Measurement: Sensors 32 (2024) 100955

Contents lists available at ScienceDirect

Measurement: Sensors

journal homepage: www.sciencedirect.com/journal/measurement-sensors

Automated malware detection using machine learning and deep learning approaches for android applications

S. Poornima [a,*], R. Mahalakshmi [a]

[a] *Department of Computer Science Engineering, Presidency University, Bangalore, 560064, India*

A R T I C L E I N F O

Keywords:
Anti-malware system
Malware analysis
Deep learning
Deep belief network
Cyber security

A B S T R A C T

The popularity of mobile devices has aided in the development of the Android platform. Malware is one of the most frequent cyberattacks with its prevalence growing daily across the network. Due to these flaws, a hacker can simply access the mobile device's private data. To alleviate this issue, this paper proposes a novel Malware Attack Detection in android using deep belief NETwork (MAD-NET) which accurately detects and mitigates the malware attacks and enhances the security of the devices. Initially, the benign and malicious data's from the CICAndMal2017 datasets are given as a input to the feature extraction process. In feature extraction, the data's are classified into two types such as signature-based data and behaviour-based data. The extracted features are converted as a sequence data and fed into the classification phase. In the classification phase, the benign and malicious data's are classified by using DBN technique. After being classified, the detected malicious data's are converted into a detection report for further processing. The proposed MAD-NET approach is evaluated by using CICAndMal2017 datasets and it is simulated by using Network Simulator. The proposed MAD-NET is evaluated by using many performance metrics such as accuracy, precision, recall, and F1-score. The MAD-NET technique achieves an overall accuracy of 99.83 % for Deep Belief Network (DBN), than Artificial Neural Networks (ANN), Generative Adversarial Network (GAN) and Long Short-Term Memory Network (LSTM) technique of 93.11 %, 96.75 %, 94.42 % respectively to detect and mitigate the android devices from malware attacks.

1. Introduction

Nowadays, people's life depends heavily on their mobile gadgets. Around 6.4 billion people use smartphones worldwide, and Statista, estimates that figure will rise by several hundred million users over the next few years. By the first quarter of 2021, the Google Play Store, the biggest app store in the world, will have 3.48 million apps available. Mobile applications are used for the majority of daily tasks, including internet shopping, bill paying, and mobile banking [1]. It has raised the danger of identity theft including private data like bank account information, credit and debit card numbers, and ATM PINs. Account manipulation, bogus track redirection to mobile money servers, and brute force attempts to steal users' PINs are all examples [2,3]. Future smartphones will be compatible with a variety of applications, including AI-powered health care, mobile edge computing, and cutting-edge industry apps, so they must be furnished with the most cutting-edge high-level security measures [4,5]. Every software asks users for a list of permissions when they install it. A user can deduce the behaviour of any programme when they have the proper rights. Key permissions for capabilities and expected permission requests can be identified, which makes it easier to take advantage of anomalous application behaviour and gives consumers a straightforward risk alert [6]. The permission-based approach alerts the user before installation in this way [7,8]. The user is given the chance to consider the dangers of granting the application access to their mobile device.

According to Statista, 529,48 new malware families are created by cyber criminals. As a result, it's important to find spyware on Smart Mobiles. If malware is discovered during installation, malicious apps can be prevented from installing. To stop this significant cyber onslaught, we need a scalable malware detection approach that can swiftly and accurately identify malware apps [9,10]. However, scaling the detection for several apps is a challenging issue. Malware identification is a significant issue that needs to be solved right away.

In order to defend against Android malware, this paper proposes a Malware Attack Detection in android using deep belief NETwork (MAD-NET) for effective malware detection and classification. The major contribution of the MAD-NET is as follows.

* Corresponding author.
E-mail addresses: poornima.spa@gmail.com (S. Poornima), mahalakshmi@residencyuniversity.in (R. Mahalakshmi).

https://doi.org/10.1016/j.measen.2023.100955
Received 29 May 2023; Received in revised form 22 August 2023; Accepted 27 November 2023
Available online 30 November 2023

❖ Initially, the benign and malicious data's from the CICAndMal2017 datasets are given as input to the feature extraction process.
❖ In feature extraction, the data's are classified into two types such as signature-based data and behaviour-based data.
❖ After feature extraction, the extracted features are converted as a sequence data and fed into the classification phase [11]. The benign and malicious data's are classified by using DBN technique.
❖ After being classified, the detected malicious data's are converted into a detection report.
❖ The proposed MAD-NET framework is evaluated by using many performance metrics such as accuracy, precision, recall, and F1-score.

The remaining section of the ML and DL-based MAD-NET techniques is organized as follows. The related studies are explained in Section 2. The proposed MAD-NET framework along with a detailed explanation is included in Section 3. Section 4 comprises the result and discussion of the MAD-NET model and Section 5 holds the conclusion and future scope.

2. Related works

In recent days, researchers have introduced several DL and Machine Learning (ML) methods, specifically to improve the detection and classification of malware attack detection in android devices. Some of the strategies are covered briefly in this section.

In 2019, Swati Nautiyal et al. [12] suggested that Artificial Neural Networks (ANNs) are used to categorize cloud server users and potentially reduce EDoS attacks in the cloud, while Genetic Algorithms (GA) are used to optimize the attributes of each server using applicable fitness functions this paper discusses numerous EDoS mitigation solutions as well as their diverse approaches. Considerations and expectations are used to compare the proposed technique to existing procedures.

In 2020 Sweta Mittal & Jayasimha [13] suggested that phishing assault carries the potential of being hacked at any time, resulting in the loss of the user's identity, finance, or highly secret information. Phishing attacks can be used to catch people who use cloud resources. After gaining complete access control over the real user, the attacker can impersonate the user and gain access to all cloud data services [14]. The legitimate user will suffer financial losses as a result of the phishing attack [15].

