



# A Systematic Overview of Android Malware Detection

Li Meijin, Fang Zhiyang, Wang Junfeng, Cheng Luyu, Zeng Qi, Yang Tao, Wu Yinwei & Geng Jiaxuan

To cite this article: Li Meijin, Fang Zhiyang, Wang Junfeng, Cheng Luyu, Zeng Qi, Yang Tao, Wu Yinwei & Geng Jiaxuan (2022) A Systematic Overview of Android Malware Detection, Applied Artificial Intelligence, 36:1, 2007327, DOI: [10.1080/08839514.2021.2007327](https://doi.org/10.1080/08839514.2021.2007327)

To link to this article: <https://doi.org/10.1080/08839514.2021.2007327>



© 2021 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 14 Dec 2021.



Submit your article to this journal [↗](#)



Article views: 6450



View related articles [↗](#)



View Crossmark data [↗](#)







Citing articles: 5 View citing articles [↗](#)

REVIEW

 OPEN ACCESS

 Check for updates

## A Systematic Overview of Android Malware Detection

Li Meijin <sup>a</sup>, Fang Zhiyang <sup>b</sup>, Wang Junfeng<sup>c</sup>, Cheng Luyu<sup>d</sup>, Zeng Qi <sup>c</sup>,  
Yang Tao <sup>c</sup>, Wu Yinwei<sup>a</sup>, and Geng Jiaxuan<sup>c</sup>

<sup>a</sup>College of Software Engineering, Sichuan University, Chengdu, China; <sup>b</sup>School of Cyber Science and Engineering, Sichuan University, Chengdu, China; <sup>c</sup>College of Computer Science, Sichuan University, Chengdu, China; <sup>d</sup>School of Business, Sichuan University, Chengdu, China

### ABSTRACT

Due to the completely open-source nature of Android, the exploitable vulnerability of malware attacks is increasing. To stay ahead of other similar review work attempting to deal with the serious security problem of the Android environment, this work not only summarizes the approaches in the malware classification phase but also lays emphasis on the Android feature selection algorithm and presents some areas neglected in previous works in the field of Android malware detection, like limitations and commonly applied datasets in machine learning-based models. In this paper, the Android OS environment, feature selection, classification models, and confronted challenges of machine learning detection are described in detail. Based on the brief introduction to Android background knowledge, feature selection methods are elaborated from key perspectives as feature extraction, raw data preprocessing, valid feature subsets selection, and machine learning-based selection models. For the algorithms of the malware classification, machine learning methods are categorized according to different standards to present an all-around view. Furthermore, this paper focuses on the study of deterioration problems and evasion attacks in machine learning detectors.

### ARTICLE HISTORY

Received 5 August 2021  
Revised 6 November 2021  
Accepted 8 November 2021

## Introduction

Due to the rapid development of mobile intelligent terminals, Android becomes the most generally used computing platform on smartphones. As TrendForce (Huang 2020) recently issued, a total of 1.25 billion smartphones were produced in 2020 and Android captured 78.4% of the market shares. However, due to the wide distribution and the open-source nature, Android applications are accessible from potentially malicious third parties besides the official Android Market, which makes the platform a target for malware attacks. According to the 2019 Android Malware Special Report (360 Internet Security Center 2020) released by 360 Security on February 28,

**CONTACT** Fang Zhiyang  [fangzhiyang@scu.edu.cn](mailto:fangzhiyang@scu.edu.cn)  School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China

This article has been republished with minor changes. These changes do not impact the academic content of the article.

© 2021 The Author(s). Published with license by Taylor & Francis Group, LLC.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2020, the platform intercepted about 1.809 million new malware samples on mobile terminals in 2019, and about 5,000 new mobile malware samples were intercepted on an average day.

Reviews for Android malware detection have been proposed in many previous studies. (Liu et al. 2020a) studied machine learning models of Android malware detection in recent years but neglected to see the limitations of machine learning models mentioned. (Bakour, Ünver, and Ghanem 2019) provided a wide taxonomy for all research directions. The overviews of static analysis (Li et al. 2017) and traditional detection (e.g., signature, heuristics based) (Sihwail, Omar and Ariffin 2018) are all included in this paper to draw the conclusion from differential perspectives.

