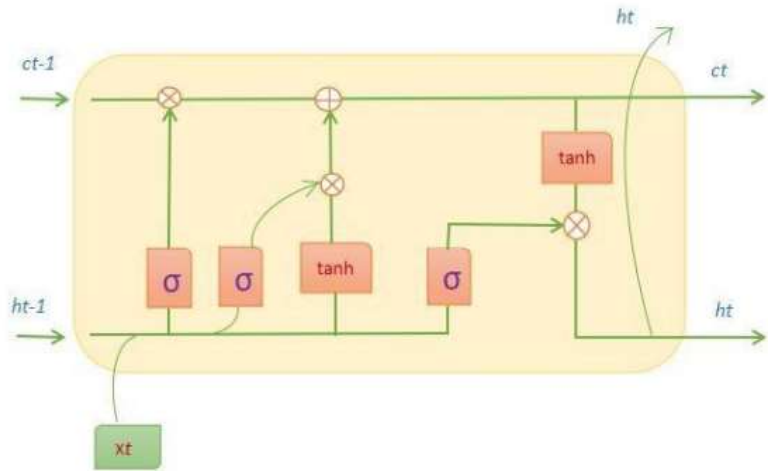


Figure 3: The fundamental composition of a long short-term memory cell.

The input gate in LSTM networks allows the network to selectively recall or forget knowledge from earlier time steps by controlling the quantity of fresh information added to the cell state. It also restricts the quantity of data forgotten from the cell state, allowing the network to discover enduring dependencies. The output gate manages the quantity of data utilized to produce the output of the LSTM cell, capturing relevant information from the input sequence. The input transform, also known as the candidate hidden state or cell update, computes the new cell state at each time step

Results and Discussion.

In this section, Deep-learning experiments were conducted using DNN, CNN, LSTM, AdaBoost, and RF models to evaluate their usefulness and performance taking into account parameters such as F1-score, recall, accuracy, loss, precision and AUC-PR. The following sections provide details on each experiment and its outcomes.

Experimental Setup:

The study utilized advanced deep learning algorithms to recognize facial expressions, using Google Collab’s cloud-based platform for efficient resource allocation and reduced local hardware requirements. The model for detecting Android malware uses critical hyperparameters and preprocessing processes, with a learning rate defaulted to 0.001, 64 batch sizes, and a standard scaling of 'Standard Scaler'. The model's architecture consists of 64 and 128 unit LSTM layers, 0.2 dropout layers, and a dense layer with sigmoid activation. Data preparation involves transforming non-numeric values to NaN and label encoding.

Comparison to other methodologies:

This paper proposes LSTM model architecture for malware detection on Android and tests on benchmark dataset with respect to pre-trained model and other machine learning algorithms. The effectiveness of the model as applied to an array of tests, as well as the tweaking of hyperparameters, demonstrates the success of the model.

Table 1: Comparison with Proposed Model

Model	Accuracy	F1-score	Precision	Recall
CNN	0.9837	0.9782	0.9786	0.9777
ANN	0.9844	0.9791	0.9786	0.9795
RF	0.9867	0.9819	0.9945	0.9697
RBF	0.9800	0.9869	0.9874	0.9865
AdaBoost	0.9727	0.9632	0.9710	0.9554
Proposed Approach LSTM	0.9844	0.9789	0.9855	0.9724

Results of CNN model

The performance summary of the CNN approach in detecting malware in Android is provided in the table below.

Table 2: Accuracy of CNN Model

Metrics	Accuracy	Precision	Recall	F1-Score
Values	0.983	0.978	0.979	0.978

It is clear that the CNN model can be used to detect Android malware with good accuracy, good precision and good recall. However, if the exact deployment scenario requires a large number of iteration, then computational cost might be a major issue.

Confusion Matrix

The CNN model’s efficiency is affirmatively confirmed with the help of the confusion matrix that reveals high True Negatives and True Positives as well as low False Positives and False Negatives proving that It can distinguish between toxic and non-toxic applications while keeping the degree of error low and the level of accurate classification high.

