Mesopotamian journal of Cybersecurity
Vol.5,No.1, **pp**. 62–77
DOI: https://doi.org/10.58496/MJCS/2025/005; ISSN: 2958-6542
https://mesopotamian.press/journals/index.php/cybersecurity

Research Article

Enhanced Android Malware Detection through Artificial Neural Networks Technique

Mustafa Abdulfattah Habeeb [1], Yahya Layth Khaleel [1,*]

[1] *College of Computer Science and Mathematics, Tikrit University, Iraq.*

ARTICLE INFO

Article History

Received 17 Oct 2024
Accepted 10 Jan 2025
Published 03 Feb 2025

Keywords

Malicious

Threats

Cybersecurity

Smartphones

Artificial intelligence

Machine learning

ANN

ABSTRACT

Android devices are rapidly being used, which makes it easy for the malware threat to rise to higher levels. This ever-growing problem has prompted the need to enhance detection systems as far as these devices are concerned. Standard techniques of machine learning (ML) are sufficient from the point of view of their speed for searching patterns and behaviors of contemporary malware; however, it is more important to have effectively enhanced methods. The purpose of this paper is to expand the utilization of Android malware identification via artificial neural networks (ANNs) and compare its efficiency with that of other ML methods. An ANN is used in this study, and the results are compared against those of several other types of ML, including logistic regression (LR), k-nearest neighbors (KNN), extremely randomized trees (extra trees), gradient boosting (GBM), adaptive boosting (AdaBoost), and categorical boosting (CatBoost). The six evaluation values include training accuracy, testing accuracy, average accuracy, precision, recall and the F1 score. The ANN model performed well, with training and testing accuracies of 0. 99 and an average accuracy of 0. 99, precision of 0. 99, recall of 0. 98, whereas the F1 score, which is an average of both precision and recall, was 0. 99. Related studies based on conventional ML methods are also highly efficient, with some accuracy and an F1 score of 0. 95 and 0. 96. On the other hand, the ANN model ranked the best in the assessed measures. Thus, this study focuses on the reliability of ANNs for improving mobile security systems against next-generation malware and their applicability to secured Android smartphones. The reason behind the high accuracy of the ANN model is the enhanced learning ability of the ANN, which helps it learn the characteristics and dynamics of malware better than traditional ML models do.

1. INTRODUCTION

Malware is a general term that is used to refer to a vast category of malicious programs a programmer can create with the aim of damaging or compromising computers or networks [1]. These include viruses [2], worms [3], Trojan horses [4], ransomware [5], spyware [6], and adware [7]. The various types of malware all have a particular intent of several forms of dishonesty: identity theft [8], fiscal fraud [9], denial of services [10], or espionage [11]. The advancement and usage of malware attacks have increased in recent years, and as a result, an individual, business, and even a government organization does not remain untouched by it [12]. The goals of malware generation are multiple. Hackers operate toward acquiring financial benefits by stealing important information, embezzlement and ransomware [13]. Some are interested in altering the user's experience [14], threatening their security, gaining information about them or even promoting political views [2]. New connections and the general expansion of the digital economy increased new entry points for threat actors [15]. The Android operating system, the most commonly used mobile device across the globe, has now become a favorite among malware developers. The Android, which is an open development platform that is very popular in the market, is a primary target for hackers [16][35]. The android malware can be transmitted through applications, online sites, documents, downloadable applications in the mail and via short message services (SMSs) [17]. The previous methods of approach in malware detection for only Android include the following: signature detection, heuristic detection and behavioral detection. Signature-based detection uses a search for well-known patterns of the malware code [2].

While being very efficient at combating well-known malware, it is not good at dealing with newer, unknown strains of viruses. Heuristic analysis involves the processes of searching for behaviors or characteristics that may be associated with malicious software, but it is normally riddled with inaccuracies. The behavior-based detection approach observes the actions performed by applications to detect malicious action; this type of detection can be quite resource-consuming and,

*Corresponding author. Email: yahya@tu.edu.iq

as a result, might slow down the device [18][50]. Sophistication in malware continues to increase, which makes it difficult for it to be detected by normal security measures. Polymorphic and metamorphic malware can change their code to avoid being distinguished by signature detection, whereas smart malware can imitate the behaviors of legitimate apps to escape heuristic and behavior-based detection [2]. This shows an increased level of complexity as a result of which appropriate and flexible mechanisms to detect threats have also evolved more. Techniques in artificial intelligence (AI), machine learning (ML), and deep learning (DL) have become major assets in the confrontation with malicious applications [19]–[21]. AI-enabled malware detection involves applying a high level of sophistication where the AI software works through a large number of data inputs to discern a given behavior pattern and what is malicious or not. This is because, unlike criminal techniques used in the past, which can easily be detected by common antimalware software, AI can learn from new data, which is an advantage given the ever-evolving malware.

The increases in the number, variety, and complexity of various types of malwares require the application of new and better models of detection to guarantee adequate security for smartphones using the Android operating system. Many approaches, such as signature, heuristic and behavior-based approaches, have been deemed incapable of capturing the sophistication, flexibility and advancement of present-day malware. These methods fail, as they are confronted with polymorphic and metamorphic malware that changes their code or disguises themselves as other programs in an attempt to avoid detection. Hence, in the face of these heuristics, AI, ML, and DL are vital assets in combating malicious software. ML helps analyze and compare massive amounts of data and determine signs of a threat, which increases the efficiency of malware detection. The following work aims to investigate the use of improving the identification and filtering of android malware and its efficiency against commonly used ML techniques. The findings of our study show that ANNs are superior to other classifiers in the context of malware detection, which indicates that they can help enhance the frameworks of mobile security. Therefore, with the use of the currently available enhanced capabilities of ANNs, this work contributes to ongoing research on more precise and accurate techniques for detecting malware on the Android operating system.

1.1. Problem Statement

The identification of viruses in systems utilizing the Android OS is an important issue because of the prevalence of the smartphone and the advancement of malware. Signature-based and heuristic detection are some of the most commonly used methods that are not effective in detecting new evolving threats that include polymorphic and metamorphic malware. Due to these limitations, there is a need to work out other efficient methods that will help in identifying these threats and stop them. This paper seeks to fill this gap by analyzing the application of Artificial Neural Networks (ANNs) in the identification of Android malware with an ambition of increasing accuracy and reliability of the outcome than the traditional methods of machine learning process for achieving the same goal.

1.2. Motivation and Contribution

The rationale for this research is found in the increased occurrence and sophistication of Android malwares that present great threats to the privacy and integrity of users' data. Traditional approaches to detection prove quite difficult to update at the same rate as that of the malware tactics. To the best of the authors knowledge, this paper makes a significant contribution to the existing literature by harnessing ANNs, which are endowed with the ability of fast learning and flexibility, for malware detection. From this paper, the reader gets an assessment of the performance of the ANN model against other ML techniques highlighting the fact that the ANN model performs well in distinguishing the malicious applications. Furthermore, the paper includes the recommendations and guidelines on the application of ANNs in cybersecurity, noting the essential characteristic that their advantages could decrease the threat of malware which infiltrate Android systems.

2. RELATED WORK

Over the years, there have been different papers written by researchers and scholars in the area of Android security, including areas of malware analysis, detection, and vulnerability of Android applications. For example, DL techniques were used to propose a system in [22] for Android malware detection. In this way, the system achieves an accuracy of 95. 31% accuracy in identifying Malware features extracted from Manifest files and Java files of the Android platform. It employs a deep neural network (DNN) and other features, including permission combinations, intent filters, invalid certificates, Android package kit (APK) files present in the asset folder, and application programming interface (API) calls.

[23] specialize in ML algorithms, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNNs) and LSTMs (CNN-LSTM), and autoencoder (AE), for the identification of threats in Android mobile devices. The algorithms are

evaluated on two datasets and obtain very high accuracy, where the CICAndMal2017 dataset is used for SVM and the Drebin dataset is used for LSTM. A comparison with other security systems reveals that the SVM, LSTM, and CNN-LSTM algorithms can be effectively used in the detection of malware in the Android ecosystem. In [24], the authors focused on the classification of Android malware by using DNNs. Researchers have suggested an algorithm for classifying malware that is inclusive in lib. Thus, files are predicted via the CNN-LSTM network. They also measure the effectiveness of their proposed method with other familiar ML algorithms, including SVM, KNN, and random forest (RF). The accuracy level of the system is 98% in identifying the occurrence of malware.