In 2021 Almahmoud, M. et al. [16] suggested Android malware detection based on recurrent neural networks. A novel model is developed by combining four static features such as permissions, API calls, monitoring system events, and permission rate. According to the experimental results, the proposed approach scored 98.58 accuracy and successfully predicted 98.36 % of the dataset.

In 2022 Al-Naji, F.H. and Zagrouba, R. et al. [17] suggested Authentication for IoT environment using blockchain. Compared to the traditional classifier, the Euclidean-SVM augmented classification methodology outperformed the CAB-IoT method. The AT&T dataset is evaluated and tested in this technique and achieves an overall accuracy of 99.97 %. The CAB-IoT method has limits, though, when it comes to conducting thorough tests to gauge how strong the suggested solution is against different types of threats.

In 2022 Sarhan, M. et al. [18] suggested a learning system linked to a decentralized blockchain platform that enables secure collaborative IoT intrusion detection and protects privacy. To protect and maintain the IoT network's integrity, this framework develops an effective, highly efficient, and secure ML-based IDS. However, ML-based secured networks face a significant security risk in the discovery of previously unknown attack classes because to the lack of CTI between enterprises.

In 2023, Gómez, A. and Muñoz, A. [19], suggested Deep Learning-Based Attack Detection and Classification in Android Devices. The effectiveness of our proposed model is validated using well-established datasets such as CICMalDroid2020, CICMalDroid2017, and CICAndMal2017 and achieves an impressive F1 score of 98.9 % and a false positive rate of 0.99 %. However, the computational complexity of the suggested technique is very high.

In 2023, Manzil, H.H.R. and Manohar Naik, S. [20], suggested Android malware category detection using a novel feature vector-based machine learning model. The suggested approach is evaluated using machine learning and deep learning methods. The experiments shows that this suggested approach significantly improves the efficiency of the detection model. The suggested model is evaluated using CICMalDroid dataset and achieves a detection accuracy of 98.70 %. However, the security of the suggested framework is need to be improved.

From the related studies, various ML and DL techniques are used for the detection and classification of malware attack detection in android devices [21,22]. The accuracy rate, specificity, recall, and sensitivity of the aforementioned methods are low. A MAD-NET technique is proposed to solve this problem by detecting and classify the malware in android devices. The major contribution of the MAD-NET is as follows, Initially, the benign and malicious data from the CICAndMal2017 datasets are given as input to the feature extraction process. In feature extraction, the data are classified into two types such as signature-based data and behaviour-based data [23]. After feature extraction, the extracted features are converted as a sequence data and fed into the classification phase. The benign and malicious data are classified by using DBN technique. After being detected, the detected malicious data's are converted into detection report.

3. Android malware categories and families

Any malicious programme, including Android malware, designed to harm the target mobile device by engaging in some illegal activity is referred to as mobile malware. Each subcategory of malware has some distinctive traits that set it apart from the others. Like humans, malware for Android also grows [24,25]. There are a number of malware families that fall under each malware category. Researchers and cybersecurity specialists face an open challenge as a result of the unparalleled threat posed by Android malware, which is the source of numerous internet security issues [26]. Only quick detection and mitigation of the malware samples will eliminate this threat. Understanding the many types and families of Android malware is essential for accomplishing this. The below description shows the list of android malwares and its family.

1. **File Infector:** A file infection is when malware is found hiding in an APK file. All the information required to operate the application is contained in the Android Package Kit, or APK. Malicious files are installed using APK files. Malware is utilized once APK files have been installed. The APK file contains all Android applications, including games, word processors, navigational tools, and others.
2. **Riskware:** Software that poses a danger to system security vulnerabilities is referred to as risky software. Despite being a legitimate tool, it is used to track users' online activity and direct them to malicious websites. It is also known as malware, and the effectiveness of its operations has an impact on the device's security [27]. Among those that are frequently discovered are the malware families badpac, mobilepay, wificrack, triada, skymobi, deng, jiagu, smspay, smsreg, and tordow.
3. **Ransomware:** It is malware that encrypts system files and folders by preventing access to them and then makes a big request to be decrypted so that access can be restored. The harmful ransomware strains congur, masnu, fusob, jisut, koler, lockscreen, slocker, and smsspy are some of the more prevalent types.
4. **Trojan:** This virus impersonates trustworthy applications. They prowl the boot system while stealing data from the device. The gluper, lotoor, rootnik, guerilla, gugi, hqwar, obtes, and hypay Trojan families are among the most well-known.
5. **Adware:** Adware is harmful software that displays advertisements. It is a malicious program that shows people unwanted adverts on the