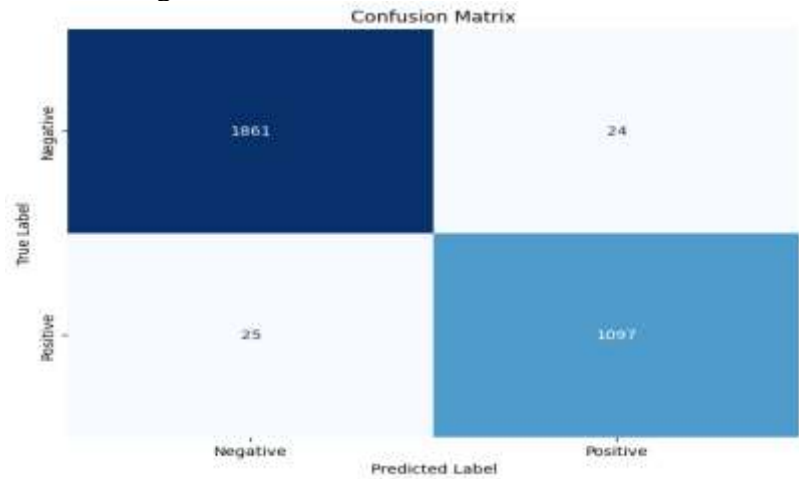


Figure 4 : Confusion Matrix of CNN Model

Training and Validation Accuracy:

This model’s learning curves of training and validation accuracy reveal improvement of the training and can generalize to new data sets with low training and validation loss. The model introduced here is not underfitting or overfitting, which contributes to the accuracy and stability of Android malware detection. The given model of FIG. 9 shows reasonable training and validation loss which indicate the successful training of the model without the problem of overfitting.

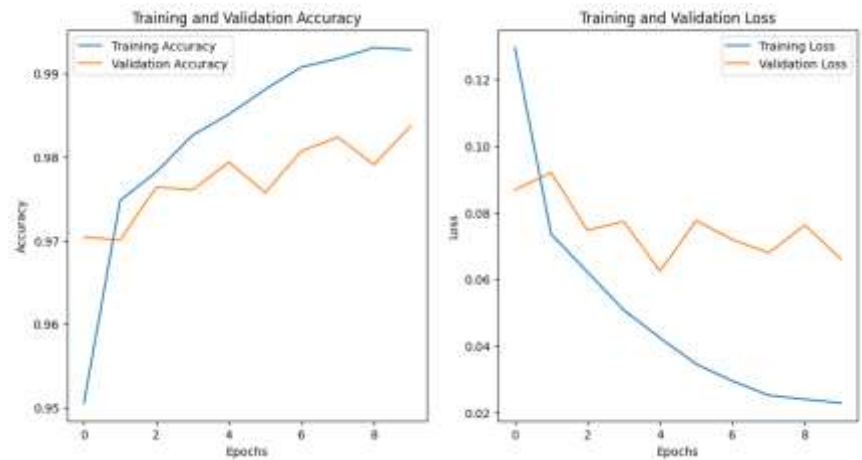


Figure 5: Graphical Explanation of Accuracy of Training and Validation and loss

ANN Model

Table of Accuracy

The performance of the ANN model used in detecting malware on Android is given by the following performance indicators as shown in the table below:

Table 3: Accuracy of ANN Model

Metrics	Accuracy	Precision	Recall	F1-Score
Accuracy	0.984	0.9786	0.9795	0.9791

In the case of the ANN, for detecting Android malware, a measure of accuracy stands at 98.4%, while the recall rate equals 0.980 making a F1-score of 0.979.

Confusion Matrix

The confusion matrix of the ANN model employed in the Android malware detection is presented below:

Table 4: Confusion Matrix of ANN Model

	Predicted Negative	Predicted Positive
Actual Negative	1861	24
Actual Positive	23	1099

The ANN model successfully detects Android malware with low false positive and false negative results High accuracy equal to 0.984, high precision of 0.979, high recall of 0.980, and a high F1-score of 0.979 makes this model appropriate to be used in cybersecurity systems.

