ANNDroid: A Framework for Android Malware Detection Using Feature Selection Techniques and Machine Learning Algorithms

Arvind Mahindru

1 Introduction

Android has gained popularity in the year 2011 due to its open-source and number of free apps in its official play store.[1] According to the statistics,[2] more than 2.87 million free apps are present in Google Play Store. Working of android apps depends upon the permissions. At the time of installation, android apps required certain permissions that are required for its proper functioning. On daily basis, cyber-criminals are taking advantage of these permissions and develop malware-infected apps for smartphone users. According to the survey done by Kaspersky Security Network,[3], there are millions of malware-infected apps which are still submitted in Google Play Store and third-party app stores.

According to the report published by Gartner,[4], the growth of smartphone is increased by 11% in the upcoming year. During pandemic, everyone dependent upon apps for their jobs. At the time of installation and run-time, android apps demand certain permissions. Google defined these permissions[5] as "normal" or "dangerous." Normal permissions do not pay any impact on user's privacy. In the reverse, dangerous permissions paid a great effect on user's privacy. The fault lies in the underneath permission model of android apps.

In the literature [12, 14–24], number of authors proposed android malware detection frameworks using supervised and unsupervised machine learning techniques.

[1] https://play.google.com/store.
[2] https://buildfire.com/app-statistics/#.
[3] https://securelist.com/ksb-2020/.
[4] https://indianexpress.com/article/technology/tech-news-technology/smartphone-sales-expected-to-grow-by-11-in-2021-5g-phones-to-play-key-role-7175925/.
[5] https://developer.android.com/guide/topics/permissions/overview.

A. Mahindru (✉)
Department of Computer Science and Applications, D.A.V. University, Jalandhar, India
e-mail: er.arvindmahindru@gmail.com

J. Singh et al. (eds.), *Mobile Application Development: Practice and Experience*,
Studies in Systems, Decision and Control 452,
https://doi.org/10.1007/978-981-19-6893-8_5

Fig. 1 Phases involved in this research article

The main limitation in their work is that researchers and academicians used limited datasets. In order to achieve better detection rate, in this research article, we proposed a framework that is based on the principle of hybrid artificial intelligence techniques approach of functional link artificial neural network (FLANN) with clonal selection algorithm (CSA), particle swarm optimization (PSO) and genetic algorithm (GA), i.e., FLANN-CSA (FCSA), FLANN-PSO (FPSO and MFPSO) and FLANN-genetic (FGA and AFGA). This study also focuses on the effectiveness of feature selection techniques, i.e., principal component analysis (PCA) and rough set analysis (RSA), which are used to reduce the complexity of the proposed model by minimizing the number of inputs.

The generic steps that are followed in this research paper to identify malware-infected apps are shown in Fig. 1. Initially in the first step, we collect Android Application Packages (.apk) files from different repositories. In the second step, we extracted dynamic features and form the features dataset. Implemented of feature selection techniques is performed in the third step. Further, features are selected by implementing feature selection approaches. In the last step, we validate our developed models by using two performance parameters, i.e., accuracy and F-measure.

The unique and novel contributions of this study are as follows:

- To build efficient and effective malware detection model, in this study more than five millions android apps are utilized.
- Dynamic analysis was performed on collected android apps, and 1844 unique features are extracted.
- In this chapter, five different hybrid functional link artificial neural networks are proposed.

The rest of the chapter is summarized as follows. In Sect. 2, related work is discussed. Collection of .apk file and formulation of feature dataset is discussed in Sect. 3. Implemented feature selection techniques are discussed in Sect. 4. Section 5 discusses the proposed hybrid machine learning algorithms. Experimental setup to proposed the framework is discussed in Sect. 6. Outcome of the experiment is discussed in Sect. 7. At last, chapter is concluded in Sect. 8.

2 Related Work

Hou et al. [7] proposed a malware detection framework named as "Droiddelver" based on Application Programming Interface (API) that is extracted from smali files. Proposed model was build by using 5000 different android apps and a deep belief network as a machine learning technique. Empirical outcome reveals that the proposed model was able to detect 96.66% of malware-infected apps. Hou et al. [6] proposed a malware detection model named as "Deep4MalDroid" developed on the basis of dynamic analysis approach called component traversal which follows code routines of particular android apps. Based on the extracted features, they construct the weighted directed graphs and then applied deep learning as a machine learning algorithm. An experiment was performed by using 3000 android apps and detect 91.4% malware-infected apps.

Mahindru and Singh [25] proposed dynamic analysis-based approach that are build by using 123 features. An experiment was performed by using 11,000 distinct android apps and five different machine learning algorithms. The malware detection model developed by using simple logistic achieved a higher detection rate as compared to others. Hou et al. [8] developed a framework entitled as "HinDroid" based on the relationships between API calls and developed higher-level semantics that require more efforts for attackers. An experiment was performed by using two different datasets; i.e., one contains 1834 distinct android apps, and the second contains 30,000 distinct android apps. Proposed malware detection framework was able to identify 99.01% malware-infected apps. Martín et al. [26] developed a model named as "MOCDroid," that is based on the integrity of genetic algorithm. An experiment was performed by using 17,135 android apps and achieved an accuracy of 94.60%. Tong and Yan [30] proposed a hybrid approach that works on the combination of static and dynamic features. Experiment was carry-out by utilizing 2000 different android apps while considering API calls as a feature. Proposed malware detection model achieved the detection rate of 90.19%.

Karbab et al. [10] developed malware detection model named as "MalDozer" that is based on the principle of deep learning techniques. Developed model uses the behavior of API calls to recognize the behavior of benign and malware apps. The developed framework was tested on 38 K benign apps and 33 K malware-infected apps and achieved an $F1$-score of 96–99%. Cai et al. [4] proposed a dynamic malware detection approach that used calls and inter-component communication as features. An experiment was performed by using 34,343 android apps and the proposed framework achieved an accuracy of 97%. Kim et al. [11] developed a malware detection model on the basis of multimodal deep learning. Features were extracted from the manifest file, dex file and shared libraries for developing the model. The developed model was tested with 41,260 android apps and achieved an accuracy of 98%. Yerima et al. [34] proposed detection model entitled as "DroidFusion," that is based on the principle of feature selection techniques and implement multiple machine learning algorithms. The proposed malware detection model was tested with 55,018 distinct smartphone apps and achieved the detection rate of 97%. Shen et al. [28] developed a malware detection model that works on the principle of information flow

analysis. Developed model is based on the structure of information flows to know the pattern behavior and which helps in distinguishing between benign and malware app. An experiment was performed by using 8598 android apps and achieved an accuracy of 82%.