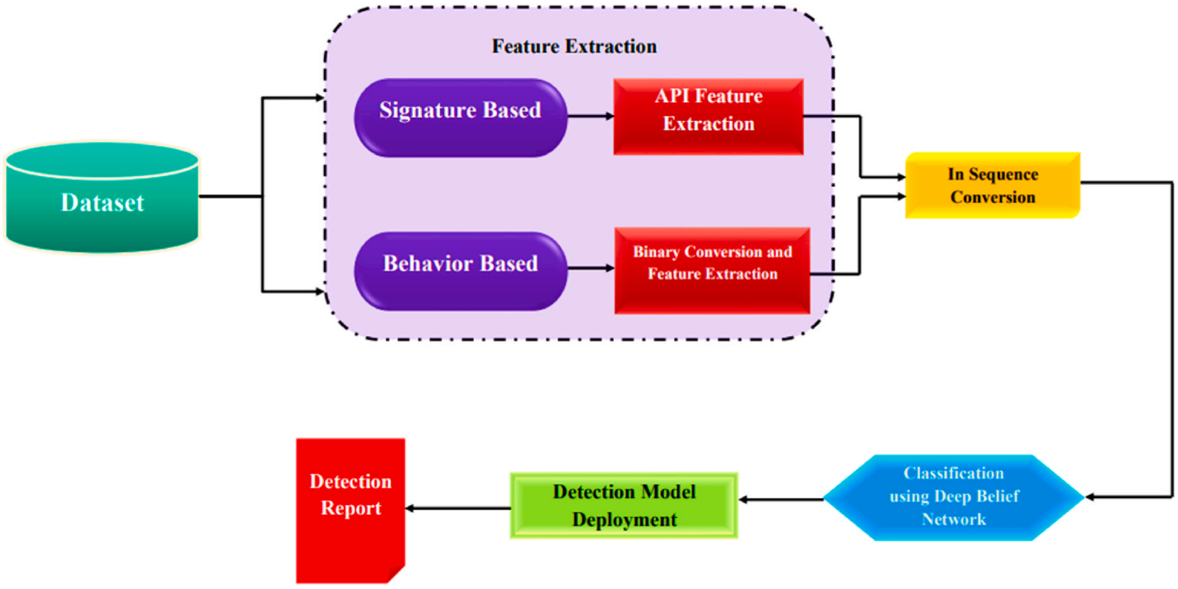

Fig. 1. Malware classification and detection approach.

screen, especially when they are utilizing web services. The adware tempts customers with alluring offers to click on flashing advertisements for profitable goods. The creator of this annoying program is compensated when a user clicks on an advertisement. Gexin, Batmobi, Ewind, Shedun, Pandaad, Appad, dianjin, Gmobi, Hummingbird, Mobisec, Loki, kyhub, and Adcolony are a few well-known adware brands.

6. **Backdoor:** Smartphones can be accessed privately through the back doors. In other words, backdoors circumvent authentication and give attackers additional permissions, enabling them to access cellphones whenever they want. Backdoors make it possible to launch remote attacks on a target device without being physically present there. Typical families of Android backdoor malware include Mobby, kapuser, hiddad, dendroid, levida, fobus, moavt, androrat, kmin, pyls, and droidkungfu.
7. **Scareware:** Scareware is a technique that uses fear to persuade users to download or purchase malicious software. Three well-known brands of scareware include Avpass, Mobwin, and Fakeapp.
8. **Spyware:** Spyware can potentially steal personal data when it is installed on a machine. The information that spyware gathers may be made available to businesses, outside agents, or advertisers [28]. Then, this data is abused for bad intentions. Spynote, qqspy, spydealer, smsthief, spyagent, spyoo, smszombie, and smforw are examples of common spyware families.
9. **PUA:** Potentially Unwanted Applications, or PUAs, are legitimate free programs that include PUA. Potentially undesirable programs (PUPs) is another name for them. Apptrack, Secapk, Wiyun, Youmi, Scamapp, Utchi, Cauly, and Umpay are a few well-known PUA malware lines for Android smartphones.

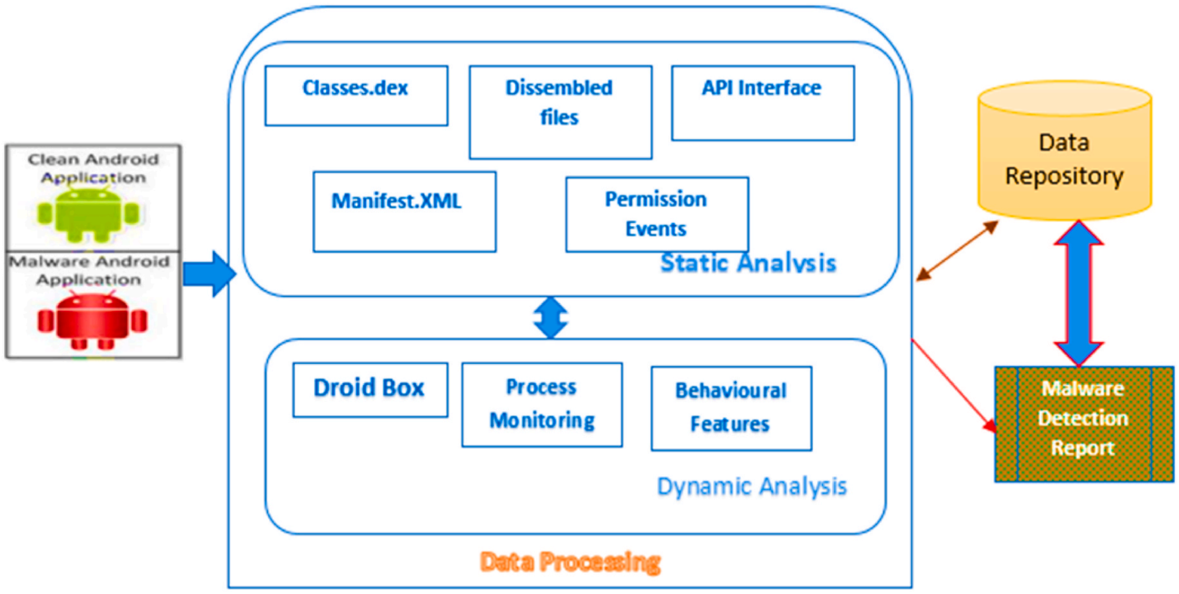

Fig. 2. Android feature extraction and processing [30].

4. The MAD-NET methodology

In this work, a novel MAD-NET technique is developed for the effectual detection of Android malware among benign applications, thereby accomplishing cybersecurity. Initially, the benign and malicious data's from the CICAndMal2017 datasets are given as an input to the feature extraction process. In feature extraction, the data's are classified into two types such as signature-based data and behaviour-based data. The API feature extraction is performed in the signature-based data and initially, the behaviour-based data's are converted into binary data for feature extraction. After feature extraction, the extracted features are converted as a sequence data and fed into the classification phase. Further, the converted sequence data's are fed to the DBN model to classify benign and malicious data. After being classified, the detected malicious data's are converted into detection report for further processing. Fig. 1 represents the overall procedure of the MAD-NET approach.