However, to address the limitations of the outdated collected research, the ignorance of the importance of feature selection algorithms and the problems of machine learning models, this paper gives a more all-around overview of the work with a large volume of relative works from 2015 to 2021.

To our best knowledge, few reviews have systematically presented research focused on feature on four feature processing stages and Android malware classification models according to the taxonomy of detection tech and machine learning type. Compared with other researches, this work points out some areas neglected in previous works in the field of Android malware detection like limitations and challenges in machine learning models. Additionally, widely applied Android datasets and disassemble tools are summarized. The correlative research on Android malware collected in this paper can provide valuable reference and broaden the research direction for future researchers.

The main contributions of this paper are summarized as follows.

(1) Compared with other similar work, a comprehensive overview of Android feature selection is provided from more detailed aspects of the feature extraction, raw feature processing, and valid subsets stages. This paper concludes each data preprocessing method in four-stage from data cleaning to data transformation. How to pick features that stand the test of time without frequent retraining of machine learning models in valid feature subsets selection is also discussed.

(2) This survey completes existing previous reviews in offering a systematic overview of detection methods, including not only those based on machine learning but also traditional methods (e.g., signature, heuristics based). For the algorithms of the Android malware classifier, machine learning detectors are elaborated from learning task and learning type. With the introduction of the taxonomy of machine learning methods, the commonly used models in Android malware detection are distinguished as traditional machine learning, currently advanced detection models, and ensemble learning.

(3) This work bridges some research gaps in peer works by presenting the limitations of machine learning models in Android malware detection. Due to the evolution of the Android operating system, the deterioration issue of

machine learning-based detector and the problem of frequent model retraining to select valid feature subset are highlighted. And the vulnerability to security attacks of machine learning detectors is also analyzed.

The paper is structured as follows: [Section 2](#) describes the procedure used for collecting the papers. [Section 3](#) gives a brief introduction to background knowledge of Android, the mechanism of Android operating system, and the key components of Android. In [Section 4](#), the researches that focused on feature selection in recent years are summarized. [Section 5](#) describes different algorithms of the classifier, especially focused on machine learning models. [Section 6](#) takes a further look into the study on evasion attacks and deterioration issues to expose the vulnerability of many machine learning models. [Section 7](#) introduces the widely used Android datasets and disassemble tools for Android applications. Finally, [Section 8](#) makes a conclusion of this paper.

The following research questions have been brought out to help follow the process of systematic review conduction:

**RQ1** What are the significant feature extraction tech and data preprocessing methods in Android feature engineering?

**RQ2** How are valid feature subset selection models categorized? What kind of machine learning models can be applied to feature subsets selection?

**RQ3** How are the most advanced machine learning frameworks applied to the research of Android malware classification in recent years, and what are the limitations?

**RQ4** What are the widely used Android datasets that can be applied to establish longitudinal comprehensive experiments?

## **Method of Literature Collection**

It is common practice to clarify the paper selection criteria for a literature review, to establish representativeness and confidence regarding the source of information. The procedure used for selecting the papers is described as follows.

(1) The collection scope is based on the main content of this paper, so the search scope is composed of two main parts as Android malware classification and valid feature subsets selection, including the adversarial attack and degradation problems in each stage. To find other relevant papers that may not be collected through the keyword, an incomplete reverse snowball search is carried on the reference list of the article determined by early keyword search.

(2) In each section, the keywords are determined by the categories. For example, since malware analysis can be categorized into static/dynamic analysis according to the type of extracted features, “static/dynamic analysis” + “Android malware detection” is applied after fusing the keywords. For machine learning-based detectors and feature selection models, the keywords are determined by combining “machine learning” with “Android malware detection” or “feature selection.” And the same searching approach is utilized for other signature and heuristics-based malware detection methods.

(3) The captured literature is mainly searched from authoritative journals or top conferences, indicating that the review has considered the most important relevant papers. Most of the reviewed works collected in this paper are from the following repositories: SpringerLink, IEEE Xplore Digital Library, Science Direct, ACM Digital Libraries, and Web of Science. Additionally, third-party online repositories such as ResearchGate, Baidu Scholar, and Google Scholar are also used by us.

(4) The papers are filtered according to the publication time to maintain the updated search of collections. Since Android security has attracted increasing attention in recent years, most of the collected papers are from the new journals from 2015 to 2021, but with a few old representative journals cited to explain some concepts. Furthermore, this work attaches great importance to research articles in recent three years.