In contrast, [25] focused on similarity-based Android malware detection via the Hamming distance of static binary features. The authors introduce four malware detection methods: the first nearest neighbors (FNN), all nearest neighbors (ANN), weighted all nearest neighbors (WANN), and k-medoid-based nearest neighbors (KMNN) algorithms. These methods aim at generating an alarm when a malevolent application is found in Android, hence eradicating the spread of malware. All the aforementioned methods were benchmarked on three datasets, namely, Drebin, Contagio and Genome, with accuracies above 90%.

Finally, [26] presented the biosentinel neural network (BSNN), which is a new DL model that is hybrid in nature and especially aimed at improving the ability to detect malware, specifically zero-day threats. The nature and components of the proposed BSNN model include the integration of a graph neural network (GNN) for feature extraction and a transformer for sequential data processing. They achieve a better accuracy of 93. 16% with a precision of 90. 89%, recall of 88. 19% and an F1 score of 90. 45% greater than traditional methods.

All these studies enrich the field of malware detection and emphasize the growing complexity of this process as well as the need for constant invention. Although older methods, such as traditional ML, have yielded promising outcomes, more flexible and effective ideas that can efficiently handle the modern threats launched by malware are needed. Thus, our work improves upon these basic works by utilizing ANNs for better detection precision and robustness, which will overcome the deficits of the current approaches and move Android malware detection to the next level. The purpose of this study is to present an extensive assessment of ANN abilities and establish a potential line of development for enhancing Android devices’ protection against malware attacks.

3. MALWARE DETECTION IN AI

Notably, malware detection has improved substantially and has been widely impacted by the use of AI and ML. It is generally observed that traditional approaches such as signature-based and heuristic detection fail to identify the new generation of intelligent malware [27], [28]. AI has enormous scalability and changeability, as it is based on algorithms that process large amounts of data and identify signs of illicit actions [29]–[32].

AI and its detection are usually conducted with the help of the training of ML models where feature sets extracted from both normal and anomalous programs are used. These features could be static properties such as the size of a file, metadata, and dynamic behavior such as API calls and network traffic. More specifically, logistic regression (LR), support vector machines (SVMs) and, last but not least, neural networks are distinguished supervised learning techniques that are frequently used and promise high accuracy in the classification of malware [27], [29], [33]–[36].

ML is categorized into subfields. DL, which uses neural networks with more than one layer, such as CNNs and recurrent neural networks (RNNs), can learn intricate features from scratch, thus improving detection. Other methods of learning without supervision include clustering and anomaly detection; new forms of malware can be detected since they are out of the usually perceived average behavior [37]–[43].

Conversely, AI-based malware detection is not without problems, some of which are evasion techniques and adversarial attacks. Persistent training makes ML models less vulnerable to new threats, whereas proper explainable AI models are very important for understanding and preventing threats [44]. The particular feature of AI as a system is its ability to learn and develop; thus, applications of AI have increased cybersecurity and fight against malicious software [45]–[51].

4. SUGGESTED METHODOLOGY

The suggested methodology used for malware detection via an ANN is shown in Figure 1, which depicts a series of steps: data collection, exploratory data analysis (EDA), preprocessing, model development, and model evaluation.

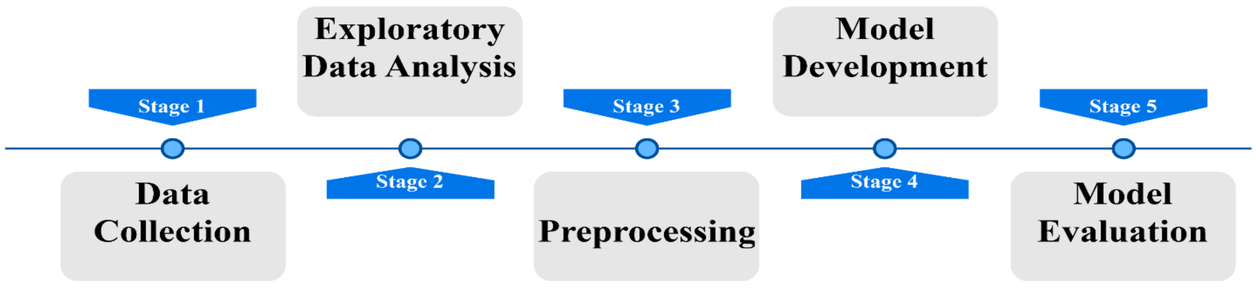

Fig. 1. Steps of the methodology.

4.1. Data collection

The data for this study were collected form [52]. This dataset is categorized and optimized for use in identifying and studying malware attacks that affect the Android operating system. It includes features extracted from Android applications with comprehensive information about their operations. The dataset consists of 4464 Android application balanced samples, whereby the samples are labeled either as malware or benign: Malware=56.74% and Benign= 43.26%, as illustrated in Figure 2. These labels are essential for supervised learning models because they help the algorithm detect patterns related to the malware as well as regular applications.

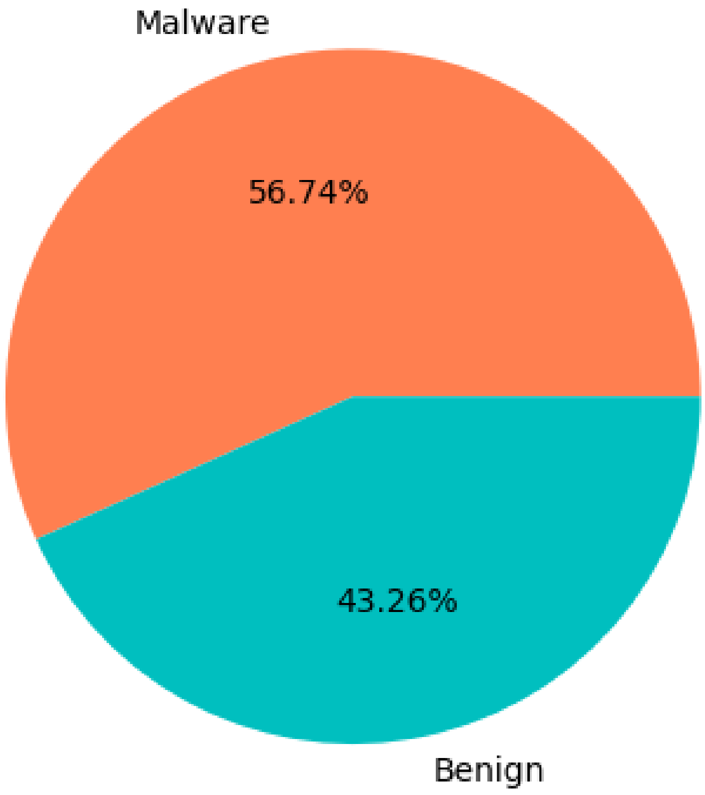

Fig. 2. Category distribution.

4.2. EDA

In an effort to better understand the structure of the given dataset and discover associations capable of improving the performance of the considered models for detecting malware, an initial analysis of the data general characteristics was performed, or an exploratory data analysis (EDA) was performed [53]. This part provides a description of the dataset and the results of the analysis that are directly correlated with it.

The obtained dataset is rather rich and contains all the features which can be extracted. Drawing from this stock, we will categorize these features as in the Figure 3:

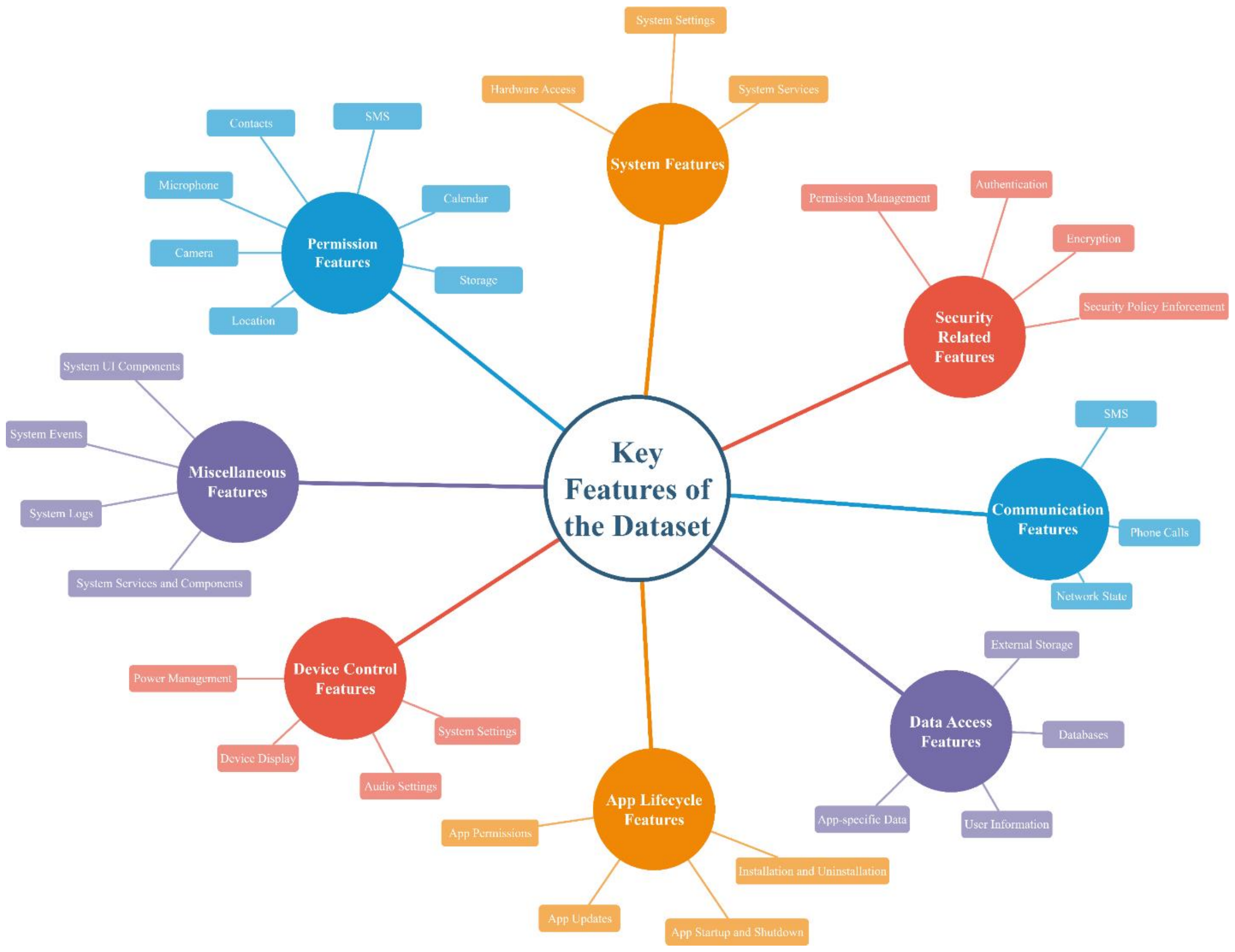

Fig. 3. Features of the dataset.

1. Permission Features: Authorization grants given by a user to an application and defines the resources and data the user wants the application to use. Examples include:
   - Location: Access to coarse and fine locations.
   - Camera: Permission to use the device's camera.
   - Microphone: Access to the microphone.
   - Contacts, SMS, Calendar, Storage: Permissions to access and modify contacts, send and receive SMS, manage calendar events, and read/write to storage.
2. System features: Aspects concerning the activities of system functions, as well as controls, which include the following:
   - Hardware Access.
   - System Settings.
   - System Services.
3. Security-related features: Items related to security functions and activities involving the following:
   - Permission Management: Granting and revoking permissions.
   - Authentication: Methods for verifying user identity.
   - Encryption: Cryptographic operations for securing data.
   - Security Policy Enforcement: Implementing security policies.
4. Communication features: The following aspects of communication are at the center of technological advances:
   - SMS: Sending and receiving messages.
   - Phone Calls: Making and receiving calls.
   - Network State: Accessing and managing network connections.

5. Data Access Features: Some of the areas related to data access and manipulation that define its capabilities include the following:
    - External Storage: Reading and writing to external storage.
    - Databases: Accessing and manipulating databases.
    - User Information: Accessing contacts and call logs.
    - App-specific Data: Managing data specific to the application
6. App Lifecycle Features: Aspects related to the application life cycle, which include the following:
    - Installation and Uninstallation: Managing app installations and removals.
    - App Startup and Shutdown: Handling app launches and closures.
    - App Updates: Managing updates.
    - App Permissions: Handling permission requests and changes.
7. Device Control Features: Manipulation features associated with a device's actions and configuration characteristics, including the following:
    - System Settings: Changing various system settings.
    - Audio settings: Modifying audio configurations.
    - Device Display: Controlling the display settings.
    - Power Management: Managing device power.
8. Miscellaneous features: Other characteristics, such as the following:
    - System Logs: Accessing system logs.
    - System Services and Components: Using services such as cameras and location managers.
    - System Events: Handling events such as incoming calls, boot completed.
    - System UI Components: Interacting with system user interface components.

These aspects and their interactions are the most fundamental in regard to the nature of Android malware and how it can be detected by ML algorithms. These findings will be useful in feature selection and model training in the sections that follow in this research.

4.3. Preprocessing

After the dataset is loaded into a data structure suitable for analysis via Python, the preprocessing in this study includes two steps:

- Splitting the dataset: To ensure that the model can generalize well to new data, the dataset is divided into two parts: the training dataset and the testing dataset. The training data constitute 70% of the total data used to train the ANN model. Hence, 30% is the testing set that is used in the assessment of the performance of the model for malware detection. This split allows the determination of the degree to which the model's performance can be generalized to other data.
- Label encoding: The labels are then converted into numerical forms with the help of a label encoder. Label encoding helps in transforming specific categorical labels into a format that can be given to ML algorithms for better and improved forecasts. In this context, the label encoder maps each label to a different integer (e.g., the label 'malware' may be encoded as 1, and the label 'benign' may be encoded as 0). This step is important because most neural networks and other popular ML algorithms accept numerical data for processing.

4.4. Model development

The DL model used for detecting Android malware in this study is an ANN [54]. The model is structured to classify inputs into one of two labels: malware or benign. The ANN is built via a sequential model from the Keras library and consists of several layers designed to process the input features and produce a binary classification output, as explained in Figure 4 below:

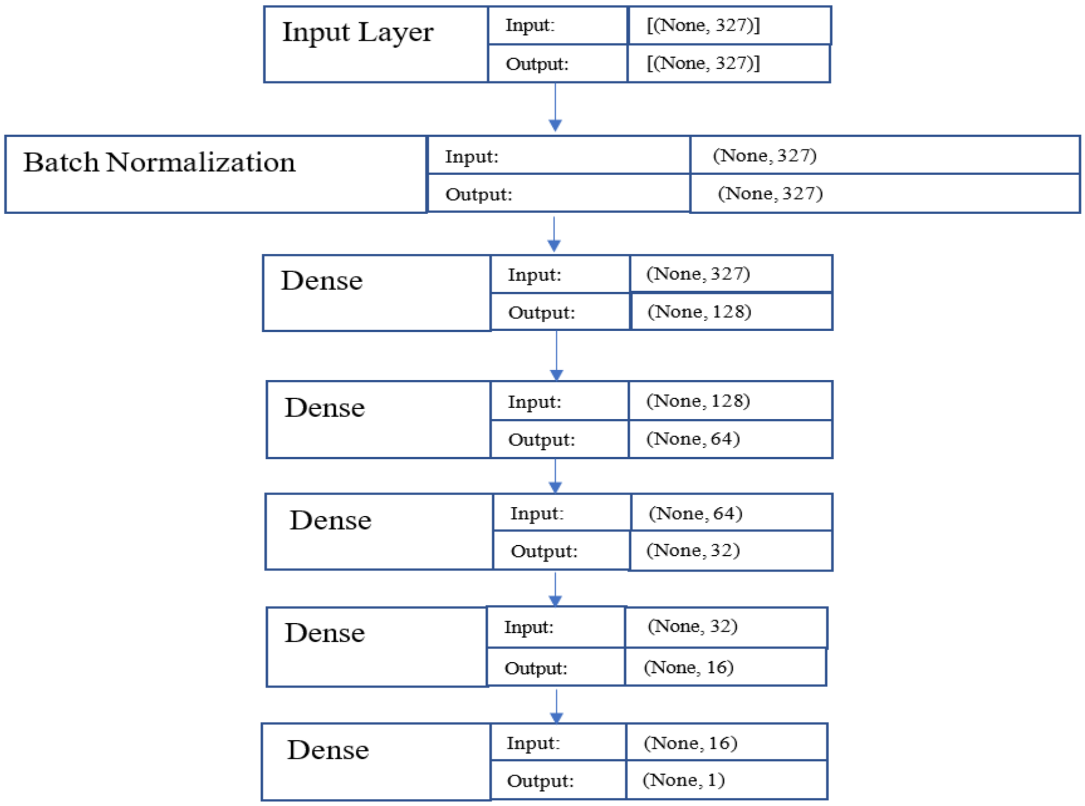

Fig. 4. ANN proposed Model.