4.1. Dataset creation and feature extraction

The Android Operating environment distributes and installs apps on Android devices using the Android Package Kit (APK) file format. Everything a mobile application needs to properly install and run is included in an APK bundle. The APK TOOL [29] is used to decompile malicious and helpful files obtained from "CICAndmal2017″ in order to extract the necessary info. Use the AAPT2 tool to extract permissions and intents from the AndroidManifest.xml. Fig. 2 represents the framework proposed for effective feature extraction.

The classification model's main goal is to predict a class label from a set of available possibilities. There are two basic types of class problems: multiclass classifications, where the class version is used to anticipate a few lessons, and binary classifiers, where the easiest lessons to classify are. The detection of malware on Android may be viewed as a binary class problem from the standpoint of system studies. We use binary classification to assess whether an Android application is safe to use or dangerous based on static features in order to accomplish our goal in this study. This work created six supervised machine learning models to distinguish between safe Android apps and malware utilizing 215 static features. Therefore, utilizing trained and supervised machine learning models of K-Nearest Neighbor (K-NN), Decision Tree (DT), machine support vectors (SVM), Random Forests (RF), Nave Bayes (NB), and logistic regression (LR), it is possible to identify unknown Android harmful apps. According to Section 5, each classification method uses a unique mathematical strategy to distinguish between classes.

4.2. Malicious software detection using signatures

In actuality, antiviral software with direct linkages began to use static analysis signatures more frequently in order to assess the effects of infestations on software, malware detection has generally relied on static tests rather than on element behavioural methodologies. Using a pre-defined database of known assaults, the signature-based technique looks for interferences. The signature database must be updated frequently for this technique to detect malware in a variety of setups. Signature-based approaches are less successful in spotting dangerous conduct because of the dynamic nature of transportable malware. Signature-based techniques use traditional explanations or unique rough bit patterns, known as impressions, to organise the malicious text [31]. It is explicitly tested to determine if the material's static attributes are defined in nefarious project management files. Advantage of completeness is tracked by signature-based approaches for all addressable options, which is the benefit.

4.3. Malware detection based on process behaviour

To monitor virus activity, dynamic analysis runs malware that operates for several minutes in a simulated sandbox environment. The most used method is system call level monitoring [32–35]. A report of detected malware activity is generated sequentially based on malware activity. Reports are usually written with behaviour-based analysis in mind, and they usually include all system calls and their arguments. Images of systems before and after infection are compared using a distinct technique termed "forensic snapshot comparison." The latter is significantly different from system-call monitoring in that it does not take into consideration the glass transition of forensics events.

Despite the benefits of dynamic analysis, it is ineffective because the virus must keep an eye on the system for several minutes. Additionally, some contemporary malware is created to function as a "virtual machine" that, upon detection, ceases to cause harm [36]. For behaviour-based techniques and recorded runtime operations, a specific model must be operated in a sandboxed environment. Virtualization and simulation are both used by the dynamic evaluation framework to remove jobs [37–39]. To find malevolent expectations that are considered malicious conduct, in-depth examination of product design and programming is performed [40].

4.4. Deep Belief Network (DBN)

The DBN is used to recognize android malware attacks. DBN can be used to classify WiFi frames into Authorized and Unauthorized access. A DBN, which can extract certain features from the original data, is made by stacking the finished Boltzmann devices numerous times. DBN seeks to boost the likelihood of the training data.

The Low Restricted Boltzmann Machine (RBM) receives the initial DBN input after which training starts. The formation procedure then proceeds gradually up the hierarchy until the top-level RBM, which contains the DBN outputs, is generated. Uses for the source function include:

$$\rho\left(L_x, L_y\right)=F_p^{-1} * e^{-F\left(L_x, L_y\right)} \tag{1}$$

Where, L_{xj} and L_{xk} represent the binary state the visible unit j and hidden unit k, respectively and F_p identifies the partition function that was developed by adding potential pairings of concealed and exposed units,

$$F_p=\sum_{L_x, L_y} e^{-F\left(L_x, L_y\right)} \tag{2}$$

Where, $F(L_x, L_y)$ is determined using the following equation, which stands for the energy of the total configuration of hidden and visible units:

$$F\left(L_x, L_y\right)=-\sum_{j=1} b_j L_{xj}-\sum_{k=1} c_k L_{yk}-\sum_{j,k} L_{xj} L_{yk} Z_{jk} \tag{3}$$

Where, b_j *and* c_k describe the blatant and covert unit biases respectively, and Z_{jk} denotes the ratio of visible to hidden units. In order to update the RBM weight,

$$\Delta Z_{j,k}=F_t\left(L_{xj} L_{yk}\right)-F_m\left(L_{xj} L_{yk}\right) \tag{4}$$

Where, $F_t(L_{xj}L_{yk})$ and $F_m = (L_{xj}L_{yk})$ is a symbol that represents the expectation in the training set and the model individually. Finally, a DBN is employed for classifying WiFi frames into authorized and unauthorized access. Hence, the DBN is used to prevent the android device from unauthorized access.

4.5. Feature engineering process for malware classification and detection

Purpose of this approach is to teach a learning agent to select characteristics for categorization in a sequential manner [41–44]. The feature selection process can be described as a Markov Decision Process under the presumption that a feature that performs well in one

