Static and Dynamic Data Analysis for Android Malware Detection using Red Fox Optimization

Ambika Dulal, Merry Singh, Dr. Gajendra Sharma,

[1]MCA, Kantipur City College, Dept. of Science and Technology, Kathmandu 44600, Nepal
[2]Assistant Professor, Kantipur City College, Dept. of Science and Technology, Kathmandu 44600, Nepal
[3]Associate Professor, Kathmandu University, Dept. of Computer Science and Engineering School of Engineering, Dhulikhel 45200, Kavre, Nepal

Abstract

The rapid evolution of smartphone technology has led to a surge in cyber-attacks specifically targeting smart devices of particular concern is the prevalent practice among users to grant permissions to arbitrary programs without due consideration, thereby undermining the efficacy of the authorization system. While numerous malware detection methods have been proposed, they often exhibit limitations, including inadequate identification and detection rates. *To* address these pressing challenges, this paper presents a novel approach called Convolutional Neural Network-Based Adaptive Red Fox Optimization (CNN-ARFO) aimed at discerning between normal and malicious malware applications on Android smartphones. The methodology involves the pre-processing of the dataset, wherein the Minmax approach is employed to effectively normalize features. Subsequently, during extraction, the CNN-ARFO method meticulously scrutinizes malicious APKs, curating a representative selection of normal apps to extract essential characteristics crucial for identifying malware operations. The foundation of the proposed method lies in harnessing the Adaptive Red Fox Optimization (ARFO) technique in conjunction with Convolutional Neural Networks (CNNs) to achieve precise and reliable malware detection. The CNN-ARFO approach is comprehensively evaluated to assess its efficacy, and its performance is compared against other state-of-the-art malware detection techniques, including DANN, DBN, MLBS, and DE-CNN. A diverse set of evaluation metrics, including accuracy, precision, recall, and f-measure, are employed for a thorough analysis. Experimental results indicate the superiority of the CNN-ARFO method, demonstrating outstanding performance with accuracy, precision, recall, and f-measure values of 99.20%, 97.45%, 91.23%, and 98.43%, respectively.

Keywords: Malware; Detection techniques; Malicious; Normal; Red Fox Optimization;

1. INTRODUCTION

1.1. Background

Since 2008, the Android operating system has been a strong mobile operating system in the global context. It has become the most popular operating system for smart mobile phones. In 2021, about 1,433.86 million units of smartphones sold globally were based on Android [1]. According to strategy analytics, 87.5 % of smartphones market with the Android operating system on the Global market. Since 2021, the use of smartmobile phones has increased due to their simplicity and effectiveness for the normal user. The rate of smartphones is low but the demand for smartphones is high compared to IOS devices. While the increasing demand for smartphones is good for smartphone suppliers, some hidden disadvantages are found for the non - technical user. Malware applications are continuously increasing as a threat and affecting real applications through various ways like bank lies, malicious APK files, phonic calls, lottery, and ticketing scams so many ways are found these days but now we are focusing the Android malicious threat on the Android operating system. As per the latest Google Play status, we will be shocked to know that there are 3.48 million apps currently at the Google Play Store. This number of apps on Google Play is on the rise as 3,739 apps are added to the Play Store every single day [2]. Many security attack surfaces exist due to many variables such as

Android applications' open ecological mode, coarse-grained permission management, and the ability to invoke third-party code, which gravely compromises the integrity of Android applications. The machine learning technique itself deduces application-based activities when integrated with program analyzing techniques. This approach is also based on two main techniques: static analysis and dynamic technique. The static approach is a most beneficial approach that quickly scans and determines the malicious application but many malware apps established a sequence of deformation technical advancements namely execution of native code, and encryption byte code with reflection [3]. According to a similar article by previous authors, the proposed approach for the effective detection of Android malware applications consists of three independent phases: pre-processing, feature extraction, and detection. In the pre-processing phase of the selected data set, the Minmax approach is utilized to normalize the features. During the extraction phase, the malicious APK and the collected regular apps are analyzed to detect and extract the features required for malware to function properly. Finally, to recognize Android mobile applications, the ARFO technique based on CNN is applied [4].