- Input Layer: The input layer is defined to take in data that have the shape of 327 features. This means that every record in the dataset under consideration is described by a 327-feature vector.
- Batch normalization layer: This layer helps normalize the parameters that are passed on to the next layer, hence reducing the internal covariate shift, which in turn makes the network train faster and more stable.
- Dense Layer 1: This layer has 128 neurons, and the activation function used here is the ReLU or rectified linear unit. This type of layer calculates the weighted sum of the inputs, applies the addition of a bias to it and then applies the ReLU activation function to provide nonlinearity to our model to help the model learn more complex functions.
- Dense Layer 2: This layer includes 64 neurons, and the activation function that is implemented is Swis, which is a smooth, nonmonotonic activation function that can sometimes be outperformed by ReLU because it can yield small negative gradients.
- Dense Layer 3: This layer contains 32 neurons, and the activation function used is ReLU. Similar to the first dense layer, ReLU activation is applied to introduce further nonlinearity and complexity.
- Dense Layer: This layer contains 16 neurons, and the activation function used is swish. This layer also uses a swish activation function, which contributes to the network's ability to learn more complex patterns.
- Output Layer: This layer contains 1 neuron, and the activation function used is sigmoid. The sigmoid activation function outputs a value between 0 and 1, making it suitable for binary classification tasks. It effectively assigns probabilities to the two classes (malware or benign).

The model is trained for 500 epochs, meaning that the entire training dataset is passed forward and backward through the neural network 500 times. This helps the model learn and adjust the weights to minimize the loss. The training process uses a batch size of 32, meaning that the model updates its weights after processing every 32 samples from the training dataset. This helps in managing memory more efficiently and often leads to faster convergence.

4.5. Model evaluation

After training, we need to evaluate the suggested model via several metrics to ensure its effectiveness, as presented in Table 1 below.

TABLE I. METRICS FOR EVALUATING MODEL PERFORMANCE

Terms	Description	
TP	Number of samples or the population of samples which was correctly segmented as malicious	
TN	The total number of samples that were correctly classified onto the benign class	
FP	Number of incorrectly classified samples as a result of the observed distinction of malice	
FN	Erroneously classified samples in minority class	
Confusion Matrix	TP	FP
	FN	TN
Accuracy	(TP+TN)/(TP+TN+FP+FN)	
Precision	TP/(TP+FP)	
Recall	TP/(TP+FN)	
F1-Score	2*((precision*recall)/(precision +recall))	
ROC-AUC	ROC Curve: A graphical representation that shows the diagnostic performance of a binary classification system. AUC: Denotes the area beneath the ROC curve.	

This table provides definitions and descriptions of the key terms and metrics used in evaluating the performance of a binary classifier. It includes true positives (TPs), the number of samples correctly identified as malicious, and true negatives (TNs), the number of samples accurately classified as benign. False positives (FPs) are incorrectly classified as malicious, whereas false negatives (FNs) are benign samples misclassified as malicious. The confusion matrix is displayed as a 2x2 table of TP, FP, FN, and TN. Accuracy measures the overall correctness of the model and is calculated as (TP+TN)/(TP+TN+FP+FN) [55]. Precision, defined as TP/(TP+FP), indicates the proportion of true positives among the predicted positives. Recall, calculated as TP/(TP+FN), measures the model’s ability to detect positive samples. The F1 score, which combines precision and recall, is computed as 2*((precision*recall)/(precision+recall)) [56]. Finally, the receiver operating characteristic-area under the curve (ROC-AUC) provides a graphical plot (ROC curve) illustrating the diagnostic ability of the classifier, with the AUC representing the area under this curve. In addition, the fitting involves checking how well the model generalizes to new data, ensuring that it is not overfitting (too tailored to training data) or underfitting (too simplistic). Together, these metrics provide a comprehensive view of the proposed model's performance and areas for improvement.

5. RESULT AND DISCUSSION

The ANN model demonstrates exceptional performance and fitting, as evidenced by its uniformly high metrics. Both the training and test accuracies stand at 0.99, indicating that the model generalizes extremely well from training to unseen data, with no significant signs of overfitting, as presented in Figure 5. The precision rate of 0.99 suggests that the model is highly reliable in its positive predictions, which is critical in applications where false positives are costly, such as malware detection. The recall of 0.98, though slightly lower than that of the other metrics, still shows that the model successfully identifies most positive instances, which is vital for minimizing false negatives. The F1 score, 0.99, underscores the model's balanced performance in terms of precision and recall, confirming its ability to manage the trade-offs between identifying as many relevant instances as possible while maintaining accuracy.

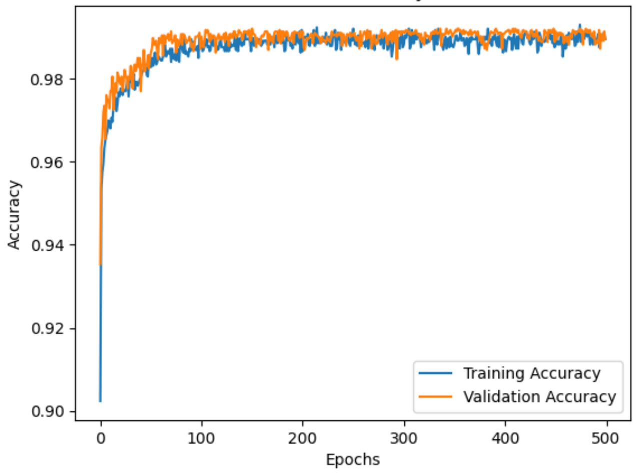

Fig. 5. Fitting of the ANN model.

Now, we compare the ANN model with several traditional ML methods and find that the ANN outperforms the other ML methods listed in Table 2 across nearly all the metrics.

TABLE II. COMPARISON BETWEEN THE RESULTS OF TRADITIONAL ML METHODS AND THE PROPOSED ANN MODEL

Method	Train accuracy	Test accuracy	Accuracy	Precision	Recall	F1-score
LR	*0.96*	*0.96*	*0.96*	*0.96*	*0.96*	*0.96*
KNN	*0.94*	*0.92*	*0.93*	*0.93*	*0.94*	*0.93*
ExtraTrees	*0.99*	*0.95*	*0.95*	*0.96*	*0.95*	*0.95*
GBM	*0.97*	*0.96*	*0.96*	*0.97*	*0.96*	*0.96*
AdaBoost	*0.96*	*0.95*	*0.96*	*0.96*	*0.96*	*0.96*
CatBoost	*0.98*	*0.96*	*0.96*	*0.96*	*0.96*	*0.96*
ANN	*0.99*	*0.99*	*0.99*	*0.99*	*0.98*	*0.99*

Table 2 provides a comparative analysis of various ML models used for Android malware detection, showing their performance in terms of training accuracy, test accuracy, overall accuracy, precision, recall, and F1 score. The models evaluated include logistic regression (LR), k-nearest neighbors (KNN), extremely randomized trees (extra trees), gradient boosting (GBM), adaptive boosting (AdaBoost), categorical boosting (CatBoost), and ANN. LR and the GBM demonstrate strong performance, with both achieving 0.96 in test accuracy and an F1 score of 0.96. KNN, albeit lower, with a test accuracy of 0.92. Even at this stage, it retains a reasonably good F1 score of 0. 93. The extra trees achieved a training accuracy of 0.99, whereas the test accuracy reached 0.95. In addition, both AdaBoost and CatBoost yield good results, with test accuracies and F1 scores of 0.95 and 0.96, respectively. However, the ANN model has a massive lead and records the highest test accuracy of 0. 99 and an F1 score of 0. 99. This improved performance signifies that ANNs are the best tool for improving Android malware detection because they are precise in identifying vicious programs. The high accuracy values demonstrated by the ANN model prove that it can be a useful instrument for developing types of innovative mobile security techniques, as presented in Figure 6.