Arora et al. [1] developed malware detection framework work on graphs that construct by utilizing permissions extract from distinct android apps. An experiment was performed by using 5993 android apps and achieved the detection rate of 95.44%. Xiao et al. [32] developed a model by using deep learning principles. The proposed model is built by using system call sequences and long short-term memory as a machine learning technique. An experiment was performed by using 7103 android apps and achieved an accuracy of 96.6%. Mahindru and Sangal [14] developed a malware detection framework entitled as "DeepDroid" by using significant features selected by feature selection approaches and deep learning as a machine learning technique. Experimental outcome reveals that the framework build by using principal component analysis (PCA) as a feature selection technique achieved a higher detection rate as compared to other techniques. Kumar et al. [12] build a detection framework by utilizing three different data sampling approaches, three different feature selection approaches and seven distinct classifier approaches. Outcome reveals that the framework developed by using upscale sampling technique and ELM with polynomial kernel achieved a higher detection rate as compared to others.

Mahindru and Sangal [16] developed a malware detection framework work on the basis of semi-supervised machine learning techniques. The proposed framework is developed by using four different feature subset selection approaches and LLGC as a machine learning algorithm. The empirical result reveals that framework build using rough set analysis as a feature selection approach achieved the detection rate of 97.8%. Mahindru and Sangal [17] developed malware detection model entitled as "GADroid" that is build by using genetic algorithm as a feature selection approach. Further, selected features are used to build the model by using deep learning as machine learning technique. Experiment was performed on 560,142 distinct android apps, and the developed model is able to achieved an accuracy of 98.6%.

Mahindru and Sangal [19] developed the model named as "PARUDroid." Proposed model is able to detect 98.8% malware-infected apps. Table 1 describes the frameworks developed in the literature. Previous malware detection model has been proposed with a limited dataset and conquered a higher accuracy with the limited dataset. On the basis of related work, the following questions have been answered in this research article:

RQ1. To identify which malware detection model is more effective in detecting malware from real-world apps?

This question helps in identifying the malware detection model which is more effective in detecting malware from real-world apps. To answer this question, in this study distinct malware detection models are developed and compared with two different performance parameters, i.e., F-measure and accuracy.

RQ2. Is the proposed malware detection framework able to identify malware from android devices or not?

Table 1 Malware detection frameworks that are availables in the literature

Framework	Machine learning algorithm implemented	Dataset used
Droiddelver [7]	Deep neural network	6000
Deep4MalDroid [6]	Deep neural network	3000
HinDroid [8]	Heterogeneous information network	31,834
MalDozer [10]	Deep neural network	71 K
DeepDroid [14]	Deep neural network	120,000
GADroid [17]	Deep neural network	560,142
PARUDroid [19]	Deep neural network, decision tree Adaboost, Naïve Bayes and random forest	560,142
DLDroid [15]	Deep neural network	11,000
PerbDroid [20]	SVM, Naïve Bayes, random forest, multiple layer perceptron, logistic regression, Bayesian network, Adaboost, decision tree, *K*NN and deep neural network	200,000
MLDroid [18]	SVM, Naïve Bayes, random forest, logistic regression, multiple layer perceptron, *k*-nearest neighbors, Adaboost, self-organizing map, Bayesian network, deep neural network, decision tree, *K*-mean, density-based clustering, filtered clustering, farthest first clustering, MLP + YATSI, J48 + YATSI, SMO + YATSI, best training ensemble approach, majority voting ensemble approach and nonlinear ensemble decision tree forest approach	
SemiDroid [21]	Farthest first clustering, *K*-mean, self-organizing map, filtered clustering, density-based clustering,	550,000
SOMDroid [22]	Self-organizing map	500,000
FSDroid [23]	LSSVM with linear polynomial and radial kernel	200,000
HybriDroid [24]	Best training ensemble majority voting ensemble and nonlinear ensemble decision tree forest	194,659 benign apps and 67,538 malware apps

To examine this question, in this study, proposed framework is compared with existing malware detection models presented in the literature.

RQ3. While selected features using feature selection approaches paid any impact on malware detection models or not?

To answer this question, developed using in this research article model developed using all extracted features compared with the models developed by using feature selection techniques.

3 Collection of *.apk* Files and Formulation of Features Dataset

Collection of five million distinct android apps is performed to use in this research article. Benign .apk files are collected from, i.e., slideme,[6] mumayi,[7] hiapk,[8] appchina,[9] Google's play store,[10] Android,[11] gfan,[12] and pandaapp,[13] and malware-infected apps are collected from Android Malware Genome project [35], 1929, botnet samples were collected from [9] and from AndroMalShare[14] along with their package names. Table 2 represents the distinct categories of android apps with respect to its numbers. Dynamic analysis was performed by using the principle mentioned in [19]. After that, we divided the extracted features into different categories to which they belong. Formulation of feature dataset is mentioned in Table 3.

4 Feature Selection Techniques

Relevant features paid an important role while developing the malware detection models in case of effectiveness and efficiency. In this research article, to select relevant features two different feature selection approaches are considered, i.e., principal component analysis (PCA) and rough set theory.

4.1 Principal Component Analysis (PCA)

To carry-out a data space, low dimension PCA is considered as feature selection. Figure 2 demonstrates the steps that are considered while selecting features using PCA.

[6] http://slideme.org/.
[7] http://www.mumayi.com/.
[8] http://apk.hiapk.com/.
[9] http://www.appchina.com/.
[10] https://play.google.com/store?hl=en.
[11] http://android.d.cn/.
[12] http://apk.gfan.com/.
[13] http://download.pandaapp.com/?app=soft&controller=android#.V-p3f4h97IU.
[14] http://202.117.54.231:8080/.