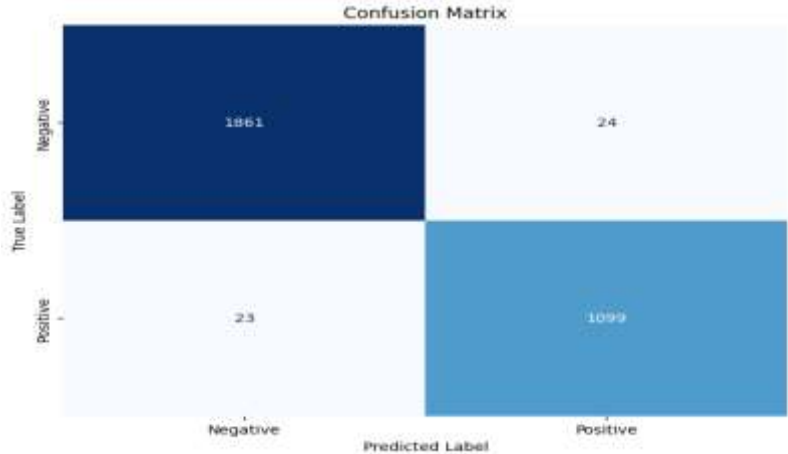


Figure 6: Confusion Matrix of ANN Model

Accuracy of Training and Validation

The learning curves of ANN reveals that the model approaches high level of accuracy and low levels of loss over time making it easy to classify the applications as either benign or malicious. This is seen through the fact that it has been used to show the convergence of the training and validation losses thus showing its stability and dependability.

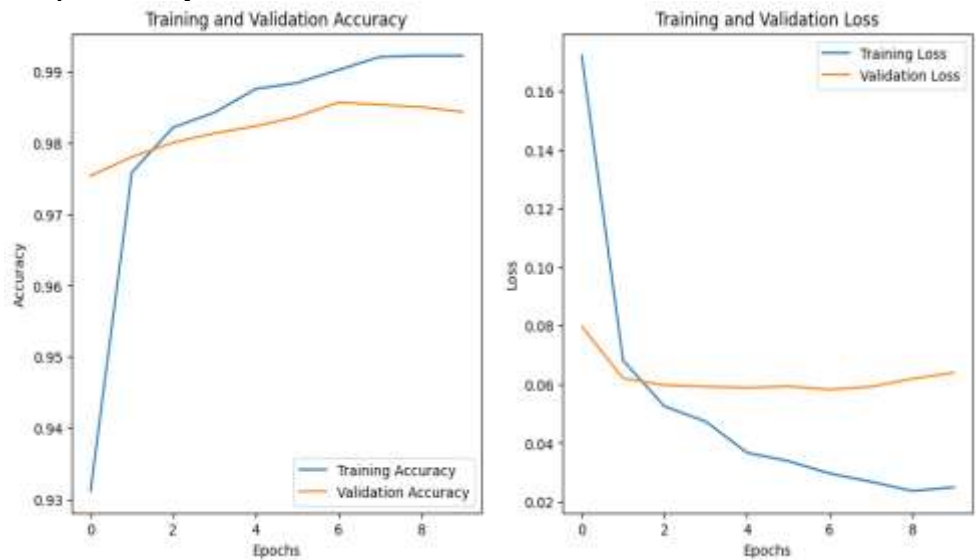


Figure 7: Graphical Explanation accuracy of Training and Validation and loss

Proposed Approach Results

The following table shows the significant performance measures concerning LSTM used in identifying malware on Android:

Table 5: Accuracy of Proposed Approach Model

Metrics	Accuracy	Precision	Recall	F1-Score
Values	0.984	0.985	0.972	0.978

The metrics used to evaluate LSTM model prove its high efficiency while dealing with Android malware detection. Specifically it’s a totally automated system and said to have an accuracy of 98.4% with very low false positive rate. It also can discover a large amount of real malware; moreover, it has a small number of missed cases. A number of points, including the stability of the model, its high precision and accuracy, contribute to the reliability of the model as a tool for enhancing the cybersecurity measures.

Confusion Matrix:

According to the mentioned performance metrics, the confusion matrix of the LSTM model may be expressed as the following:

Table 6: Confusion Matrix of Proposed Approach Model

	Predicted Negative	Predicted Positive
Actual Negative	1869	16
Actual Positive	31	1091

The developed LSTM model for Android malware detection had an accuracy of 98.4%, Precision of 0.985, Recall of 0.972 F1 score of 0.978, which makes it a perfect tool for cybersecurity.