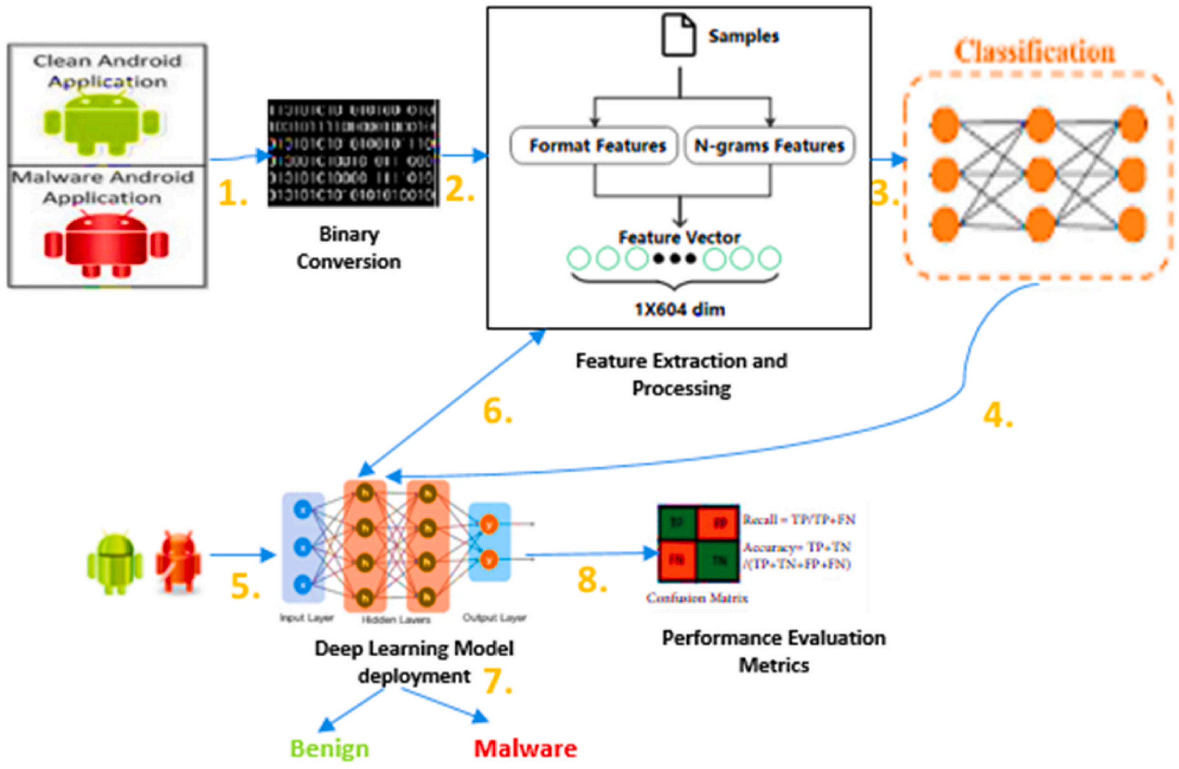

Fig. 3. Android malware classification and detection framework.

classification test should similarly contribute to the result of another classification challenge. The agent sequentially selects features under the -greedy strategy until it reaches a termination state [45,46]. The graph in Fig. 3 demonstrates how API call attributes, assembly traits, and binary characteristics are all effective malware detection methods. These elements leverage artificial intelligence methods for determining and spotting harmful activities.

There are two categories of analysis in our database.

1. We performed two different sorts of analyses on the database files that were the focus of our inquiry: The features resulting from static analysis were taken straight from the applications' source code. Additionally, the following characteristics could potentially be taken from APK applications:
   a. Permissions: The permissions system is used on Android platforms to safeguard user privacy. Its specific objective is to ask the application for permission to access data and system functionalities like calls and the camera because most of the time these programmes wish to ask for a particular set of permissions. It is prudent to carefully review this before obtaining the desired features as a consequence.
   b. Intention: This section is used to mention the subject of the application.
   c. Suspicious API calls: These are a group of interfaces for applications that can be used to get unauthorized access to private data, resources, and occasionally harmful behaviour.
   d. Restricted API: calls are permissions-protected but otherwise have roughly the same goals as suspicious API queries.
2. Dynamic analysis: In a secure environment where the application implementation process is documented, this sort of analysis focuses on the features that can be derived during application implementation. According to our research, many dynamic features from Android applications can be retrieved.

Then, it proceeds to the stage of selecting features once the features extraction phase is finished and novel patterns have been found. One of the most crucial processes is the feature selection process, which aims to select the attributes from a pool of recently discovered attributes in order to maximise accuracy, decrease complexity, and, at the same time, prevent over fitting. To identify malware in apps, researchers have historically used a variety of feature categorization techniques [47–49]. We specifically use the feature rank strategy in this attempt because it makes use of certain essential components in the organisation of features and can select the proper features required to create malware detection models. Some of the topmost ranked features are considered and defined in Table 1.

Table 1
Some hand-picked features for malware detection.

Features	Feature type	Analysis Type	Details
Permissions	Signature based	Static	Internal words
API Calls	Behaviour based	Dynamic	Hashing tricks
Strings Extracted	Signature Based	Static	Internal words
Native Commands	Signature based	Static	Opcode verification
XML Elements	Signature based	Static	Opcode verification
Meta Data	Behaviour Based	Static	Opcode verification
.dex file	Behaviour based	Dynamic	Process extraction
Task Intents	Signature based	Static	Internal words
Process ID	Behaviour based	Static	Process tracking
SMS	Signature based	Static	Opcode verification
Power Usage	Behaviour based	Static	Hashing tricks
Log files	Behaviour based	Dynamic	Process monitor
Internet Traffic	Behaviour based	Static	Process monitor
System calls	Signature based	Static	Opcode verification
Battery Performance	Signature based	Static	Process monitor
Process Info	Behaviour based	Dynamic	Process Monitor

Table 2
Validation metrics for Malware Classification.

Classifier	Feature's Used	Feature Dimensions	Accuracy	Precision	Recall	F1 Score
KNN	11	100,200	95.59	94.59	96.89	95.87
Decision Tree	12	100,200	97.78	96.65	96.34	97.59
Random Forest	11	200,300	95.56	97.05	97.87	98.45
Naïve Bayes	13	100,300	98.00	98.90	97.56	97.54
SVM	11	200,300	97.65	98.99	99.45	98.87
CNN	15	100,200,300	98.99	99.37	98.69	99.12

Table 3
Validation metrics for Malware Detection.