Table 2 Collected android application packages (*.apk*)

ID	Category	Normal	Trojan	Backdoor	Worms	Botnet	Spyware
DS1	Arcade and action (AA)	15,291	13,300	5000	1004	3300	5000
DS2	Books and reference (BR)	16,235	8000	6500	4060	5650	1600
DS3	Brain and puzzle (BP)	13,928	10,820	204	2008	5010	5010
DS4	Business (BU)	18,208	1420	1150	3250	1842	1602
DS5	Cards and casino (CC)	12,786	7610	6520	8002	5010	4220
DS6	Casual (CA)	16,000	8270	8080	6052	7840	5840
DS7	Comics (CO)	20,967	6050	9950	9900	9900	6100
DS8	Communication (COM)	76,309	8503	5007	8904	8791	8020
DS9	Education (ED)	38,764	8610	8121	8980	8219	8021
DS10	Entertainment (EN)	23,988	8100	8100	7000	1870	5397
DS11	Finance (FI)	23,099	8990	7609	9199	6985	9012
DS12	Health and fitness (HF)	18,661	9181	6852	4825	1840	1940
DS13	Libraries and demo (LD)	13,755	1479	1989	1300	6291	6900
DS14	Lifestyle (LS)	19,650	1855	9805	1808	1093	5082
DS15	Media and video (MV)	18,119	7807	7023	8662	2450	6971
DS16	Medical (ME)	36,000	1128	1983	2344	2884	4805
Ds17	Music and audio (MA)	27,057	6935	5900	6125	1165	2665
DS18	News and magazines (NM)	28,164	4500	3100	2100	1100	1032
DS19	Personalization (PE)	14,334	1580	1042	2590	4280	2170
DS20	Photography (PH)	19,033	3109	4190	2850	9161	5200
DS21	Productivity (PR)	19,750	3600	8903	4350	3290	2972
DS22	Racing (RA)	23,766	1458	3109	4219	8190	2189
DS23	Shopping (SH)	14,673	3120	1950	3120	3150	1959
DS24	Social (SO)	36,159	3190	4550	1210	5159	7159
DS25	Sports (SP)	32,669	6100	7249	9180	4490	8022
DS26	Sports games (SG)	31,889	9200	8045	8125	8250	9198
DS27	Tools (TO)	25,646	9720	8844	7259	9205	4763
DS28	Transportation (TR)	23,796	3102	4200	9100	8002	4120
DS29	Travel and local (TL)	38,180	9508	8220	7050	1248	8100
DS30	Weather (WR)	20,841	7190	2323	9790	3950	2925

4.2 Rough Set Theory

Rough set theory used to eliminate irrelevant features by using approximation, reduced attributes and information method. Steps that are followed in rough set theory are shown in Fig. 3.

Table 3 Formulation of feature datasets

Set number	Description	Set number	Description
FS1	Contain info. Associated to rating and downloads	FS2	Associated to SMS_MMS
FS3	Associated to IMAGE	FS4	Associated to HARDWARE_CONTROLS
FS5	Associated to READ	FS6	Associated to BROWSER_INFORMATION
FS7	Associated to WIDGET	FS8	Associated to SYSTEM_SETTINGS
FS9	Associated to CONTACT_INFORMATION	FS10	Associated to FILE_INFORMATION
FS11	Associated to default group	FS12	Associated to LOCATION_INFORMATION
FS13	Associated to BUNDLE	FS14	Associated to CALENDAR_INFORMATION
FS15	Associated to SYNCHRONIZATION _DATA	FS16	Associated to DATABASE_INFORMATION
FS17	Associated to READ_AND_WRITE	FS18	Associated to UNIQUE_IDENTIFIER
FS19	Associated to LOG_FILE	FS20	Associated to ACCOUNT_SETTINGS
FS21	Associated to PHONE_CALLS	FS22	Associated to ACCESS_ACTION R
FS23	Associated to SERVICES_THAT_COST_YOU_MONEY	FS24	Associated to SYSTEM_TOOLS
FS25	Associated to YOUR_ACCOUNTS	FS26	Associated to NETWORK_INFORMATION and BLUETOOTH_INFORMATION
FS27	Associated to AUDIO and VIDEO	FS28	Associated to PHONE_STATE and PHONE_CONNECTION
FS29	Contain info. Associated to API calls	FS30	Associated to STORAGE_FILE

5 Proposed Hybrid Machine Learning Techniques

In this section, we discuss various machine learning algorithms that are developed by using genetic algorithm, clonal selection and particle swarm optimization for detection malware from android apps.

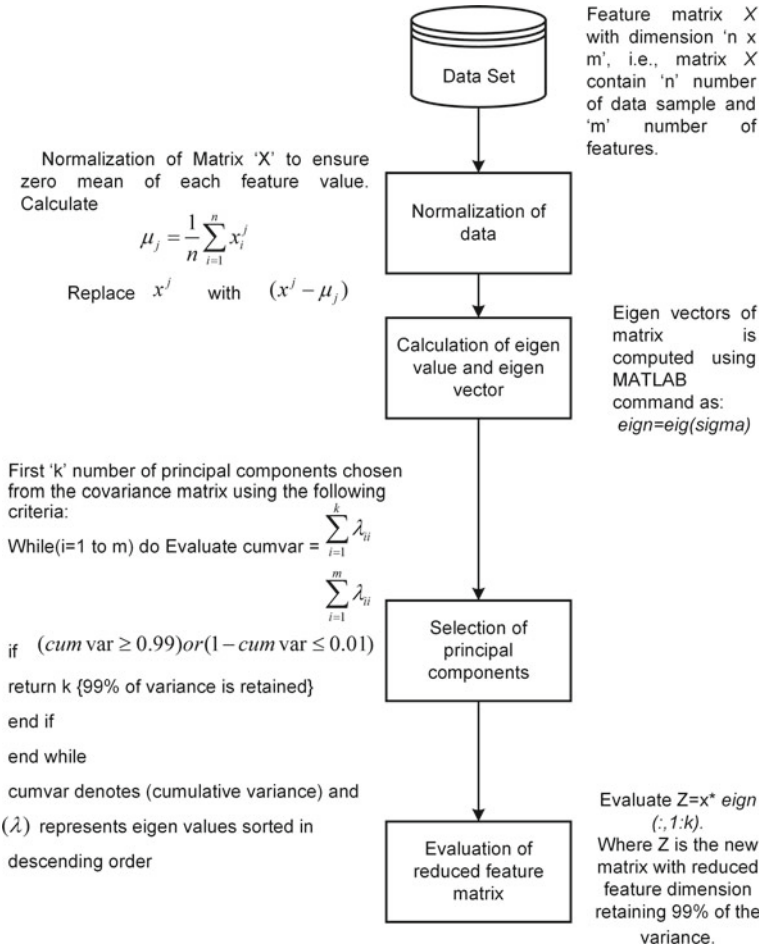


Fig. 2 Framework of PCA calculation

5.1 Functional Link Artificial Neural Network (FLANN)

In this research article, FLANN is implemented to detect malware from android apps. FLANN is worked on the architecture of single layered of artificial neural network (ANN), that is responsible to perform complex decision. The computational cost of ANN is very high, but in the case of FLANN it is very less due to not present of hidden layers. Figure 4 demonstrate the basis architecture of FLANN.