1.2. Related Works

(I) A Review of Android Malware Detection Approaches Based on Machine Learning

The backdrop to Android malware was briefly provided, followed by a full analysis of machine learning-based methodologies for detecting Android malware, roughly in the order of the machine learning development pipeline. At each level, a variety of different approaches were evaluated, with extensive assessment of their advantages in specific scenarios. Some significant knowledge was summarized in the form of diagrams and tables to aid comprehension and comparative analysis. Finally, we identified five areas for further research: sample set formation, data optimization and processing, feature extraction and establishment, machine learning application, and classifier evaluation. This paper focuses on machine learning methods for detecting malware on Android. We believe that this work adds to previous assessments by filling some research gaps and highlighting some open issues in this sector. We hope that this evaluation will serve as a starting point for interested readers and inspire them to pursue new research directions [5].

(II) A Systematic Literature Review of Android Malware Detection Using Static Analysis

According to this SLR, (1) the most commonly used static analysis technique in Android malware detection is Android characteristic; (2) in empirical studies, in-lab data sets such as Drebin and Genome occupy the biggest proportion. Most research use Apk tool as a static analysis assistance tool. The most commonly utilized feature reduction approach is IG. The most commonly utilized features are permissions and sensitive API requests. The machine learning model accounts for the majority of the models employed. And accuracy is the most commonly used performance metric; (3) empirical evidence shows that static analysis techniques are successful at detecting malware; (4) Based on the analysis and comparison of main investigations, they arrive at the preliminary conclusion that the neural network model outperforms the non-neural network model. This SLR concludes that Android malware detection via static analysis still faces several obstacles based on the outcomes of research questions. To address this issue, we suggest some criteria for developing novel strategies that increase the effectiveness of Android malware detection and establishing an uniform platform for properly assessing the performance of multiple techniques. We will follow these principles in the future to boost Android malware detection methods [6].

(III) Convolution Neural Network for Classification of Android Applications Represented as Grayscale Images

The investigations are carried out on a data set of 1747 grayscale photos of Android applications from 13 malware families. We claim that our data sets are made up of latest Android harmful apps. The results show that photos made with classess.dex files have higher categorization accuracy than images created with AndroidManifest.xml files. It is also determined that when image size increases for both types of files, classification accuracy increases. Thus, images formed from the classess.dex file with a resolution of 256x256 had the maximum classification accuracy (74.5%). This article employed assessment parameters to evaluate and compare the suggested CNN model design on various picture sizes created from AndroidManifest.xml and classess.dex files. It also includes an analysis of the results acquired [8].

(IV) A Hybrid Approach for Android Malware Detection and Family Classification

They worked on both detection and family classification of Android malware in this paper. Here, detection refers to a binary classification problem with two classes: "malware" and "normal," whereas family classification refers to a multi class classification problem with 13 malicious families. The term "Android malware family" refers to a group of malicious programs that exhibit similar behavior and are derived from the same source code. They propose a hybrid approach for detecting and classifying malicious Android apps. It is based on the combination of static and dynamic malware analysis. They begin with static malware analysis in order to extract static features based on API calls, command strings, permissions, and intents. Then, using CuckooDroid, we performed dynamic malware analysis to extract features.

(V) An Efficient Android Malware Detection Using Adaptive Red Fox Optimization Based CNN

This paper proposes a convolutional neural network-based adaptive red fox optimization (CNN-ARFO) method for classifying malware applications as normal or malicious. For the effective detection of Android malware applications, the proposed approach consists of three distinct phases: pre-processing, feature extraction, and detection. The Minmax technique is used to normalize the features in the pre-processing phase of the selected data set. In the extraction phase, the malicious APK and the collected normal apps are investigated to identify and extract the essential features for the proper operation of malware. Finally, the ARFO approach based on CNN is used to detect Android mobile applications. The results of detecting normal and malicious applications on Android mobiles are then demonstrated by evaluating parameters such as model accuracy rate, model loss rate, accuracy, precision, recall, and f-measure. The outcome revealed that the proposed approach achieves a detection accuracy of 97.29% [7].

2. METHODOLOGY

The proposed framework’s workflow includes three distinct phases: pre-processing, feature extraction, and detection. This section presents a proposed CNN-ARFO approach for detecting malicious Android applications. The proposed framework’s workflow includes three distinct phases: pre-processing, feature extraction, and detection. This section presents a proposed CNN-ARFO approach for detecting malicious Android applications.

Table 1. Confusion matrix

Class	Malicious	Normal
Predicted	1	2
Actual	3	4

Through the pre-processing, feature extraction, and detection phases, the researchers created a framework for Android malware detection using the ARFO technique. Data collection, preprocessing, cleaning, splitting, model training, evaluation, and comparison with previous papers were all part of the process. VirusTotal.com data was collected over three years. The data was cleaned and split into a training and test set in 80:20 ratios.