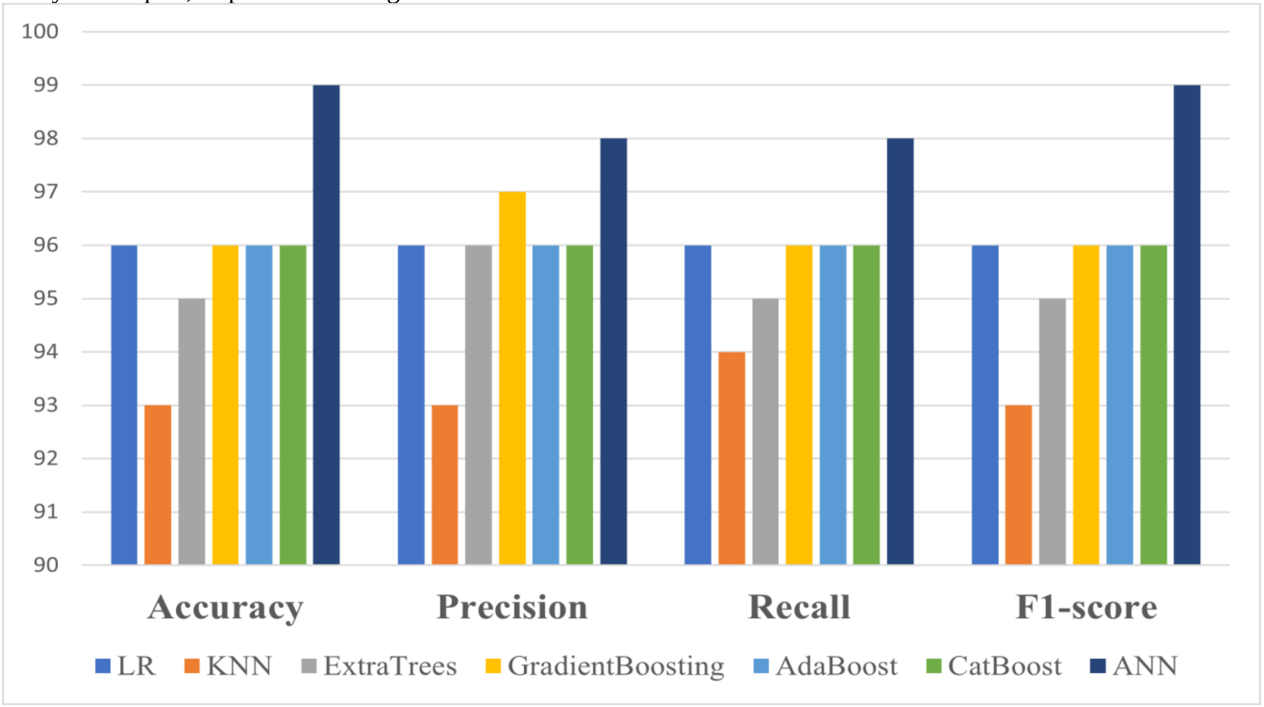

Fig. 6. Results of the ANN model

The generalization capability of the chosen ANN model shows promising performance in regard to extended use for different types of malware and the Android context. On the basis of the results, the ANN model's superior performance, marked by high training and testing accuracies (both 0.99), indicates its robust ability to capture intricate patterns and behaviors inherent to different malware types. This heightened learning capability ensures that the ANN can effectively distinguish between benign and malicious applications across diverse datasets. Figure 7 displays confusion matrices for six different ML approach ANN models.

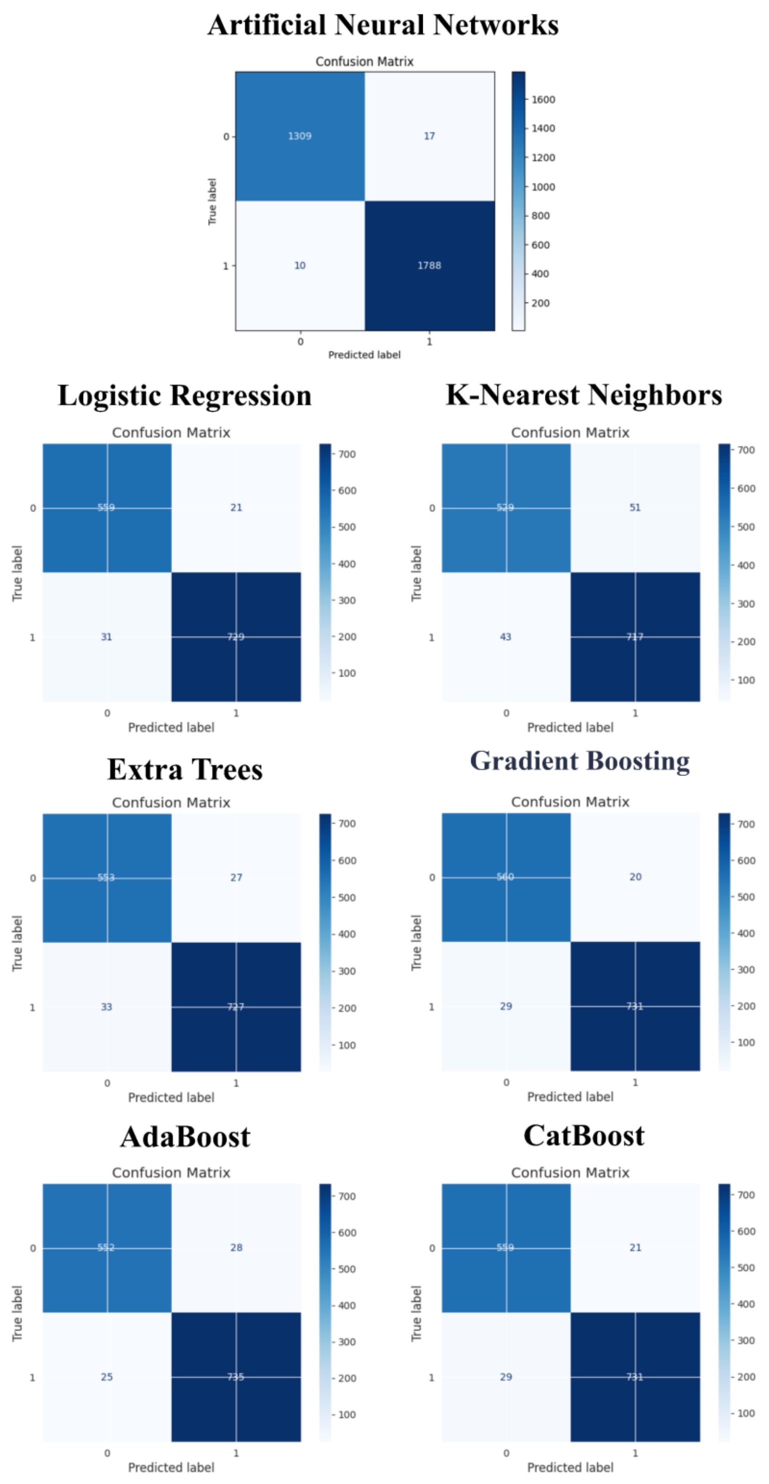

Fig. 7. Confusion matrices

The ANN model shows strong performance, with 1788 TPs, 1309 TNs, 17 FPs, and 10 FNs. This indicates high accuracy and low misclassification rates. LR also performs well but has higher misclassification rates, with 739 TPs, 559 TNs, 21 FPs, and 31 FNs. The KNN model has 717 TPs, 539 TNs, 51 FPs, and 43 FNs, indicating slightly lower performance than ANN and LR. The extra trees model performs comparably to the LR model, with 737 TPs, 553 TNs, 27 FPs, and 33 FNs. The GBM shows robust performance similar to that of the ANN, with 731 TPs, 540 TNs, 20 FPs, and 29 FNs. AdaBoost has 735 TPs, 552 TNs, 28 FPs, and 25 FNs, showing balanced performance but with slightly higher FNs than the ANN. Finally, CatBoost has 731 TPs, 519 TNs, 21 FPs, and 29 FNs, performing similarly to the GBM and showing strong predictive capabilities. While all the models demonstrate reasonable performance, the ANN appears to have the highest accuracy and the lowest misclassification rate, making it the most effective model in this comparison.