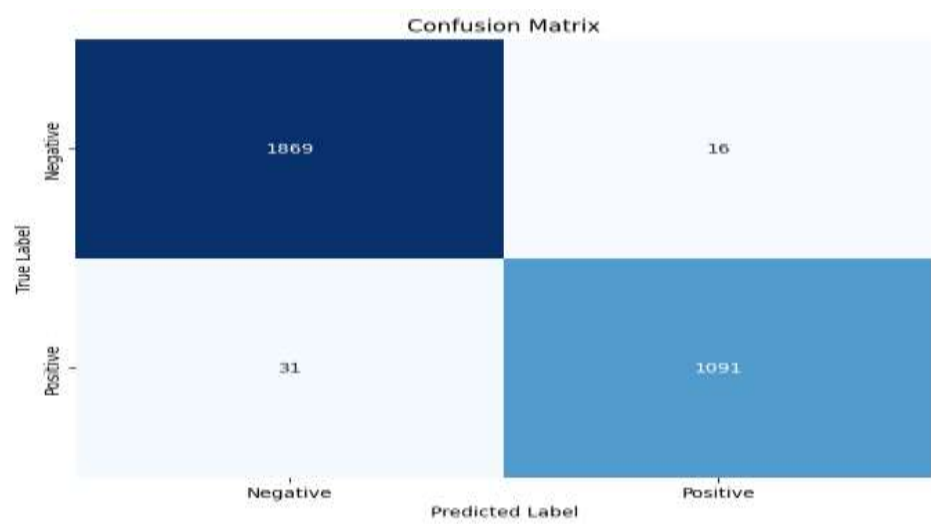


Figure 8: Confusion Matrix of Proposed Approach

Training and Validation Accuracy

When it comes to Other Performance Measurements LSTM model the learning curves revealed high training and validation accuracies which ranges near 0.984 in the Training data set showing that the given model can learn the given data set, identifying good patterns while accurately predicting on the data sets which it has not encountered.

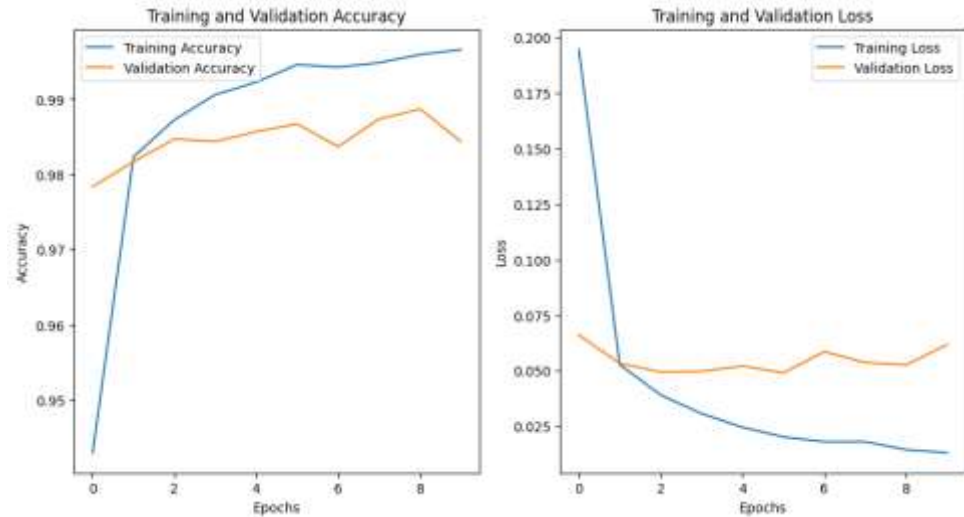


Figure 9: Visual Representation of Training accuracy and Validation loss

Conclusion and Future Work:

Android, one of the most common mobile platform, is vulnerable to attack because of its open, API interfaces. The mobile gadget is infected by viruses, worms, ransomware, spyware, Trojan horses, scareware, rootkit through system vulnerability. This research proposes a framework that utilizes LSTM to detect malware variants using a deep learning algorithm for this strategy. This research seeks to analyze and correlate long-term dependencies using LSTM to subsequent connection sequences. The research evaluates LSTM when detecting Android malware in variable sequences of benign and infected apps. It also shows how deep learning approaches namely LSTM are beneficial when identifying hidden virus patterns in distinct applications making it possible to safeguard users against malware. Future study will enhance detecting Android malware using network strategy, feature selection, statistical analysis, and machine learning strategy, focusing on memory block physics for improved results.

References:

A. R. Nasser, A. M. Hasan, and A. J. Humaidi, ‘DL-AMDet: Deep learning-based malware detector for android’, *Intelligent Systems with Applications*, vol. 21, Mar. 2024, doi: 10.1016/j.iswa.2023.200318.

T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, ‘Android malware detection based on a hybrid deep learning model’, *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/8863617.

Y. Wu, J. Shi, P. Wang, D. Zeng, and C. Sun, ‘DeepCatra: Learning flow- and graph-based behaviours for Android malware detection’, *IET Inf Secur*, vol. 17, no. 1, pp. 118–130, Jan. 2023, doi: 10.1049/ise2.12082.