Output is computed by using following equations:

$$\hat{z} = \sum_{i=1}^{n} W_i a_i \tag{1}$$

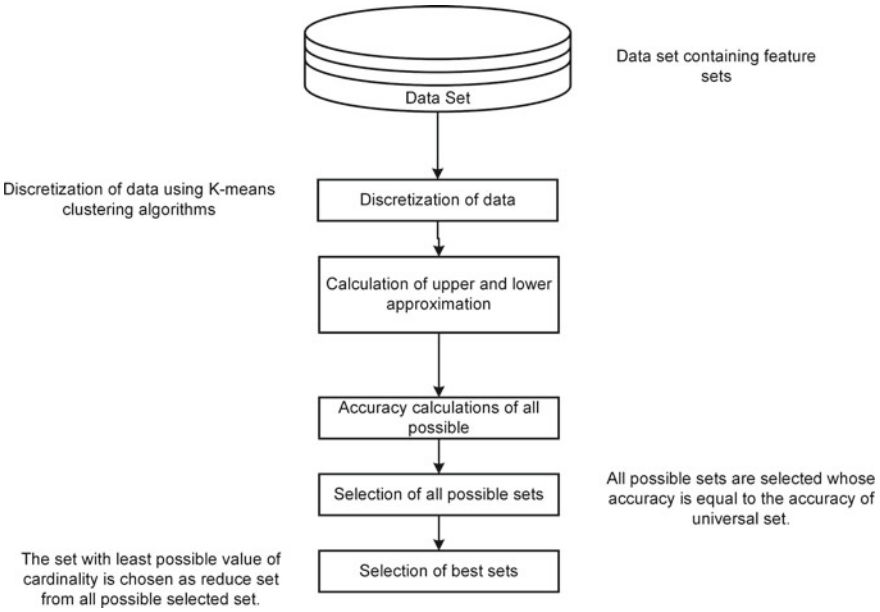


Fig. 3 Rough set theory framework

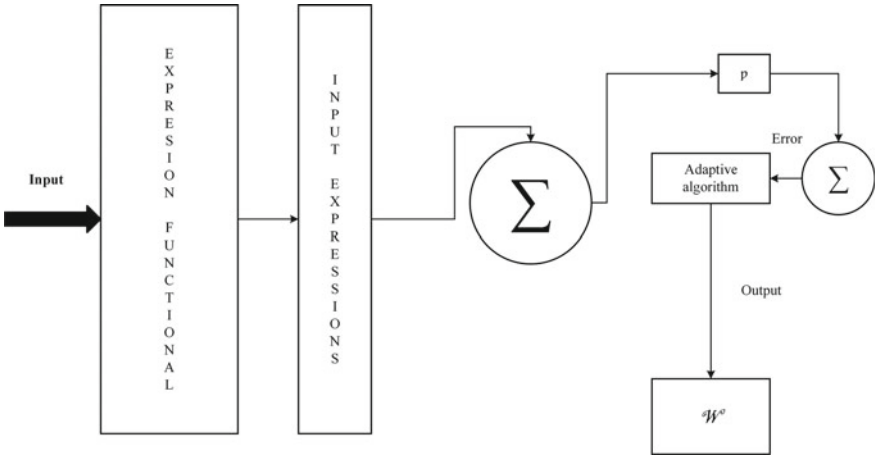


Fig. 4 Architecture of FLANN

where z and $\hat{z}$ are the estimated and actual values, a_i is the function block and W is the weight vector that is defined by using

$$A = [1, a_1, \sin \pi a_1, \cos \pi a_1, a_2, \sin \pi a_2, \cos \pi a_2, \ldots] \tag{2}$$

The revised weight is updated as:

$$W_i(k+1) = W_i(k) + \alpha e_i(k) a_i(k) \tag{3}$$

where e_i is the error value and α is the learning rate that is determined as:

$$e_i = z_i - \hat{z}_i \tag{4}$$

5.2 FLANN-Genetic (FGA) Technique

This technique is very effective at the time of learning, and it is utilized mostly there for upgrading the weight. A function link neural network with a form of '$a - x$' is deemed as estimation; i.e., the network contains l number of input neurons and x number of output neurons.

Weights are calculated using the following equation:

$$W_a = \begin{cases} -\frac{y_{ad+2} * 10^{d-2} + y_{ad+3} * 10^{d-3} + \cdots + y_{(a+1)d}}{10^{d-2}} & \text{if } 0 \leq y_{ad+1} < 5 \\ \frac{y_{ad+2} * 10^{d-2} + y_{ad+3} * 10^{d-3} + \cdots + y_{(a+1)d}}{10^{d-2}} & \text{if } 5 \leq y_{ad+1} \leq 9 \end{cases}$$

5.3 Adaptive FLANN-Genetic (AFGA) Technique

This approach, paid an impact on two different parameters for its advancement, i.e., probability for mutation (P_m) and probability for cross over (P_c). Updated values of $(P_m)_{k+1}$ and $(P_c)_{k+1}$ is calculated by using the following equations:

$$(P_m)_{k+1} = (P_m)_i - \frac{C_2 * n}{5} \tag{5}$$

$$(P_c)_{k+1} = (P_c)_i - \frac{C_1 * n}{5} \tag{6}$$

5.4 FLANN Particle Swarm Optimization (FPSO) Technique

It is based on the principle of particle swarm optimization and Function link neural network. PSO is utilized to update the weight at learning phase. Figure 5 represents the execution of PSO. Formula to calculate the fitness value is:

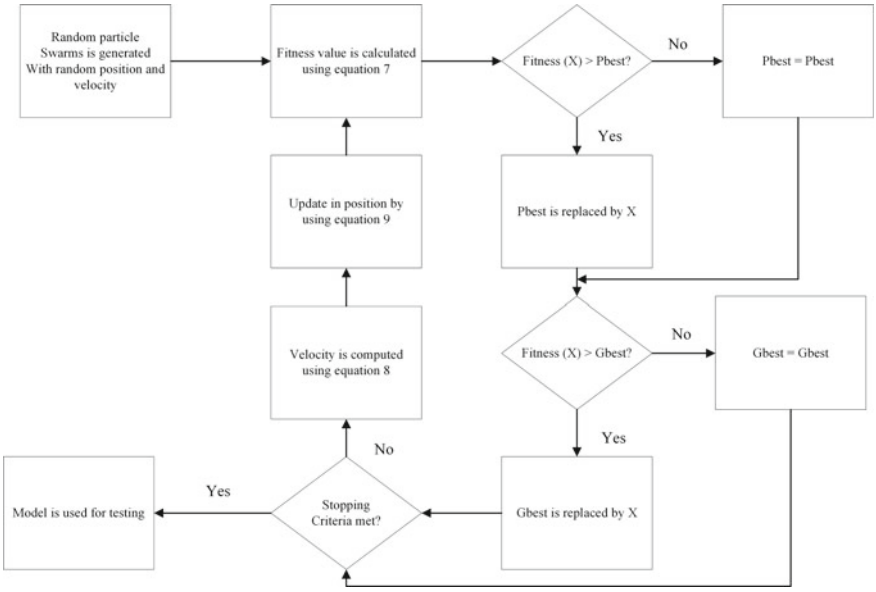


Fig. 5 Flowchart representing PSO execution

$$F_i = 1/E_i \tag{7}$$

$$V_{k+1}^i = V_k^i + C_1 * R_1 * (Pbest_k^i - X_k^i) + C_2 * R_2 * (Gbest_k^n - X_k^i) \tag{8}$$

$$X_{k+1}^i = X_k^i + V_{k+1}^i \tag{9}$$

where X is the position of particles and V is the velocity.

5.5 FLANN-Clonal Selection Algorithm (FCSA) Approach

FCSA is a hybrid approach using clonal selection algorithm and functional link neural network [13].

5.6 Modified-FLANN Particle Swarm Optimization (MFPSO) Technique

The main difference between PSO and MFPSO approach is that in case of MFPSO mutation stage is included just the completion of first stage. The following equation is required to calculate the update value of mutation.

$$(P_m)_{k+1} = (P_m)_i - \frac{C * n}{10} \tag{10}$$

where P_m is the first state of mutation and n is the generation number.

6 Experimental Setup

In this section of the chapter, we discuss the experimental setup done to find that developed malware detection model is effective or not. Six different hybrid functional link artificial neural network machine learning algorithms are implemented in this chapter. In Fig. 6, representation of proposed framework is demonstrated. In the first phase, feature selection techniques are implemented, i.e., PCA and rough set theory to select significant features. In the second phase, to normalize the features min-max approach is implemented. Distinct malware detections are developed by using six different machine learning techniques. After that, confusion matrix is developed by using the technique mentioned in [23, 24]. By comparing the malware detection model, best suitable model is selected and compared with the existing framework mentioned in the literature. If the detection rate is high after comparing the models with the existing framework, then proposed framework is useful or vice versa.

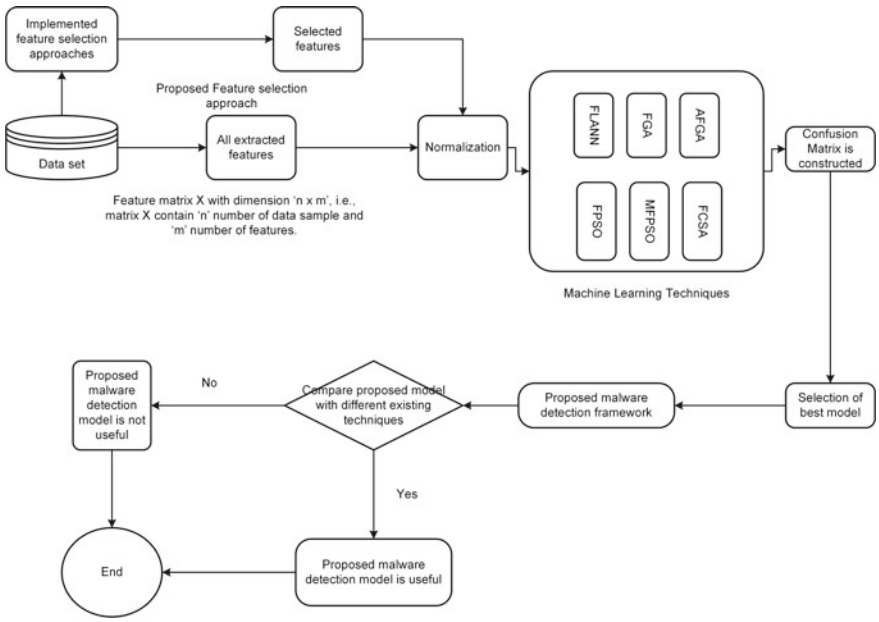
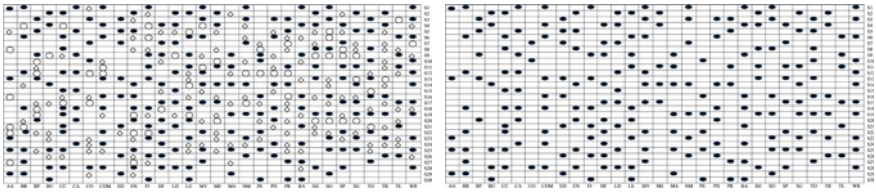


Fig. 6 Proposed framework, i.e., ANNDroid



(a) PCA (b) RSA

Fig. 7 Feature selected using PCA and rough set analysis

7 Outcomes

In this section, the outcomes are gained by performing feature selection approaches and machine learning algorithms.

7.1 Feature Selection Approaches

Relevant features are selected using PCA whose eigenvalue is greater than 1, and features selected using rough set analysis are basis on heuristic search. Features selected using PCA and rough set analysis are demonstrated in Fig. 7.

7.2 Machine Learning Approaches

Tables 4 and 5 represent the measured value of accuracy and F-measure using PCA and rough set analysis using the equations mentioned in the literature [18, 19]. From tables, it may be inferred that:

- Highest detection rate is represented by bold value.
- It is observed from tables that models developed using features selection techniques achieved higher detection rate as compared to all extracted feature set.
- Model developed using FLANN-genetic accomplished higher detection rate as resembled to FLANN-PSO and FLANN-CSA.

In order to search, developed malware detection model is effective or not, box-plot diagrams of the individual developed model is constructed. Figures 8 and 9 demonstrate the box-plot diagrams for accuracy and F-measure using feature selection approaches. From figures, it can be concluded that:

- Based on Figs. 8 and 9, model developed by using RSA as feature selection technique achieve higher detection rate.