Figure 8 presents ROC curves for two approaches: traditional ML approaches and an ANN approach. The ROC curve on the left shows the performance of various ML models, with an AUC of 0.99. This high AUC indicates that the ML models have excellent discrimination capabilities between the positive and negative classes, with minimal false positive rates. On the right side of the figure, the ROC curve for the ANN approach is plotted, with an AUC of 1.00. This indicates that the ANN model is well capable of classifying the positive and the negative classes without any mistakes. The ROC curve in the case of the ANN is close to the curve with the y-axis, which means that it achieves a high true positive rate even at very low false positive rates. Overall, the performance of the ML models is rather high, with an AUC of 0. 99, whereas the other classifiers fail to do so, with an average AUC of 0. 99, while the ANN approach for classifying the given dataset has an impeccable AUC of 1.00, which demonstrates the ability of the proposed approach to classify the given data more accurately.

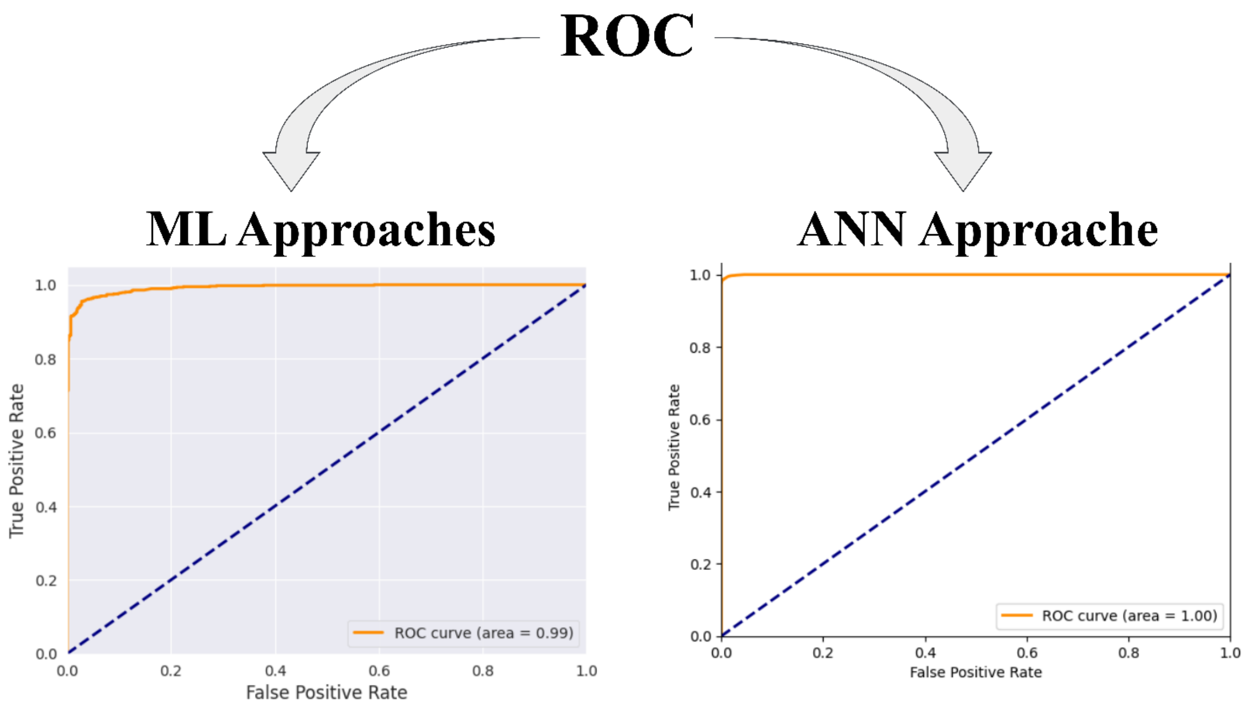

Fig. 8. ROC curves of the traditional ML techniques and the ANN technique

Overall, the ANN technique reveals higher accuracy; moreover, this result is obtained regardless of the choice of a specific subset of data, which indicates the robustness and reliability of detection. This makes it appropriate for environments that involve many risks; hence, high levels of precision are needed in regard to the results of malware detection.

6. LIMITATIONS AND FUTURE WORK

Despite the fact that this research shows how ANNs can be efficiently used for detecting Android malware, several limitations are inherent and should be considered in further investigations.

6.1. Limitations

- **Dataset Limitations:**
  Despite the fact that the data set used in our study is rather extensive, it is possible that the study does not cover all the types of malware and their actions. It has a specific drawback of the features and samples that it has been trained on might not include all the potential variants of the real-world malware, which in turn may affect its ability to deal with the new and emerging threats. This is the major shortcoming of the model, when the model is

presented with new strains of malware that it has not been exposed to before, there is a possibility of reduced detection.

- **Computational Resource Requirements:**
  The process of training and deploying deep learning models especially ANNs is a very computational intensive process. High computational power and memory are not always the best for all situations especially when the device in use is a mobile phone. This may be a disadvantage in extending the use of the proposed method, especially when the computing power is an issue in the target application area.
- **Overfitting Concerns:**
  Overfitting means if the training data is not very diverse even though it could give high accuracy. This is known as overfitting where the model learns the characteristics of the training data and noise in it and does not work well on new data. This means that when the model is applied in real life situations where the data may slightly vary from the training data, this may lead to a reduction in the model's accuracy.
- **Adversarial Attacks:**
  These include Adversarial attacks where the attackers alter the input data to deceive the model which includes ANNs and other forms of AI and ML [57], [58]. This is a well-known problem in cybersecurity since the opponents can craft inputs that will not be blocked. The study does not discuss how the proposed ANN model will avoid the adverse manipulation from the attackers.

6.2. Future Work

- **Expanding the Dataset:**
  The future work will include the expansion of the presented dataset and the variety of malware and characteristics. It could entail data collection from various sources including new types of malwares to enhance the model's ability to apply transfer learning to other types of threats. In addition, It can describe the dynamic characteristics of the behavior to give more details regarding the malware's attributes.
- **Optimizing Model Architecture:**
  For the future work, since it can be considered as one of the drawbacks of the paper that the computational power has not been fully utilized, which might attempt to further fine-tune the architecture of the ANN.
- **Enhancing Robustness Against Adversarial Attacks:**
  For the further research, it is advisable to extend the work on the methods that may improve the robustness of the ANN model against the adversarial attack. This may involve the consideration of certain strategies that could be of help in fighting the problem, such as the adversarial training that involve presenting the model to normal data as well as the contaminated data and or the development of specific measures to fight against adversarial data.
- **Other security systems and application integration:**
  Future research should be carried out on the proposed model of ANN for other security systems and to compare the results in real environment. Some of the practical uses of applying the model could involve bringing in the cybersecurity firms to explain the model's efficiency at recognizing malware on Android OS in several conditions.
- **Explainability and Interpretability:**
  One of the other areas that could be further investigated in this research is the improvement of the interpretability of the outcomes and the developed ANN model. Hence, it is crucial to understand how the model arrives at such conclusions to gain the trust of the users and improve the model's performance. For the interpretation of the model's choice, it can employ SHAP, which stands for SHapley Additive exPlanations or LIME, which is the acronym for Local Interpretable Model-agnostic Explanations.

Therefore, extending these future directions and overcoming the mentioned limitations, the contribution of the research can be in the enhancement of the Android malware detection techniques and provision of better and more practical tools for this purpose that finally leads to the enhancement of the mobile devices security.

7. CONCLUSION

This paper aims to provide an in-depth investigation of how the use of ANNs can improve the detection of Android malware in comparison with other known ML methods. The types of data analysis performed in the study included the use of an ANN model and comparisons with other forms of ML methods. By observing the results, it is found that the ANN model outperforms the considered models in all the observed aspects. The general enhancement in the learning aptitude of the ANN model makes it successful in this area. The use of a neural network structure for the model results in the identification of the patterns typical of modern malware being more effective than conventional ML. This learning ability and adaptation capability make the detection and differentiation of the new generation of Android malware simpler with the help of the ANN-based approach.