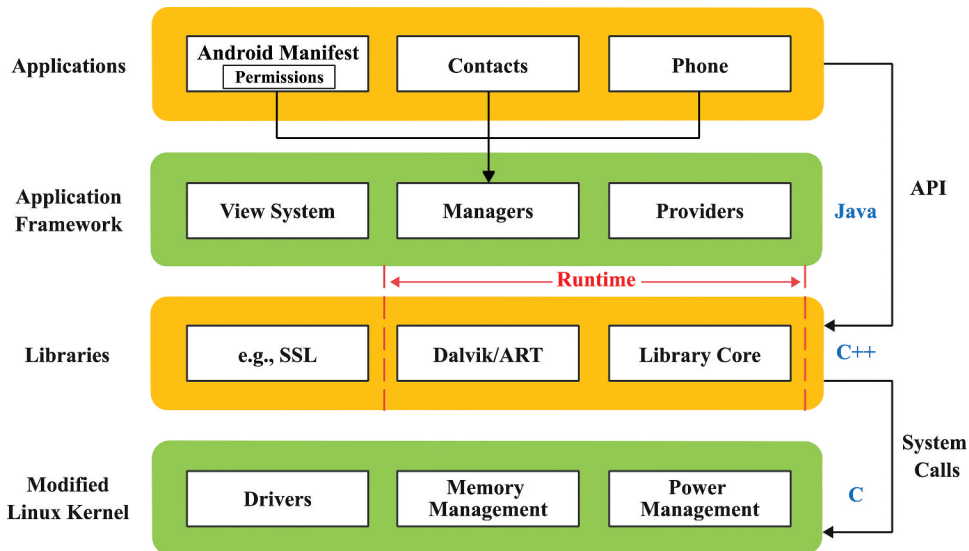
The limitations exist since these articles are manually selected, so there are inevitably some omissions in the coverage. However, these selected papers are carefully screened to meet the requirements of good quality content and high relevance. Therefore, the existing article collection is capable to provide a sufficient basis for the review work of this paper.

## Overview of Android

### *Android OS Architecture*

Android OS is an open-source operating system based on Linux for mobile platform, which is released by Google. Nowadays, Android is being updated rapidly though the key architecture of Android OS has remained unchanged. The architecture of Android can be divided into Modified Linux Kernel Layer, Libraries, System Runtime Library Layer, and Application Framework Layer as shown in [Figure 1](#).

(1) **Modified Linux Kernel.** The core system services provided by Android are based on the Linux system, such as security, power management, and drivers. Acting as an abstraction layer between hardware and software, the modified Linux kernel hides details in the hardware layer and provides services to the upper layers to reduce coupling.



**Figure 1.** Android OS architecture.

(2) **Libraries.** Linux-based process sandbox mechanism in Library Layer of Android is one of the cornerstones of the entire security design. Relied on modified Linux kernel layer for basic functions such as thread management and memory management, the Dalvik virtual machine is built with optimization to efficiently run multiple instances of virtual machines simultaneously in limited memory, and each Android application executes as a Linux process with a instance of the Dalvik virtual machine.

(3) **System Runtime.** From the perspective of the overall architecture, the developer only has active control over the System Runtime Layer and the structures above it, so the detection of Android malware should also focus on the same location. It can be classified into system library and Android runtime. The core library of Android runtime provides most of the APIs, such as Android OS, Android.net, and Android.media.

(4) **Application Framework Layer.** Android has an application framework layer that provides a variety of APIs for Android development. Developers are free to use these APIs to build their applications, subject to the security limitations of the framework's implementation.

### **Key Components of Android Apps**

The four key components of Android are Activity, Service, Broadcast Receiver, and Content Provider, which are highly associated with the behavior of Android Malware.

(1) **Activity.** The Activity provides the users with a graphical window for actions such as buttons, text blocks, input blocks, etc. Users interact with the application by tapping these elements. Activity typically acts as an intermediate layer between users and app functions, responsible for conveying user intent.

(3) **Service.** Service is often used for the time-consuming logical processing in the background therefore many malicious behaviors are associated with Service for its invisibility to users. The Service does not run in a separate process but depends on the application process in which the Service was created.