Classifier	Feature's Used	Feature Dimensions	Accuracy	Precision	Recall	F1 Score
KNN	11	100,200	97.78	98.7	98.35	97.23
RNN	12	100,200	95.56	97.55	97.34	96.34
CNN	11	200,300	98.00	98.76	98.23	98.45
LSTM	13	300	97.65	98.23	98.15	98.12
CLSTM	11	100,200,300	98.99	98.87	98.76	98.89

5. Result and discussion

5.1. Dataset description

In this study, the android malware detection outcomes of the MAD-NET technique are examined on the CICAndMal2017 Dataset. In order to avoid the behaviour of increasingly advanced malware samples that change their behaviour to give false results when received, the CICAndMal2017 dataset was created by running malicious and benign programs on cell phones. The researchers put 5000 markers, including 426 malicious and 5065 benign samples, on actual devices. They then collected data for each marker in three states and recorded traffic patterns. In order to create datasets. The complete dataset currently consists of 2126 samples and 2,583,878 network hits, each of which represents a single instance of the Android app running on a mobile device. Every sample's network flows are recorded while it is being executed. Each network stream has 84 features overall that were recorded. Each of these samples has three tags: a binary tag that indicates whether the pattern is harmful, a category tag with five possible values that identifies a particular malware type, and a family map with 42 different values that designate a particular malware family.

5.2. Performance evaluation

The efficiency of the detection model network is quantitatively assessed in this article using the five following parameters as assessment indicators. The calculation formula is shown below:

$$Accuracy=\frac{RC+RA}{RC+RA+FC+FA}$$

$$Recall=\frac{RC}{RC+FC}$$

$$Precision=\frac{RC}{RC+FC}$$

$$F1\ Score=\frac{2RC}{2RC+FC+FA}$$

$$AUC=\frac{\sum rank_{ins}-\frac{M\ X\ (1+M)}{2}}{M\ X\ N}$$

5.3. Result analysis

To validate the outcomes, seven algorithms were examined using two exclusion strategies and k-fold validation. Eighty percent of the data

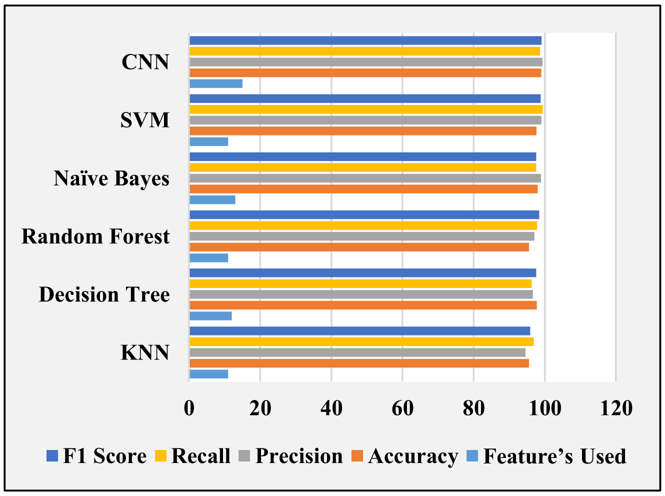

Fig. 4. Representation of validation metrics for classification techniques.

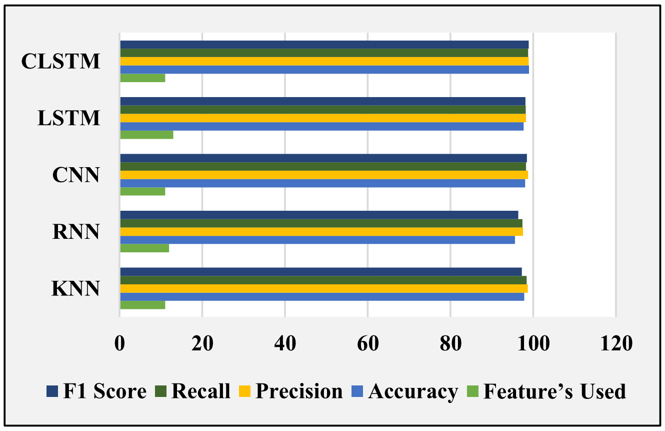

Fig. 5. Representation of validation metrics for Malware Detection techniques.

are utilized for retention validation and twenty percent are used for k-fold cross-validation. The data are split into training and testing datasets. Ten times are used to partition the data, with each serving as nine (9) training sessions and one test. The average of the accuracy over all folds is then given as the final accuracy.

The performance ratio for classification strategies employing various

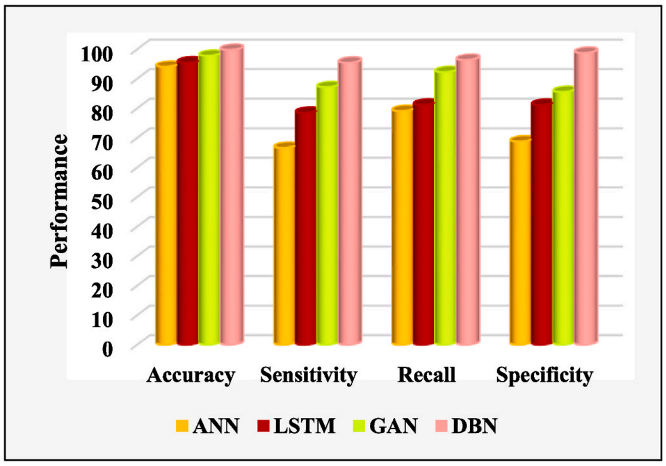

Fig. 6. Performance comparison of DL Networks.

segmentation approaches is shown in Tables 2 and 3 using the substantial training dataset. NN, SVM, and J48 were the most effective algorithms in terms of effectiveness and false positives. The Naive Bayes assumption that characteristics are independent is the cause of NB's low efficiency, which leads to a low degree of accuracy and a large false positive rate. SVM is the technique with the highest accuracy for employing a subset of functions. Using 80 % of the data for training and 20 % for testing, the efficiency ratio of the classification algorithms using various feature extraction techniques was computed. The validation metrics for several malware detection and classification strategies are shown in Figs. 4 and 5.