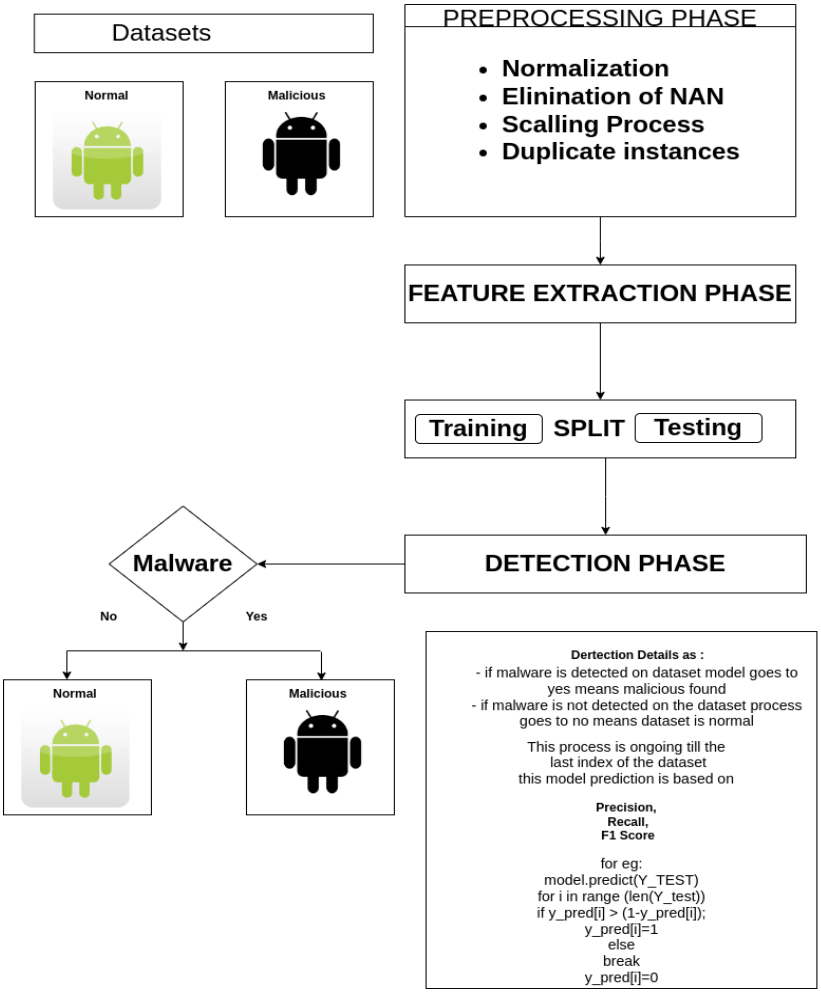

Fig. 1. Workflow of the Proposed Framework

A. Data Pre-processing

In general, the pre-processing phase includes normalization, NaN elimination, scaling, and duplicate instance removal. The data set thus chosen is ambiguous and has low variances. As a result, when normalizing the features, a MinMax scaling is chosen. The term normalizing refers to the re-scaling of numerical attributes with fixed scale values of 0 and 1, respectively. Scaling the input attributes that depend on magnitude is extremely important. Normalization is a scaling method for shifting and rescaling data set values.

B. Feature Extraction

In general, the feature extraction phase searches the manifest file for intents and permissions. This phase includes two distinct phases:

- Permission type (regular or hazardous) and its number,
- The type of intent (regular or hazardous) and the number of intents.

Furthermore, both intentions and permissions are classified into four major groups: regular permissions, hazardous permissions, regular intents, and hazardous intents [9].

C. Detection Phase

This section combines a convolution neural network and an adaptive red fox optimization algorithm to determine whether an Android application is malicious or normal. Malicious mobile detection is concerned with requesting permission and displaying a variety of malicious codes. The following section discusses the technical backgrounds of CNN and the ARFO algorithm [10].

D. Database Description

There are 12,000 databases in this section, which include both malicious and normal datasets. In this study, we used normal and malicious applications with 12,000 each as training data and 1000 malicious and normal applications as testing data.