S. Hosseini, A. E. Nezhad, and H. Seilani, ‘Android malware classification using convolutional neural network and LSTM’, *Journal of Computer Virology and Hacking Techniques*, vol. 17, no. 4, pp. 307–318, Dec. 2021, doi: 10.1007/s11416-021-00385-z.

E. Mbunge, B. Muchemwa, J. Batani, and N. Mbuyisa, ‘A review of deep learning models to detect malware in Android applications’, Dec. 01, 2023, *KeAi Communications Co.* doi: 10.1016/j.csa.2023.100014.

R. Lu, ‘Malware Detection with LSTM using Opcode Language’, Jun. 2019, [Online]. Available: <http://arxiv.org/abs/1906.04593>

N. Zhang, J. Xue, Y. Ma, R. Zhang, T. Liang, and Y. a Tan, ‘Hybrid sequence-based Android malware detection using natural language processing’, *International Journal of Intelligent Systems*, vol. 36, no. 10, pp. 5770–5784, Oct. 2021, doi: 10.1002/int.22529.

F. Tchakounté, R. C. N. Ngassi, V. C. Kamla, and K. P. Udagepola, ‘LimonDroid: a system coupling three signature-based schemes for profiling Android malware’, *Iran Journal of Computer Science*, vol. 4, no. 2, pp. 95–114, Jun. 2021, doi: 10.1007/s42044-020-00068-w.

I. Martín, J. A. Hernández, and S. de los Santos, ‘Machine-Learning based analysis and classification of Android malware signatures’, *Future Generation Computer Systems*, vol. 97, pp. 295–305, Aug. 2019, doi: 10.1016/j.future.2019.03.006.

H. R. Sandeep, ‘Static analysis of android malware detection using deep learning’, *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, pp. 841–845, May 2019, doi: 10.1109/ICCS45141.2019.9065765.

F. Ullah, S. Ullah, G. Srivastava, and erry C. W. Lin, ‘Droid-MCFG: Android malware detection system using manifest and control flow traces with multi-head temporal convolutional network’, *Physical Communication*, vol. 57, p. 101975, Apr. 2023, doi: 10.1016/J.PHYCOM.2022.101975.

A. Mahindru and A. L. Sangal, ‘MLDroid—framework for Android malware detection using machine learning techniques’, *Neural Comput Appl*, vol. 33, no. 10, pp. 5183–5240, May 2021, doi: 10.1007/S00521-020-05309-4/TABLES/37.

P. Bhat, S. Behal, and K. Dutta, ‘A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning’, *Comput Secur*, vol. 130, p. 103277, Jul. 2023, doi: 10.1016/J.COSE.2023.103277.

P. G. Balikcioglu, M. Sirlanci, O. A. Kucuk, B. Ulukapi, R. K. Turkmen, and C. Acarturk, 'Malicious code detection in android: the role of sequence characteristics and disassembling methods', *Int J Inf Secur*, vol. 22, no. 1, pp. 107–118, Feb. 2023, doi: 10.1007/s10207-022-00626-2.

U. Kargén, N. Mauthe, and N. Shahmehri, 'Android decompiler performance on benign and malicious apps: an empirical study', *Empir Softw Eng*, vol. 28, no. 2, Mar. 2023, doi: 10.1007/s10664-022-10281-9.

S. A. Nikale and S. Purohit, 'Comparative analysis of android application dissection and analysis tools for identifying malware attributes', *Big Data Analytics and Intelligent Systems for Cyber Threat Intelligence*, pp. 87–103, Nov. 2022, doi: 10.1201/9781003373384-6/COMPARATIVE-ANALYSIS-ANDROID-APPLICATION-DISSECTION-ANALYSIS-TOOLS-IDENTIFYING-MALWARE-ATTRIBUTES-SWAPNA-AUGUSTINE-NIKALE-SEEMA-PUROHIT.

H. Hasan, B. Tork Ladani, and B. Zamani, 'Maaker: A framework for detecting and defeating evasion techniques in Android malware', *Journal of Information Security and Applications*, vol. 78, p. 103617, Nov. 2023, doi: 10.1016/J.JISA.2023.103617.

H. Rafiq, N. Aslam, M. Aleem, B. Issac, and R. H. Randhawa, 'AndroMalPack: enhancing the ML-based malware classification by detection and removal of repacked apps for Android systems', *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-23766-w.