Table 4 Measured accuracy and *F*-measure using PCA

Accuracy								*F*-measure						
ID	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA
DS1	62.33	73.33	75.0	77.37	77.66	**78.4**	0.68	**0.79**	0.77	0.76	**0.79**	0.72	0.71	**0.79**
DS2	68.18	72.66	78	76	73	76.6	**83**	0.63	0.73	0.72	0.76	0.77	0.76	**0.82**
DS3	70.8	84	83	82	82.6	81.6	**89.7**	0.72	0.84	0.83	0.82	0.84	0.83	**0.88**
DS4	62.8	76	75	72	75.6	79.6	**82**	0.68	0.70	0.72	0.79	0.78	0.77	**0.80**
DS5	70	82	85	81	86	87	**89**	0.67	0.86	0.84	0.82	0.84	0.81	**0.89**
DS6	68.8	78	76	81	86.6	85.6	**88**	0.71	0.72	0.76	0.80	0.82	0.83	**0.89**
DS7	67.8	87	87	85	80	88	**89**	0.71	0.84	0.82	0.86	0.88	0.89	**0.90**
DS8	67	83	84	85	88	87	**90**	0.71	0.82	0.82	0.83	0.83	0.84	**0.87**
DS9	78	90	90.8	**93**	91	91	92	0.78	0.90	0.92	**0.95**	0.90	0.90	0.94
DS10	66.8	87	86	85	89	88	**90**	0.68	0.77	0.75	0.82	0.84	0.83	**0.89**
DS11	69	88.7	88	81	85	87	**89**	0.68	0.84	0.83	0.83	0.84	0.83	**0.90**
DS12	70.8	87	85	88	82	85	**89**	0.71	0.84	0.83	0.86	0.86	0.88	**0.89**
DS13	71	86.7	87	88	88.2	88.1	**89.8**	0.67	0.81	0.85	0.83	0.84	0.83	**0.89**
DS14	77.3	85	87	86	87.6	87.3	**89.7**	0.71	0.88	0.83	0.84	0.83	0.86	**0.88**
DS15	70	89	89.9	89.8	91.8	90.7	**92.4**	0.73	0.84	0.85	0.88	0.88	0.89	**0.91**
DS16	67	83	82	80	84	85	**88**	0.70	0.86	0.84	0.85	0.84	0.83	**0.88**
DS17	79	87.7	88	88.4	87	86.8	**89.9**	0.76	0.85	0.86	0.88	0.86	0.88	**0.89**
DS18	79	88	87	88	86	89	**89.9**	0.69	0.83	0.85	0.86	0.86	0.86	**0.88**
DS19	68.9	82	83	84.8	85.8	86.9	**88**	0.72	0.84	0.81	0.82	0.80	0.82	**0.87**
DS20	68	80	81.9	88.3	**89.6**	81	89	0.68	0.78	0.79	0.80	**0.81**	0.80	0.79
DS21	67	87.8	88.7	89.8	90	89.7	**90.7**	0.74	0.84	0.84	0.83	0.81	0.80	**0.85**
DS22	70	86	81	82	84	85	**88**	0.70	0.83	0.84	0.83	0.84	0.85	**0.89**

(continued)

Table 4 (continued)

Accuracy												*F*-measure		
ID	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA
DS23	68.9	88.78	83.71	83.9	**89**	84	88	0.66	0.81	**0.87**	0.83	0.82	0.80	0.81
DS24	65	88	88.2	**89.3**	89	87.7	82	0.70	0.84	0.81	**0.87**	0.82	0.85	0.85
DS25	78	88	89	87	89	84	**89.8**	0.71	0.85	0.83	0.82	0.81	0.80	**0.88**
DS26	66	80.8	82.1	84	85	85	**88**	0.72	0.85	0.84	0.85	0.86	0.86	**0.87**
DS27	67	88	81.9	89.6	87	86.9	**89**	0.67	0.81	0.82	0.83	0.84	0.82	**0.88**
DS28	67	89	88	**89.8**	86	81	88	0.71	0.86	0.84	0.81	0.86	0.85	**0.88**
DS29	67	81	87	**88.9**	85	84	81.9	0.67	0.87	0.86	0.87	**0.88**	0.81	0.82
DS30	60	84	89	89	88	81	**89.8**	0.67	0.84	0.85	0.86	0.83	0.81	**0.88**

AF stands for all extracted features

Table 5 Measured accuracy and *F*-measure using rough set analysis

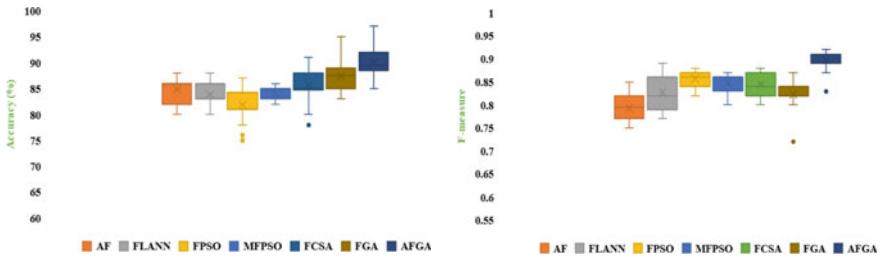
Accuracy												*F*-measure		
ID	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA
DS1	68.33	82	83	85	86	83	**89.8**	0.79	0.83	0.85	0.82	0.87	0.81	**0.89**
DS2	65	85	82	85	86	89	**91.8**	0.75	0.81	0.85	0.83	0.85	0.81	**0.87**
DS3	67	85	83	81	84	89	**90.8**	0.78	0.85	0.86	0.85	0.84	0.87	**0.89**
DS4	62.8	83	89	85	87	89	**90.7**	0.72	0.86	0.86	0.87	0.81	0.86	**0.88**
DS5	68.8	86	87	82	83	85	**89.8**	0.67	0.83	0.84	0.85	0.86	0.87	**0.90**
DS6	67.9	84	88	89	92	94	**96.7**	0.69	0.87	0.85	0.88	0.87	0.88	**0.90**
DS7	78	89.6	88.7	86	86.8	89.7	**93.8**	0.70	0.89	0.86	0.87	0.82	0.81	**0.89**
DS8	65	84	85	86	87	88	**91**	0.67	0.84	0.83	0.84	0.84	0.88	**0.89**
DS9	68	83	84	**96**	95	93	86	0.78	0.92	0.96	**0.99**	0.91	0.92	0.91
DS10	66.8	82	89	89	89.8	89.7	**97**	0.70	0.87	0.85	0.88	0.82	0.88	**0.96**
DS11	79	89	89	80	86	88	**98**	0.72	0.87	0.85	0.84	0.82	0.85	**0.93**
DS12	66.8	81	83	88	87	89	**90**	0.75	0.81	0.82	0.86	0.86	0.84	**0.89**
DS13	69.1	82	87	82	86	88	**89.8**	0.60	0.79	0.81	0.82	0.80	0.81	**0.88**
DS14	67	85	82	81	86	86	**90.9**	0.67	0.88	0.83	0.85	0.81	0.82	**0.89**
DS15	70.7	88	88	89.8	91	92	**97**	0.69	0.85	0.83	0.84	0.86	0.86	**0.94**
DS16	67	82	83	81	86	87	**88**	0.72	0.81	0.83	0.80	0.80	0.81	**0.83**
DS17	80	90	92	93	96	91	**98**	0.67	0.88	0.82	0.85	0.88	0.98	**1**
DS18	72	92.8	91	92.9	95	96	**98.9**	0.70	0.84	0.85	0.86	0.88	0.90	**0.93**
DS19	77	92	93	92	95	96	**97**	0.72	0.92	0.91	0.92	0.90	0.88	**0.95**
DS20	68	90	92	93	**95**	91	92	0.78	0.85	0.87	0.88	**0.89**	0.84	0.85
DS21	62	80	80.7	82	83	84	**85.7**	0.67	0.87	0.88	0.88	0.87	0.89	**0.9**