3. RESULT

The result section presents the findings of a study that focused on detecting normal and malicious Android applications. To assess the performance of the proposed CNN-ARFO approach, various parameters such as accuracy rate, loss rate, precision, recall, and f-measure were evaluated. The accuracy and loss values for both the training and testing data sets are shown in Figure 1. According to the results of the analysis, the proposed approach achieves the highest testing accuracy rate of 92.1% at the sixth epoch. After the seventh epoch, the training accuracy improves to 92.2%. The training loss reaches a low of about 25% at the second epoch and then drops slightly before stabilizing at 10%. The testing loss begins at 66% and smoothly increases as the epoch value increases. Overall, the results show that the proposed CNN-ARFO approach is effective at detecting malicious applications on Android mobile devices.

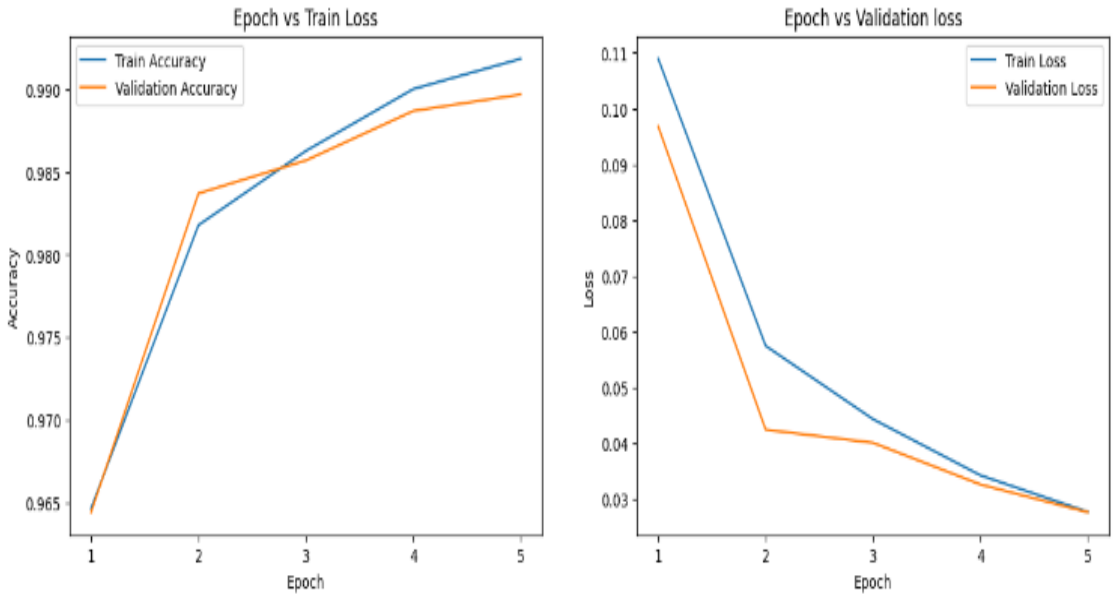

Fig. 2. Epoch Vs Train Loss Evaluation on Epoch 5

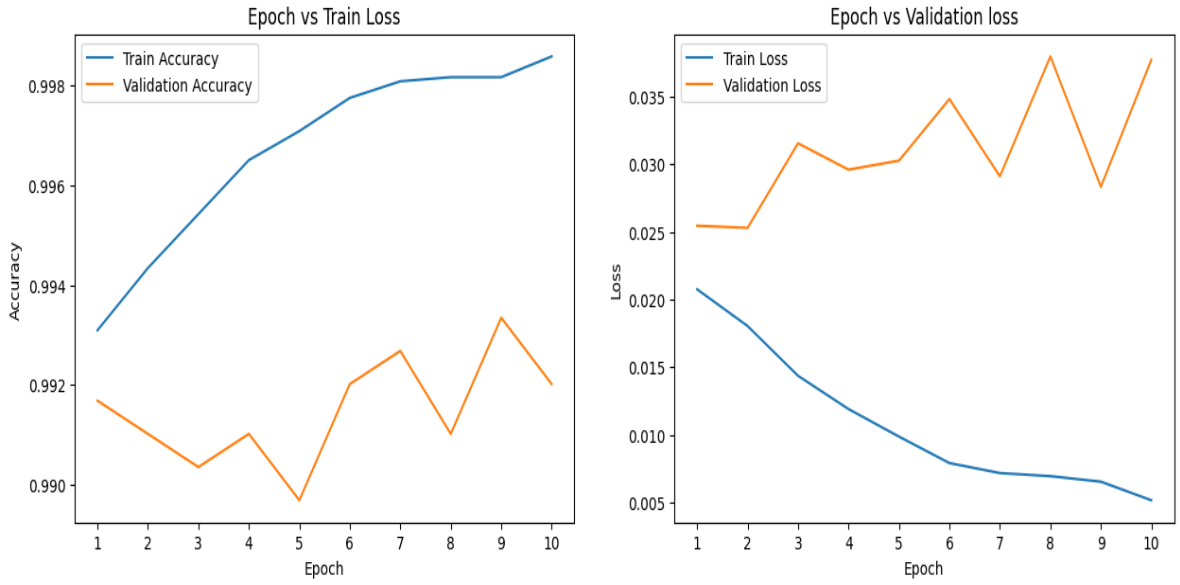

Fig. 3. Epoch Vs Train Loss Evaluation on Epoch 10

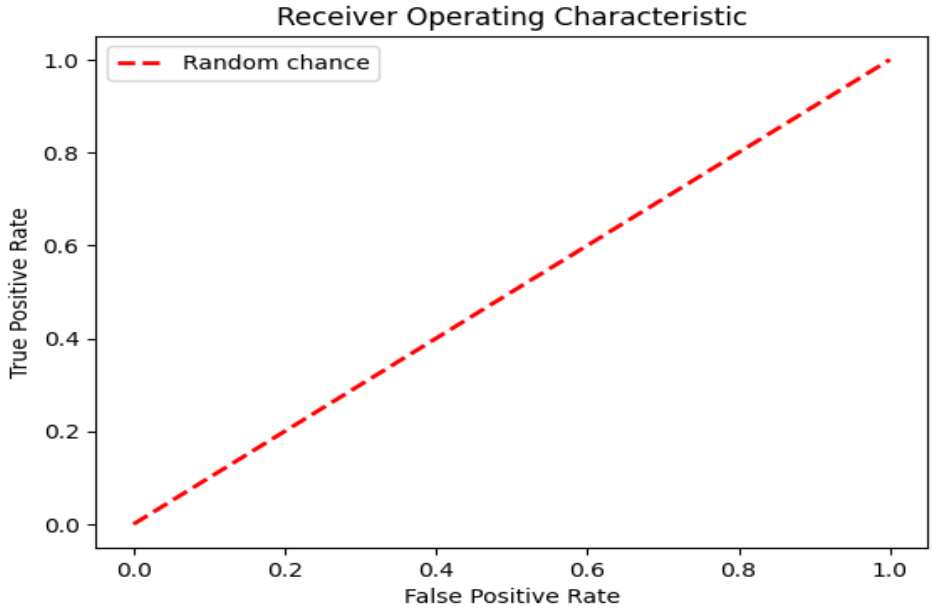

Fig.4. ROC Curve Evaluation on Epoch 5

The proposed optimizer achieved approximately 98.20% precision, while the other approaches obtained precision values ranging from 91.38% to 98.42%. As a result, the analysis shows that the proposed approach outperforms the other approaches tested in terms of both recall and precision values.