The overall significance of the study’s findings for mobile security science is large. Therefore, the use of ANN-based detection systems can be employed to present a dependable and defendable strategy to counter the rising Android smartphone threat in the form of malware. With the help of the ANN, which is the basis for the suggested approach, even the most complex and concealed malicious programs are easy to detect and eliminate. Furthermore, on the basis of the better performance of the ANN model illustrated in this paper, the ANN model can be viewed as a useful tool for security researchers and practitioners in designing future generation Android malware detection systems. The possibility of achieving a high degree of accuracy and speed in applying classification reveals great potential for enhancing the security of the Android environment, thus safeguarding users, companies, and even governmental structures from the harm caused by malware. Adaptability is a major advantage that can be tagged to the ANN-based approach, which forms the core of the review. However, concerning the type of Android malware that will be developed in the future, the model comprising a neural network will be capable of being updated and trained again. Thus, flexibility is essential for the security system, and this aspect makes it possible to improve it to prevent newly appeared malware threats, which makes it a valuable and effective solution for the protection of Android devices in the future.

As stated above, this study has some shortcomings, despite yielding positive results. First, although the dataset used is large and diverse, it is still a sample of overall malware and could contain somewhat diverse kinds and actions of malware, so the results may not be universal. Second, it might determine that this model can be influenced by the movement of malware where the pattern is frequently changed so that it is not easily detected. Moreover, the work is based mainly on the static and dynamic attributes of the applications, which may not include all the potential indicators of malicious activities. Finally, the computational power that is needed to train these high-level neural networks can be very intensive and hence may not be feasible for implementation in real-world scenarios that may be characterized by limited computational power.

Future research should overcome these limitations by using more diverse datasets containing increased numbers of various types of malware and their behaviors. Improving the model to incorporate new and variant malware threats will be relevant. Researchers should also consider enhancing the detection of the proposed features with information in the context with which it is being used as well as other patterns of user behavior. Furthermore, advancements in training algorithms and feature architectures require effort to reduce the training complexities related to the ANN automated malware detection system. Finally, research in making the models understandable would be crucial to enhance the use of these models, given that the users need to comprehend the manner in which the models arrive at certain decisions.

Conflicts Of Interest

The author's paper clearly states that no conflicts of interest exist in relation to the research or its publication.

Funding

The absence of any funding statements or disclosures in the paper suggests that the authors had no institutional or sponsor backing.

Acknowledgment

The authors is grateful to the institution for their collaboration and provision of necessary facilities that contributed to the successful completion of this research.y

References

[1] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi, “A Survey on malware analysis and mitigation techniques,” *Comput. Sci. Rev.*, vol. 32, pp. 1–23, May 2019, doi: 10.1016/j.cosrev.2019.01.002.

[2] O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[3] J. Singh and J. Singh, “A survey on machine learning-based malware detection in executable files,” *J. Syst. Archit.*, vol. 112, p. 101861, 2021, doi: 10.1016/j.sysarc.2020.101861.

[4] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, “A Survey on Machine Learning against Hardware Trojan Attacks: Recent Advances and Challenges,” *IEEE Access*, vol. 8, pp. 10796–10826, 2020, doi: 10.1109/ACCESS.2020.2965016.

[5] A. Alqahtani and F. T. Sheldon, “A Survey of Crypto Ransomware Attack Detection Methodologies: An Evolving Outlook,” *Sensors*, vol. 22, no. 5, p. 1837, 2022, doi: 10.3390/s22051837.

[6] K. M. E. NarasimaMallikarajunan, S. R. Preethi, S. Selvalakshmi, and N. Nithish, "Detection of spyware in software using virtual environment," in *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, IEEE, 2019, pp. 1138–1142. doi: 10.1109/icoei.2019.8862547.

[7] E. Arul and A. Punidha, "Adware Attack Detection on IoT Devices Using Deep Logistic Regression SVM (DL-SVM-IoT)," in *Congress on Intelligent Systems: Proceedings of CIS 2020, Volume 1*, Springer, 2021, pp. 167–176. doi: 10.1007/978-981-33-6981-8_14.

[8] I. Guedes, M. Martins, and C. S. Cardoso, "Exploring the determinants of victimization and fear of online identity theft: an empirical study," *Secur. J.*, vol. 36, no. 3, pp. 472–497, 2023, doi: 10.1057/s41284-022-00350-5.

[9] N. K. Gyamfi and J.-D. Abdulai, "Bank Fraud Detection Using Support Vector Machine," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 37–41. doi: 10.1109/IEMCON.2018.8614994.

[10] X. M. Zhang, Q. L. Han, X. Ge, and L. Ding, "Resilient Control Design Based on a Sampled-Data Model for a Class of Networked Control Systems under Denial-of-Service Attacks," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3616–3626, 2020, doi: 10.1109/TCYB.2019.2956137.

[11] R. Rivera, L. Pazmiño, F. Becerra, and J. Barriga, "An Analysis of Cyber Espionage Process," in *Smart Innovation, Systems and Technologies*, Springer, 2022, pp. 3–14. doi: 10.1007/978-981-16-4884-7_1.

[12] J. DiMaggio, "The art of cyberwarfare : an investigator's guide to espionage, ransomware, and organized cybercrime," p. 254.

[13] M. Wade, "Digital hostages: Leveraging ransomware attacks in cyberspace," *Bus. Horiz.*, vol. 64, no. 6, pp. 787–797, 2021, doi: 10.1016/j.bushor.2021.07.014.

[14] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: Recent advances, analysis, challenges and future research directions," *Comput. Secur.*, vol. 111, p. 102490, 2021, doi: https://doi.org/10.1016/j.cose.2021.102490.

[15] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018, doi: 10.1109/COMST.2018.2855563.

[16] K. Bakour, H. M. Ünver, and R. Ghanem, "The Android malware detection systems between hope and reality," *SN Appl. Sci.*, vol. 1, no. 9, p. 1120, Sep. 2019, doi: 10.1007/s42452-019-1124-x.

[17] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy &amp; future directions," *Futur. Gener. Comput. Syst.*, vol. 97, pp. 887–909, Aug. 2019, doi: 10.1016/j.future.2019.03.007.

[18] A. V. Pandit and D. Mondal, "Real-Time Malware Detection on IoT Devices using Behavior-Based Analysis and Neural Networks," *Res. J. Comput. Syst. Eng.*, vol. 4, no. 2, pp. 117–129, Dec. 2023, doi: 10.52710/rjcse.82.

[19] A. S. Albahri *et al.*, "A systematic review of trustworthy artificial intelligence applications in natural disasters," *Comput. Electr. Eng.*, vol. 118, p. 109409, 2024, doi: 10.1016/j.compeleceng.2024.109409.

[20] M. A. Habeeb, Y. L. Khaleel, and A. S. Albahri, "Toward Smart Bicycle Safety: Leveraging Machine Learning Models and Optimal Lighting Solutions," in *Proceedings of the Third International Conference on Innovations in Computing Research (ICR'24)*, K. Daimi and A. Al Sadoon, Eds., Cham: Springer Nature Switzerland, 2024, pp. 120–131.

[21] L. A. E. Al-saeedi *et al.*, "Artificial Intelligence and Cybersecurity in Face Sale Contracts: Legal Issues and Frameworks ," *Mesopotamian J. CyberSecurity*, vol. 4, no. 2 SE-Articles, pp. 129–142, Aug. 2024, doi: 10.58496/MJCS/2024/0012.

[22] A. Naway and Y. LI, "Using Deep Neural Network for Android Malware Detection." 2019.

[23] H. Alkahtani and T. H. H. Aldhyani, "Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices," *Sensors*, vol. 22, no. 6, p. 2268, Mar. 2022, doi: 10.3390/s22062268.

[24] S. Hosseini, A. E. Nezhad, and H. Seilani, "Android malware classification using convolutional neural network and LSTM," *J. Comput. Virol. Hacking Tech.*, vol. 17, no. 4, pp. 307–318, Dec. 2021, doi: 10.1007/s11416-021-00385-z.

[25] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian, and M. Conti, "Similarity-based Android malware detection using Hamming distance of static binary features," *Futur. Gener. Comput. Syst.*, vol. 105, pp. 230–247, 2020, doi: 10.1016/j.future.2019.11.034.

[26] D. S. Rani, K. Gnaneshwar, K. Sampurnima Pattem, S. Sekhar, G. B. Krishna, and S. Kakarla, "Advancing Android Malware Detection with BioSentinel Neural Network using Hybrid Deep Learning Techniques," in *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, Feb. 2024, pp. 1754–1760. doi: 10.23919/INDIACom61295.2024.10498302.

[27] A. A. Almazroi and N. Ayub, "Deep learning hybridization for improved malware detection in smart Internet of Things," *Sci. Rep.*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-57864-8.

[28] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 1, 2024, doi: 10.1186/s42400-024-00205-z.