Fig. 6 displays a graphical comparison of the proposed and existing approaches with various criteria, such as accuracy, sensitivity, recall and specificity. This graph demonstrates the effectiveness of the proposed method for detecting malware in Android devices with existing techniques. This comparison demonstrates that the proposed MAD-NET technology outperforms the current approaches. While existing models like ANN have an accuracy of 93.11 %, the GAN of 96.75 %, LSTM of 94.42 % and the Proposed MAD-NET approach has a maximum accuracy of 99.83 % for DBNs. The ANN, LSTM, GAN, and existing approaches significantly improved the 14.3 %, 7.5 %, and 4.05 % sensitivity of the proposed methods. The proposed method for recall is higher than existing approaches by 8.6 %, 7.6 %, and 2.0 %.

6. Conclusion

This study focuses on the identification and tracking of malware as well as the usage of system mastering and deep mastering for character restoration. This paper proposed MAD-NET, an automated hybrid analysis framework for Android Malware Detection. This framework was evaluated with 41,125 android applications with the inclusion of 450 Static, 540 dynamic features with their comparison report helping the researchers to do further developments in this area. The results presented here clearly shows that the framework achieved high accuracy performance with Deep Neural Networks than existing Machine Learning based architectures. This framework explores the malware classification and detections using hybrid analysis techniques on the reliable dataset, collected as primary and secondary resources. This work relies on effective malware classification also with reinforcement learning makes the system perform dynamic analysis. Future work could involve adapting self-adaptation, recently introduced and researched for intrusion detection systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support. Also like to acknowledge the Dean - Research and Innovation Presidency University, Bangalore for providing required lab resource.

References

[1] A. Damodaran, F. Di Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, Journal of Computer Virology and Hacking Techniques 13 (1) (2017) 1–12, https://doi.org/10.1007/s11416-015-0261-z.
[2] W. Ahmed, A. Rasool, A.R. Javed, Security in next generation mobile payment systems: a comprehensive survey, IEEE Access 9 (2021) 115932, https://doi.org/10.1109/ACCESS.2021.3105450.
[3] A. Ananya, T.R. Aswathy, P.G. Amal, P. Swathy, Vinod, S. Mohammad, SysDroid: a dynamic ML-based android malware analyzer using system call traces, Cluster Comput. 23 (4) (2020) 2789–2808, https://doi.org/10.1007/s10586-019-03045-6.
[4] P.K.R. Maddikunta, Q.V. Pham, Prabadevi, "Industry 5.0: a survey on enabling technologies and potential applications,", Journal of Industrial Information Integration 26 (2022) 100257.
[5] C. Nicholas, Malware Analysis in the Large vs. Malware Analysis in the Small, 2017.
[6] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: Proceedings of the 34th ICML, vol. 70, 2017, pp. 933–941. JMLR. org.
[7] B.C. Ooi, K.-L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, Y. Tung Singa, A distributed deep learning platform, in: ACM Multimedia, ACM, 2015, pp. 685–688.
[8] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, Malware classification with recurrent networks, in: 2015 IEEE ICASSP, IEEE, 2015, pp. 1916–1920.
[9] J. Gajrani, U. Agarwal, V. Laxmi, EspyDroid+: precise reflection analysis of android apps, Comput. Secur. 90 (2020), https://doi.org/10.1016/j.cose.2019.101688.
[10] D. Gibert, C. Mateu, J. Planes, R. Vicens, Classification of malware by using structural entropy on convolutional neural networks, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, https://doi.org/10.1609/aaai.v32i1.11409.
[11] F. Ahmed, H. Hameed, M.Z. Shafiq, M. Farooq, Using spatio-temporal information in api calls with machine learning algorithms for malware detection, in: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, 2009, pp. 55–62, https://doi.org/10.1145/1654988.1655003.
[12] S. Nautiyal, R.K. Challa, S. Wadhwa, Mitigating economical denial of sustainability (EDoS) in cloud environment using genetic algorithm and artificial neural network, Int. J. Innovative Technol. Explor. Eng. India 8 (10) (2019).
[13] Sweta Mittal, S.R. Jayasimha, Detection of phishing attacks using content analysis in the cloud, Int. J. Recent Technol. Eng. 9 (1) (2020) 2622–2625.
[14] W. Liu, P. Ren, K. Liu, H.-x. Duan, Behavior-based malware analysis and detection, in: 2011 First International Workshop on Complexity and Data Mining, IEEE, 2011, pp. 39–42.
[15] M. Rhode, P. Burnap, K. Jones, Early-stage malware prediction using recurrent neural networks, Comput. Secur. 77 (2018) 578–594.
[16] M. Almahmoud, D. Alzu'bi, Q. Yaseen, ReDroidDet: android malware detection based on recurrent neural network, Procedia Comput. Sci. 184 (2021) 841–846.
[17] F.H. Al-Naji, R. Zagrouba, CAB-IoT: continuous authentication architecture based on Blockchain for internet of things, Journal of King Saud University-Computer and Information Sciences 34 (6) (2022) 2497–2514.
[18] M. Sarhan, W.W. Lo, S. Layeghy, M. Portmann, HBFL: a hierarchical blockchain-based federated learning framework for collaborative IoT intrusion detection, Comput. Electr. Eng. 103 (2022) 108379.
[19] A. Gómez, A. Muñoz, Deep learning-based attack detection and classification in android devices, Electronics 12 (15) (2023) 3253.
[20] H.H.R. Manzil, S. Manohar Naik, Android malware category detection using a novel feature vector-based machine learning model, Cybersecurity 6 (1) (2023) 6.
[21] K. Santosh Jhansi, S. Chakravarty, R.K.P. Varma, Feature selection and evaluation of permission-based android malware detection, in: Proceedings of the 4th International Conference on Trends in Electronics and Informatics, ICOEI, vols. 795–799, 2020. Tirunelveli, India.
[22] Z. Salehi, M. Ghiasi, A. Sami, A miner for malware detection based on api function calls and their arguments, in: The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), IEEE, 2012, pp. 563–568.