(continued)

Table 5 (continued)

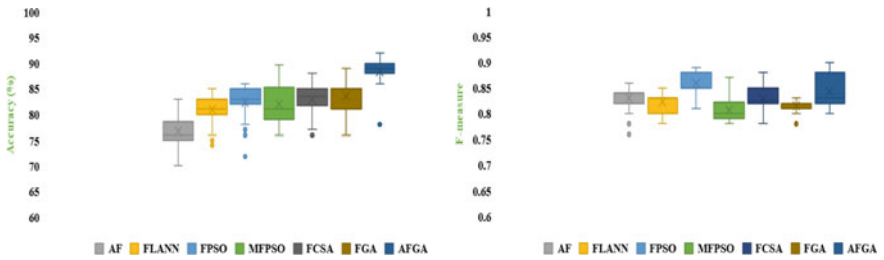
Accuracy												*F*-measure		
ID	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA	AF	FLANN	FCSA	MFPSO	FPSO	AFGA	FGA
DS22	69.8	88	90	92	96	95	**98**	0.68	0.82	0.85	0.85	0.88	0.89	**0.91**
DS23	68.9	87.78	87.71	87.9	**91**	90	90.1	0.68	0.82	**0.85**	0.83	0.82	0.80	0.82
DS24	65	89	89.2	**91.3**	90	89.7	88	0.67	0.85	0.82	**0.89**	0.82	0.85	0.85
DS25	69.9	97	92	93	96	97	**98.8**	0.77	0.88	0.89	0.89	0.89	0.88	**1**
DS26	69.9	94.8	96.1	94	97	95	**97.9**	0.73	0.86	0.86	0.85	0.88	0.86	**0.92**
DS27	67	90.1	91.9	93.6	97	95.8	**97.9**	0.71	0.89	0.87	0.85	0.86	0.87	**0.91**
DS28	63	92	91	**98**	86	81	91	0.78	0.85	0.86	**0.89**	0.88	0.85	0.88
DS29	67	82	87	**92**	85	84	89	0.77	0.88	0.87	0.86	**0.87**	0.85	0.88
DS30	60	89	91	92	95	91	**98**	0.67	0.86	0.87	0.85	0.87	0.87	**1**

AF stands for all extracted features



(a) Accuracy (b) F-Measure

Fig. 8 Box-plot diagram of accuracy and F-measure using PCA



(a) Accuracy (b) F-Measure

Fig. 9 Box-plot diagram of accuracy and F-measure using RSA

- On the basis of Fig. 9, it is seen that model developed by using FLANN-genetic is having higher median value and few outliers. Model build by using RSA achieved higher detection rate as compared to PCA.

7.3 Comparison with Existing Developed Frameworks

In order to find out developed malware detection model is effective in detecting malware or not, in this chapter comparison is done by using existing frameworks present in the literature. To perform this experiment, freely available dataset; i.e., Drebin [2] is considered. Table 6 represent the comparison with existing approaches or frameworks presented in the literature.

7.3.1 Experimental Findings

In this chapter, a framework is developed by using android apps and by utilizing hybrid artificial neural network. Based on the outcome, this study is able to answer the questions discussed in Sect. 2.

Table 6 Comparison of developed model with available frameworks present in the literature

Framework/ approach	Purpose	Approach	Deployment	Data set	Accuracy
Paranoid Android [27]	Detection	Dynamic and behavioral	Off-device	Limited	–
Crowdroid [3]	Detection	Dynamic, system call/API and behavioral	Distributed	Very-limited	High
Aurasium [33]	Detection	Dynamic and behavioral	Off-device	Limited	High
CopperDroid [29]	Analysis and detection	Dynamic, system/API and VMI	Off-device	Limited	Moderate
TaintDroid [5]	Detection	Run-time system call/API and behavioral	Off-device	Very-limited	Moderate
HinDroid [8]	Detection	Dynamic and API	Off-device	Limited	Moderate
Mahindru and Singh [25]	Detection	Run-time	Off-device	Limited	Moderate
MalDozer [10]	Detection	Run-time	Off-device	Limited	Moderate
DroidDet[36]	Detection	Static	Off-device	Limited	Moderate
Wei Wang[31]	Detection	Run-time	Off-device	Limited	Moderate
DeepDroid [14]	Detection	Run-time	Off-device	Limited	Moderate
PerbDroid [20]	Detection	Run-time	Off-device	Limited	High
Mahindru and Sangal [16]	Detection	Run-time	Off-device	Limited	High
ANNDroid (our proposed framework)	Detection	Run-time, permissions, API calls, user-rating and number of user download app	Off-device	Unlimited	Higher

RQ1: In the present study, implementation of six different machine learning techniques is used to develop malware detection model. Based on Tables 4 and 5, it can be implicit that model build using FLANN-genetic is more effective in detecting malware-infected from android.

RQ2: Yes, proposed detection model is effective in identifying malware-infected apps when compared to existing frameworks present in the literature.

RQ3: From Tables 4 and 5, it can be concluded that feature selection techniques have a significant role in building the malware detection model. Models developed using feature selection techniques are very effective when compared to the model developed using all extracted features.

8 Conclusion

This chapter paid a significant role while developing the malware detection models by using distinct android apps. In addition to that, it is observed that feature selection approach also paid an significant role while selecting the relevant features from all extracted features. Moreover, model developed using hybrid approach is more capable in detecting malware as compared to previously developed frameworks.