E. Amer and S. El-Sappagh, 'Robust deep learning early alarm prediction model based on the behavioural smell for android malware', *Comput Secur*, vol. 116, p. 102670, May 2022, doi: 10.1016/J.COSE.2022.102670.

Z. U. Rehman et al., 'Machine learning-assisted signature and heuristic-based detection of malwares in Android devices', *Computers & Electrical Engineering*, vol. 69, pp. 828–841, Jul. 2018, doi: 10.1016/J.COMPELECENG.2017.11.028.

Z. Fu, Y. Ding, and M. Godfrey, 'An LSTM-Based Malware Detection Using Transfer Learning', *Journal of Cyber Security*, vol. 3, no. 1, pp. 11–28, 2021, doi: 10.32604/jcs.2021.016632.

M. E. Z. N. Kamar, A. Esmaeilzadeh, Y. Kim, and K. Taghva, 'A Survey on Mobile Malware Detection Methods using Machine Learning', 2022 IEEE 12th Annual Computing and Communication Workshop and Conference, CCWC 2022, pp. 215–221, 2022, doi: 10.1109/CCWC54503.2022.9720753.

J. Senanayake, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, and L. Piras, 'Android Source Code Vulnerability Detection: A Systematic Literature Review', Jan. 13, 2023, Association for Computing Machinery. doi: 10.1145/3556974.

M. T. Ahvanooey, Q. Li, M. Rabbani, and A. R. Rajput, 'A Survey on Smartphones Security: Software Vulnerabilities, Malware, and Attacks', *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, Jan. 2020, doi: 10.14569/ijacsa.2017.081005.

S. Jeon and J. Moon, 'Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences', *Inf Sci (N Y)*, vol. 535, pp. 1–15, Oct. 2020, doi: 10.1016/J.INS.2020.05.026.

H. Darabian et al., 'Detecting Cryptomining Malware: a Deep Learning Approach for Static and Dynamic Analysis', *J Grid Comput*, vol. 18, no. 2, pp. 293–303, Jun. 2020, doi: 10.1007/S10723-020-09510-6/METRICS.

- A. Alzubaidi, 'Recent Advances in Android Mobile Malware Detection: A Systematic Literature Review', IEEE Access, vol. 9, pp. 146318–146349, 2021, doi: 10.1109/ACCESS.2021.3123187.
- R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, 'Robust Intelligent Malware Detection Using Deep Learning', IEEE Access, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- E. S. Alomari et al., 'Malware Detection Using Deep Learning and Correlation-Based Feature Selection', Symmetry (Basel), vol. 15, no. 1, Jan. 2023, doi: 10.3390/sym15010123.
- H. Alkahtani and T. H. H. Aldhyani, 'Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices', Sensors, vol. 22, no. 6, Mar. 2022, doi: 10.3390/s22062268.
- Y. Ban, S. Lee, D. Song, H. Cho, and J. H. Yi, 'FAM: Featuring Android Malware for Deep Learning-Based Familial Analysis', IEEE Access, vol. 10, pp. 20008–20018, 2022, doi: 10.1109/ACCESS.2022.3151357.
- S. K. Smmarwar, G. P. Gupta, and S. Kumar, 'A Hybrid Feature Selection Approach-Based Android Malware Detection Framework Using Machine Learning Techniques', Lecture Notes in Networks and Systems, vol. 370, pp. 347–356, 2022, doi: 10.1007/978-981-16-8664-1_30.
- H. Alshahrani, H. Mansourt, S. Thorn, A. Alshehri, A. Alzahrani, and H. Fu, 'DDefender: Android application threat detection using static and dynamic analysis', 2018 IEEE International Conference on Consumer Electronics, ICCE 2018, vol. 2018-January, pp. 1–6, Mar. 2018, doi: 10.1109/ICCE.2018.8326293.
- N. C. Le, T. M. Nguyen, T. Truong, N. D. Nguyen, and T. Ngo, 'A Machine Learning Approach for Real Time Android Malware Detection', Proceedings - 2020 RIVF International Conference on Computing and Communication Technologies, RIVF 2020, Oct. 2020, doi: 10.1109/RIVF48685.2020.9140771.
- R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, 'Android malware classification using optimum feature selection and ensemble machine learning', Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100–111, Jan. 2023, doi: 10.1016/J.IOTCPS.2023.03.001.