[23] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, A. Smola, Feature Hashing for Large Scale Multitask Learning, 2009 arXiv preprint arXiv:0902.2206.
[24] R. Islam, R. Tian, L.M. Batten, S. Versteeg, Classification of malware based on integrated static and dynamic features, J. Netw. Comput. Appl. 36 (2) (2013) 646–656.
[25] X. Jiang, B. Mao, J. Guan, X. Huang, Android malware detection using fine-grained features, Sci. Program. 2020 (2020) 5190138.
[26] F. Xiao, Z. Lin, Y. Sun, Y. Ma, Malware detection based on deep learning of behavior graphs, Math. Probl Eng. 2019 (2019) 1–10.
[27] Y. Yang, Z. Wei, Y. Xu, H. He, W. Wang, DroidWard : an Effective Dynamic Analysis Method for Vetting Android Applications, Cluster Computing, 2016, p. 21.
[28] W. Zhang, H. Wang, J. He, P. Liu, DAMBA: detecting android malware by ORGB analysis, IEEE Trans. Reliab. 69 (1) (2020) 55–69.
[29] Y. Qiao, Y. Yang, L. Ji, J. He, Analyzing, malware by abstracting the frequent itemsets in api call sequences, in: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE., 2013, pp. 265–270.
[30] Q. Wu, M. Li, X. Zhu, B. Liu, MVIIDroid: a multiple view information integration approach for android malware detection and family identification, IEEE Multimedia 27 (4) (2020) 48–57.
[31] S.S. Hansen, T.M.T. Larsen, M. Stevanovic, J.M. Pedersen, An approach for detection and family classification of malware based on behavioral analysis, in: 2016 ICNC, IEEE, 2016, pp. 1–5, https://doi.org/10.1109/ICCNC.2016.7440587.
[32] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, "DL-Droid: deep learning based android malware detection using real devices,", Comput. Secur. 89 (2020) 101663 https://doi.org/10.1016/j.cose.2019.101663.
[33] R. Angaveloo, W. Wang Jing, C. Kang Leng, J. Abdullah, DATDroid: dynamic analysis technique in android malware detection, Int. J. Adv. Sci. Eng. Inf. Technol. 10 (2) (2020) 536–541.
[34] M. Ahmad, V. Costamagna, B. Crispo, F. Bergadano, Y. Zhauniarovich, StaDART: addressing the problem of dynamic code updates in the security analysis of android applications, J. Syst. Software 159 (2020), https://doi.org/10.1016/j.jss.2019.07.088.
[35] Y. Chen, W. Fang Li, "Android malware identification based on traffic analysis," in Lecture Notes in Computer Sciencevol, LNCS 11632 (2019) 293–303, https://doi.org/10.1007/978-3-030-24274-9_26.
[36] CEA, The Cost of Malicious Cyber Activity to the U.S. Economy, 2018. https://www.whitehouse.gov/wpcontent/uploads/2018/03/The-Cost-of-Malicious-CyberActivity-to-the-U.S.-Economy.pdf.
[37] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2016, pp. 137–149.
[38] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, G. Vigna, Polymorphic worm detection using structural information of executables, in: International Workshop on Recent Advances in Intrusion Detection, Springer, 2005, pp. 207–226.
[39] S. Lou, S. Cheng, J. Huang, F. Jiang, Tfdroid: android malware detection by topics and sensitive data flows using machine learning techniques, in: Proceedings of the 2019 IEEE 2nd International Conference on Information and Computer Technologies, ICICT, vols. 30–36, 2019. Hawaii, HI, USA.
[40] j. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, H. Ye, Significant permission identification for machine-learning based android malware detection, IEEE Trans. Ind. Inf. 14 (7) (2018) 3216–3225.
[41] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, A latent semantic model with convolutional-pooling structure for information retrieval, in: ACM CIKM, ACM, 2014, pp. 101–110.
[42] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian, M. Conti, Similarity-based Android malware detection using Hamming distance of static binary features, Future Generat. Comput. Syst. 105 (2020) 230–247.
[43] L. Taheri, A.F.A. Kadir, A.H. Lashkari, Extensible android malware detection and family classification using network-flows and API-calls, in: Proceedings of the - International Carnahan Conference on Security Technology, 2019. Chennai, India.
[44] R. Tian, R. Islam, L. Batten, S. Versteeg, Differentiating malware from cleanware using behavioural analysis, in: 2010 5th International Conference on Malicious and Unwanted Software, IEEE, 2010, pp. 23–30.
[45] P. Trinius, C. Willems, T. Holz, K. Rieck, A Malware Instruction Set for Behavior-Based Analysis, None., 2009.
[46] P. Vinod, R. Jaipur, V. Laxmi, M. Gaur, Survey on malware detection methods, in: Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security in 2014 ICACCI, 2337–2342, IEEE. (IITKHACK'09), 2009, pp. 74–79.
[47] O.E. David, N.S. Netanyahu, Deepsign: deep learning for automatic malware signature generation and classification, in: 2015 IJCNN, IEEE, 2015, pp. 1–8, https://doi.org/10.1109/IJCNN.2015.7280815.
[48] J. Feng, L. Shen, Z. Chen, Y. Wang, H. Li, A two-layer deep learning method for android malware detection using network traffic, IEEE Access 8 (2020) 125786, https://doi.org/10.1109/ACCESS.2020.3008081.
[49] Y. Fang, B. Yu, Y. Tang, L. Liu, Z. Lu, Y. Wang, Q. Yang, A new malware classification approach based on malware dynamic analysis, in: Australasian Conference on Information Security and Privacy, Springer, 2017, pp. 173–189, https://doi.org/10.1007/978-3-319-59870-3_10.