(3) **Broadcast Receiver.** As a widely used mechanism for transmitting information between applications, Broadcast Receiver is usually used by malware developers to monitor various events related to sensitive information. Broadcast Receiver filters, receives and responds to outgoing broadcasts. Broadcast Receiver allows Android Apps the ability to respond to an external event, such as powering on the phone, receiving a text message or a phone call.

(4) **Content Provider.** It is possible that Content Provider can help Android malware implement malicious behavior for getting the permission to share data. Content Provider supports storage and reading of data in multiple applications, performing as a database to applications, so it allows accessibility to the exposed data such as contact books and messages for malware developers.

The required permissions and each of the four components used in an Android application need to be registered in AndroidManifest.xml. Therefore, analyzing AndroidManifest.xml can give an overview of the functionality and malicious behavior of the applications. AndroidManifest.xml file is commonly used as an auxiliary indicator to cooperate with other analysis methods for detection (Bai et al. 2020) (Chen et al. 2021).

## Android Feature Selection

Feature selection improves malware detection efficiency by eliminating redundant and irrelevant features in Android malware detection. Figure 2 sketches the process of the feature selection phase, which is described in detail in the following subsections.

To demonstrate the importance of feature selection, (Babaagba and Adesanya 2019) tested the efficiency of feature selection in malware detection, by using supervised and unsupervised machine learning algorithms with or without feature selection. Taking the prediction accuracy of the algorithm as the performance indicator, the results showed that the best detection rate was supervised learning using a feature selection algorithm. Compared with the application without feature selection, the main accuracy jumped from 54.56% to 74.5%, which showed the influence of feature selection.

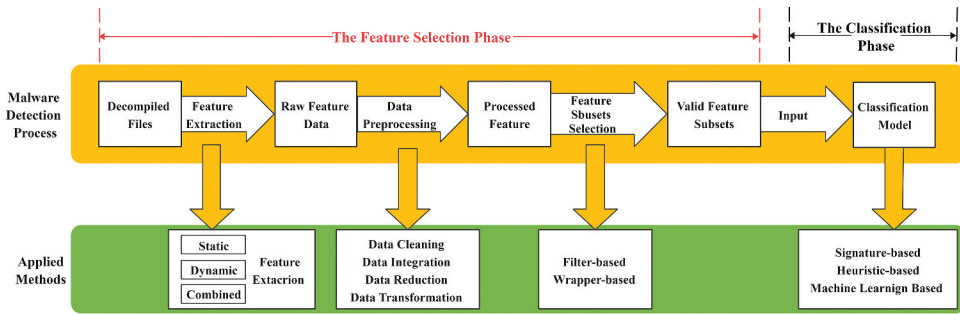


Figure 2. Detailed process of feature selection.

### Feature Extraction

According to the approach of feature extraction using static features, dynamic features, or both, Android malware detection tech can be categorized into dynamic analysis, static analysis, and hybrid analysis as illustrated in Table 1.

(1) **Dynamic analysis.** Dynamic analysis is an approach which runs the program in a sandbox environment and tracks the behavior of the program’s API call sequence, system call, network traffic, and CPU data to monitor the data flow during the program running, thus revealing the real behavior of the program processing closer to the actual situation. But it is not widely used on account of a large number of resources and the slow detection speed while running the program.

Table 1. Summary of Android feature extraction.

Reference	Features	Type	Classifiers
(Onwuzurike et al. 2017)	API calls	Dynamic analysis	RF, KNN, SVM
(Cai et al. 2016)	Behavioral features	Dynamic analysis	RF, SVM, DT, KNN, NB
(Afonso et al., 2015)	API calls and system call traces	Dynamic analysis	RF
(Dash et al. 2016)	System calls	Dynamic analysis	SVM
(Kumar et al. 2019)	Manifest files, decompiling DEX, shared libraries	Static analysis	DNN
(Nicheporuk et al., 2020)	APIs, permissions	Static analysis	CNN
(Cai et al. 2021)	Function call graph	Static analysis	SVM
(Gao 2019)	Strings, function call graph	Static analysis	SVM, KNN, RF
(Lu et al. 2019)	API calls	Static analysis	Bidirectional LSTM
(Yung and Juang 2017)	Permission, activity, service, receiver, sensitive strings, dexclass etc.	Hybrid analysis	SVM
(Chen et al. 2016)	Permissions, API calls, sensitive information request	Hybrid analysis	SVM, MLP, NB, IBK, Bagging