[29] A. Mahindru *et al.*, "PermDroid a framework developed using proposed feature selection approach and machine learning techniques for Android malware detection," *Sci. Rep.*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-60982-y.

[30] S. I. Mihali and Ștefania L. Niță, "Credit Card Fraud Detection based on Random Forest Model," in *2024 17th International Conference on Development and Application Systems, DAS 2024 - Proceedings*, 2024, pp. 111–114. doi: 10.1109/DAS61944.2024.10541240.

[31] B. K. Sedraoui, A. Benmachiche, A. Makhlouf, and C. Chemam, "Intrusion Detection with deep learning: A literature review," in *PAIS 2024 - Proceedings: 6th International Conference on Pattern Analysis and Intelligent Systems*, 2024, pp. 1–8. doi: 10.1109/PAIS62114.2024.10541191.

[32] A. Sheneamer, "Visualized Malware Images using Hybrid Ensemble Deep Transfer Learning," in *Proceedings - 2024 7th International Conference on Information and Computer Technologies, ICICT 2024*, 2024, pp. 7–12. doi: 10.1109/ICICT62343.2024.00008.

[33] W. Z. A. Zakaria, N. M. K. M. Alta, M. F. Abdollah, O. Abdollah, and S. M. W. M. S. M. M. Yassin, "Early Detection of Windows Cryptographic Ransomware Based on PreAttack API Calls Features and Machine Learning," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 39, no. 2, pp. 110–131, 2024, doi: 10.37934/araset.39.2.110131.

[34] R. A. Yunmar, S. S. Kusumawardani, W. Widyawan, and F. Mohsen, "Detecting Android Malware by Mining Enhanced System Call Graphs," *Int. J. Comput. Netw. Inf. Secur.*, vol. 16, no. 2, pp. 28–41, 2024, doi: 10.5815/ijcnis.2024.02.03.

[35] S. salman Qasim and S. M. NSAIF , Trans., "Advancements in Time Series-Based Detection Systems for Distributed Denial-of-Service (DDoS) Attacks: A Comprehensive Review", BJN, vol. 2024, pp. 9–17, Jan. 2024, doi: 10.58496/BJN/2024/002.

[36] C. C. Moreira, D. C. Moreira, and C. Sales, "A comprehensive analysis combining structural features for detection of new ransomware families," *J. Inf. Secur. Appl.*, vol. 81, 2024, doi: 10.1016/j.jisa.2024.103716.

[37] S. A. Hamad, Q. Z. Sheng, and W. E. Zhang, *Security Framework for The Internet of Things Applications*. CRC Press, 2024. doi: 10.1201/9781003478683.

[38] H. Zhao, C. Zi, Y. Liu, C. Zhang, Y. Zhou, and J. Li, "Weakly Supervised Anomaly Detection via Knowledge-Data Alignment," in *WWW 2024 - Proceedings of the ACM Web Conference*, Association for Computing Machinery, Inc, 2024, pp. 4083–4094. doi: 10.1145/3589334.3645429.

[39] K. Shaukat, S. Luo, and V. Varadharajan, "A novel machine learning approach for detecting first-time-appeared malware," *Eng. Appl. Artif. Intell.*, vol. 131, p. 107801, 2024, doi: 10.1016/j.engappai.2023.107801.

[40] R. Liao and S. Wang, "Malicious domain detection based on semi-supervised learning and parameter optimization," *IET Commun.*, vol. 18, no. 6, pp. 386–397, 2024, doi: 10.1049/cmu2.12739.

[41] M. Fleming and O. Olukoya, "A temporal analysis and evaluation of fuzzy hashing algorithms for Android malware analysis," *Forensic Sci. Int. Digit. Investig.*, vol. 49, 2024, doi: 10.1016/j.fsidi.2024.301770.

[42] A. Mondal, A. Ghosh, S. Karmakar, M. H. Mahalat, S. Roy, and B. Sen, "Identification of Hardware Trojan in Gate-Level Netlist," *J. Circuits, Syst. Comput.*, vol. 33, no. 9, 2024, doi: 10.1142/S0218126624300058.

[43] A. Saihood, M. A. Al-Shaher, and M. A. Fadhel, "A New Tiger Beetle Algorithm for Cybersecurity, Medical Image Segmentation and Other Global Problems Optimization," *Mesopotamian J. CyberSecurity*, vol. 2024, pp. 17–46, 2024, doi: 10.58496/MJCS/2024/003.

[44] A. S. Albahri, Y. L. Khaleel, and M. A. Habeeb, "The Considerations of Trustworthy AI Components in Generative AI; A Letter to Editor," *Appl. Data Sci. Anal.*, vol. 2023, pp. 108–109, 2023, doi: 10.58496/adsa/2023/009.

[45] A. Galli, V. La Gatta, V. Moscato, M. Postiglione, and G. Sperli, "Explainability in AI-based behavioral malware detection systems," *Comput. Secur.*, vol. 141, 2024, doi: 10.1016/j.cose.2024.103842.

[46] S. Gulmez, A. Gorgulu Kakisim, and I. Sogukpinar, "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," *Comput. Secur.*, vol. 139, 2024, doi: 10.1016/j.cose.2024.103703.

[47] J. Mitchell, N. McLaughlin, and J. Martinez-del-Rincon, "Generating sparse explanations for malicious Android opcode sequences using hierarchical LIME," *Comput. Secur.*, vol. 137, 2024, doi: 10.1016/j.cose.2023.103637.

[48] S. R. Sindiramutty *et al.*, *Explainable AI for Cybersecurity*. 2024. doi: 10.4018/978-1-6684-6361-1.ch002.

[49] O. Arreche, T. R. Guntur, J. W. Roberts, and M. Abdallah, "E-XAI: Evaluating Black-Box Explainable AI Frameworks for Network Intrusion Detection," *IEEE Access*, vol. 12, pp. 23954–23988, 2024, doi:

10.1109/ACCESS.2024.3365140.

[50] D. Zaman and M. Mazinani, "Cybersecurity in Smart Grids: Protecting Critical Infrastructure from Cyber Attacks", SHIFRA, vol. 2023, pp. 86–94, Aug. 2023, doi: 10.70470/SHIFRA/2023/010.

[51] M. AL-Essa, G. Andresini, A. Appice, and D. Malerba, "PANACEA: a neural model ensemble for cyber-threat detection," *Mach. Learn.*, 2024, doi: 10.1007/s10994-023-06470-2.

[52] PCSL, "Android Malware Detection Test," 2014. https://www.kaggle.com/datasets/dannyrevaldo/android-malware-detection-dataset

[53] E. Camizuli and E. J. Carranza, "Exploratory Data Analysis," in *The Encyclopedia of Archaeological Sciences*, Wiley, 2018, pp. 1–7. doi: 10.1002/9781119188230.saseas0271.

[54] M. G. M. Abdolrasol *et al.*, "Artificial Neural Networks Based Optimization Techniques: A Review," *Electronics*, vol. 10, no. 21, 2021, doi: 10.3390/electronics10212689.

[55] F. K. H. Mihna, M. A. Habeeb, Y. L. Khaleel, Y. H. Ali, and L. A. E. Al-Saeedi, "Using Information Technology for Comprehensive Analysis and Prediction in Forensic Evidence," *Mesopotamian J. CyberSecurity*, vol. 4, no. 1, pp. 4–16, 2024, doi: 10.58496/MJCS/2024/002.

[56] H. M. Abdulfattah, K. Y. Layth, and A. A. Raheem, "Enhancing Security and Performance in Vehicular Adhoc Networks: A Machine Learning Approach to Combat Adversarial Attacks," *Mesopotamian J. Comput. Sci.*, vol. 2024, pp. 122–133, 2024, doi: 10.58496/MJCSC/2024/010.

[57] Y. L. Khaleel, M. A. Habeeb, A. S. Albahri, T. Al-Quraishi, O. S. Albahri, and A. H. Alamoodi, "Network and cybersecurity applications of defense in adversarial attacks: A state-of-the-art using machine learning and deep learning methods," *J. Intell. Syst.*, vol. 33, no. 1, 2024, doi: 10.1515/jisys-2024-0153.

[58] Y. L. Khaleel, H. M. Abdulfattah, and H. Alnabulsi, "Adversarial Attacks in Machine Learning: Key Insights and Defense Approaches," *Appl. Data Sci. Anal.*, vol. 2024, pp. 121–147, 2024, doi: 10.58496/ADSA/2024/011.