References

1. Arora, A., Peddoju, S.K., Conti, M.: Permpair: android malware detection using permission pairs. IEEE Trans. Inf. Forensics Secur. **15**, 1968–1982 (2019)
2. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C.: Drebin: effective and explainable detection of android malware in your pocket. NDSS **14**, 23–26 (2014)
3. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15–26 (2011)
4. Cai, H., Meng, N., Ryder, B., Yao, D.: Droidcat: effective android malware detection and categorization via app-level profiling. IEEE Trans. Inf. Forensics Secur. **14**(6), 1455–1470 (2018)
5. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst. (TOCS) **32**(2), 1–29 (2014)
6. Hou, S., Saas, A., Chen, L., Ye, Y.: Deep4maldroid: a deep learning framework for android malware detection based on linux kernel system call graphs. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), pp. 104–111. IEEE (2016)
7. Hou, S., Saas, A., Ye, Y., Chen, L.: Droiddelver: an android malware detection system using deep belief network based on api call blocks. In: International Conference on Web-Age Information Management, pp. 54–66. Springer (2016)
8. Hou, S., Ye, Y., Song, Y., Abdulhayoglu, M.: Hindroid: an intelligent android malware detection system based on structured heterogeneous information network. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1507–1515 (2017)

9. Kadir, A.F.A., Stakhanova, N., Ghorbani, A.A.: Android botnets: What URLs are telling us. In: International Conference on Network and System Security, pp. 78–91. Springer (2015)
10. Karbab, E.B., Debbabi, M., Derhab, A., Mouheb, D.: Maldozer: automatic framework for android malware detection using deep learning. Digit. Invest. **24**, S48–S59 (2018)
11. Kim, T., Kang, B., Rho, M., Sezer, S., Im, E.G.: A multimodal deep learning method for android malware detection using various features. IEEE Trans. Inf. Forensics Secur. **14**(3), 773–788 (2018)
12. Kumar, L., Hota, C., Mahindru, A., Neti, L.B.M.: Android malware prediction using extreme learning machine with different kernel functions. In: Proceedings of the Asian Internet Engineering Conference, pp. 33–40 (2019)
13. Kumar, L., Rath, S.K.: Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software. J. Syst. Softw. **121**, 170–190 (2016)
14. Mahindru, A., Sangal, A.: Deepdroid: feature selection approach to detect android malware using deep learning. In: 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), pp. 16–19. IEEE (2019)
15. Mahindru, A., Sangal, A.: Dldroid: feature selection based malware detection framework for android apps developed during covid-19. Int. J. Emerg. Technol. **11**(3), 516–525 (2020)
16. Mahindru, A., Sangal, A.: Feature-based semi-supervised learning to detect malware from android. In: Automated Software Engineering: A Deep Learning-Based Approach, pp. 93–118. Springer (2020)
17. Mahindru, A., Sangal, A.: Gadroid: a framework for malware detection from android by using genetic algorithm as feature selection approach. Int. J. Adv. Sci. Technol. **29**(5), 5532–5543 (2020)
18. Mahindru, A., Sangal, A.: Mldroid-framework for android malware detection using machine learning techniques. Neural Comput. Appl., 1–58 (2020)
19. Mahindru, A., Sangal, A.: Parudroid: validation of android malware detection dataset. J. Cybersecur. Inform. Manag. **3**(2), 42–52 (2020)
20. Mahindru, A., Sangal, A.: Perbdroid: effective malware detection model developed using machine learning classification techniques. In: A Journey Towards Bio-Inspired Techniques in Software Engineering, pp. 103–139. Springer (2020)
21. Mahindru, A., Sangal, A.: Semidroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches. Int. J. Mach. Learn. Cybernet., 1–43 (2020)
22. Mahindru, A., Sangal, A.: Somdroid: android malware detection by artificial neural network trained using unsupervised learning. Evol. Intell., 1–31 (2020)
23. Mahindru, A., Sangal, A.: Fsdroid:-a feature selection technique to detect malware from android using machine learning techniques. Multimedia Tools Appl., 1–53 (2021)
24. Mahindru, A., Sangal, A.: Hybridroid: an empirical analysis on effective malware detection model developed using ensemble methods. J. Supercomput., 1–43 (2021)
25. Mahindru, A., Singh, P.: Dynamic permissions based android malware detection using machine learning techniques. In: Proceedings of the 10th Innovations in Software Engineering Conference, pp. 202–210 (2017)
26. Martín, A., Menéndez, H.D., Camacho, D.: Mocdroid: multi-objective evolutionary classifier for android malware detection. Soft Comput. **21**(24), 7405–7415 (2017)
27. Portokalidis, G., Homburg, P., Anagnostakis, K., Bos, H.: Paranoid android: versatile protection for smartphones. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 347–356 (2010)
28. Shen, F., Del Vecchio, J., Mohaisen, A., Ko, S.Y., Ziarek, L.: Android malware detection using complex-flows. IEEE Trans. Mob. Comput. **18**(6), 1231–1245 (2018)
29. Tam, K., Khan, S.J., Fattori, A., Cavallaro, L.: Copperdroid: Automatic reconstruction of android malware behaviors. In: NDSS (2015)
30. Tong, F., Yan, Z.: A hybrid approach of mobile malware detection in android. J. Parallel Distrib. Comput. **103**, 22–31 (2017)

31. Wang, W., Zhao, M., Wang, J.: Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. J. Ambient Intell. Hum. Comput. **10**(8), 3035–3043 (2019)
32. Xiao, X., Zhang, S., Mercaldo, F., Hu, G., Sangaiah, A.K.: Android malware detection based on system call sequences and LSTM. Multimedia Tools Appl. **78**(4), 3979–3999 (2019)
33. Xu, R., Saïdi, H., Anderson, R.: Aurasium: Practical policy enforcement for android applications. In: Presented as Part of the 21st USENIX Security Symposium (USENIX Security 12), pp. 539–552 (2012)
34. Yerima, S.Y., Sezer, S.: Droidfusion: a novel multilevel classifier fusion approach for android malware detection. IEEE Trans. Cybernet. **49**(2), 453–466 (2018)
35. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: 2012 IEEE Symposium on Security and Privacy, pp. 95–109. IEEE (2012)
36. Zhu, H.J., You, Z.H., Zhu, Z.X., Shi, W.L., Chen, X., Cheng, L.: Droiddet: effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing **272**, 638–646 (2018)