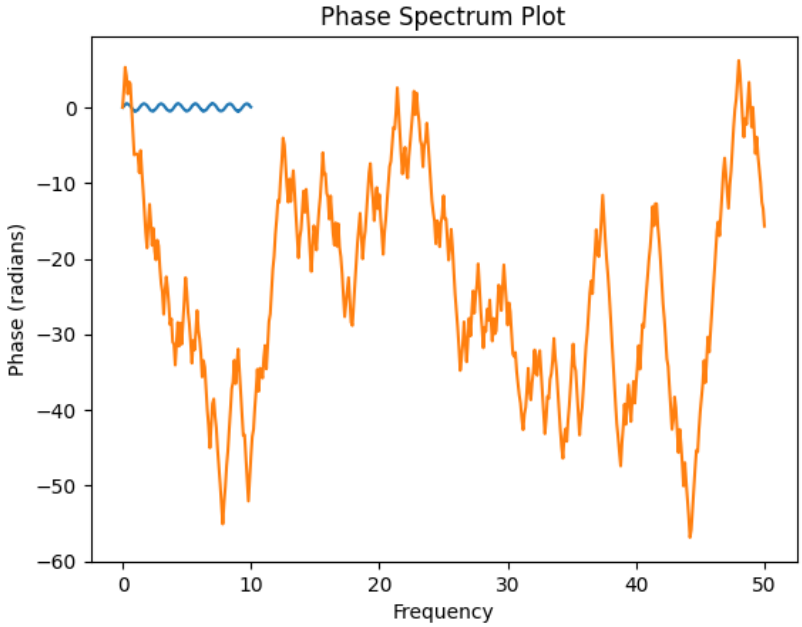

Fig.5. Phase Spectrum Plot on Epoch 5

Figures 3 and 4 depict the proposed method's ROC curve, which represents the classification model's performance at various classification thresholds. The curve is drawn using the CNN-ARFO classifier's true positive and false positive rates at various thresholds. Based on the true positive value, the analysis shows an overall accuracy rate of 94.88% and a false positive rate of 3.39%. The evaluation results show that the proposed approach achieves a high true positive rate with a low false positive rate, which improves malware detection performance. Figure 5 contrasts the proposed CNN-ARFO method with other malware detection methods such as DANN [11], DBN [12], MLBS [13], and DE-CNN[14].

4. CONCLUSION

Numerous research scholars developed machine learning approaches for detecting malicious Android applications, but due to certain drawbacks, effective malware application detection becomes a difficult task. As a result, the proposed CNN-ARFO approach is presented in this paper to detect malicious Android applications. The proposed approach's accuracy rate is based on API call behaviors and permission requests and system file history in this case. To improve detection accuracy, the datasets are first pre-processed using the Minmax scaling approach. Then, for effective feature extraction, each Android application is accomplished by three distinct features: the permission feature, the API call feature, and the statics files feature. The CNN-ARFO approach is then proposed to detect malicious applications, and as a result, it can perform relatively quick classification with minimal computational overhead once training is completed.

References

1. O'Dea, S., & 27, J. (2022, July 27). Smartphone sales worldwide 2007-2021. Statista. Retrieved August 2, 2022, from https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/
2. Sharma, A. (2022, July 26). Key Google Play Store statistics in 2022 you must know. Appinventiv. Retrieved August 1, 2022, from https://appinventiv.com/blog/google-play-store-statistics
3. Połap, Dawid, and Marcin Woźniak. “Red Fox Optimization Algorithm.” Expert Systems With Applications, vol. 166, Elsevier BV, Mar. 2021, p. 114107.
4. Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of Android Malware Detection Approaches based on machine learning. IEEE Access,8,124579–124607. https://doi.org/10.1109/access.2020.3006143
5. Pan, Y., Ge, X., Fang, C., & Fan, Y. (2020). A systematic literature review of Android malware detection using static analysis. IEEE Access, 8, 116363–116379. https://doi.org/10.1109/access.2020.3002842
6. Convolutional Neural Network for classification of Android applications represented as Grayscale Images. (2019). International Journal of Innovative Technology and Exploring Engineering, 8(12S), 835–843. https://doi.org/10.35940/ijitee.l1189.10812s19
7. Hota, A., & Irolla, P. (2019). Deep Neural Networks for Android malware detection. Proceedings of the 5th International Conference on Information Systems Security and Privacy. https://doi.org/10.5220/0007617606570663
8. Wang, X., Zhang, L., Zhao, K., Ding, X., & Yu, M. (2022). MFDroid: A Stacking Ensemble Learning Framework for Android Malware detection. Sensors, 22(7), 2597. https://doi.org/10.3390/s22072597
9. Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2017). Android malware detection using deep learning on API method sequences. ArXiv:1712.08996
10. Wang, X., Zhang, L., Zhao, K., Ding, X., & Yu, M. (2022). MFDroid: A Stacking Ensemble Learning Framework for Android Malware Detection. Sensors, 22(7), 2597. https://doi.org/10.3390/s22072597
11. Wu, Q., Li, M., Zhu, X., & Liu, B. (2020, October 1). MVIIDroid: A Multiple View Information Integration Approach for Android Malware Detection and Family Identification. IEEE MultiMedia, 27(4), 48–57. https://doi.org/10.1109/mmul.2020.3022702
12. Alazab, M., Alazab, M., Shalaginov, A., Mesleh, A., & Awajan, A. (2020, June). Intelligent mobile malware detection using permission requests and API calls. Future Generation Computer Systems, 107, 509–521. https://doi.org/10.1016/j.future.2020.02.002

13. Jiang, X., Mao, B., Guan, J., & Huang, X. (2020, January 25). Android Malware Detection Using Fine-Grained Features. Scientific Programming,2020,1–13. https://doi.org/10.1155/2020/5190138
14. Garg, M., & Dhiman, G. (2020). Deep Convolution Neural Network Approach for Defect Inspection of Textured Surfaces. Journal of the Institute of Electron and Computer, 2, 28-38. https://doi.org/10.33969/JIEC.2020.2100