(Zhao et al. 2014) built a sandbox, monitored the system APIs, and used scripts to simulate various events to see the software behaviors. (Cai et al. 2018) extracted dynamic features based on API calls and inter-component communication (ICC) distilled from a behavioral characterization study, and trained a multi-class classifier using supervised machine learning. (Martin, Rodríguez-Fernández and Camacho 2018) utilized dynamic analysis with a Markov chain-based presentation to simulate the behavior of individual applications from different families of malware.

(2) **Static analysis.** Static analysis is performed by analyzing Android files and extracting information like requested permissions, opcode sequences, and API calls, etc. Static detection is widely used in the field of Android malware detection for many optional features that is easy to extract.

(Kumar et al. 2019) proposed a static analysis method based on multi-information features, using the integration of different features to promote detection accuracy. (Nicheporuk et al. 2020) took static analysis method to detect Android malware, using API method calls and permissions as features, applying convolutional neural network for training. (Suarez-Tangil et al. 2017) proposed an Android malware classifier that exploited features and artifacts introduced by obfuscation mechanisms used in malware.

(3) **Hybrid analysis.** The combination of dynamic and static analysis can make Android malware detection more accurate and efficient. (Onwuzurike et al. 2018) compared the detection performance between static and dynamic analysis on the same behavioral model relying on Markov chains built from the API sequences. The result showed that dynamic code loading has better performance for data in free conditions.

(Yung and Juang 2017) used Androguard to extract static features like permissions and Android four components, DroidBox to obtain dynamic features, and SVM to analyze the combination of the dynamic and static features. (Chen et al. 2016) introduced two newly defined features determined by the frequency of sensitive API calls and information requests. They adopted a streaming-based machine learning-based framework to support large-scale analysis, which observed app behaviors statically and dynamically.

Furthermore, API is commonly used to detect Android malware in both static feature analysis and dynamic feature analysis. After processing the API sequence or function call graph, and feeding it into RNN, the behavior path of the software, which is different from the benign software can be discovered. The problem of gigantic repetitive API sequences can be dealt through the size of the information entropy of the API (Lu et al. 2019). Some researchers (Gao 2019) (Fan et al. 2018) converted the frequency relation matrix obtained from the function call graph of sensitive into vectors and combined with other features to detect Android malware.

## **Raw Feature Data Preprocessing**

Data preprocessing is essential after obtaining the raw feature from the feature extraction phase, which generally involves the stages as follows. Table 2 gives a brief introduction to each data preprocessing method from different researches.

(1) **Data cleaning.** Data cleaning removes the irrelevant features directly after obtaining the original data through feature extraction. For example, the irrelevant permissions possessed by both benign and malicious Android applications should be cleaned.

(2) **Data integration.** Data integration means integrating information of two or more features provided for comprehensive detection. It is necessary to use effective data fusion methods when multi-source features are combined to be the input of the classifier during the detection phase.

(3) **Data reduction.** Data reduction mostly refers to the process of dimensionality reduction of the data involving complex algorithms, with an attempt to address the problems of too large dimensionality of the Android feature vector.

(4) **Data transformation.** Data transformation is to transform data from one form to another. The most commonly used research method is converting the extracted features into images and feeding them into the deep neural network.

It can be seen from Table 2 that there is a range of technologies usually applied in each stage of data preprocessing.

In the data cleaning phase, some original data are cleaned up for meeting the requirements of the next data processing step. For example, the data unable to be analyzed was deleted for subsequent N-Gram processing. Also, some papers removed redundant original features to generate a vector map containing only features associated with malicious behavior.

In the data integration phase, static features and dynamic features can be combined to obtain better performance. The multi-source features are commonly the inputs of one branch of the neural network, and the outputs of all branches are combined to form the input of a fully connected layer.

In the data reduction phase, many algorithms such as feature weighting, evolutionary genetic algorithm, and machine learning models like natural language processing were adopted to reduce the dimensionality of feature vectors. Meanwhile, some studies used the image embedding method technique to represent the features and reduce the graph dimensionality.

In the data transformation phase, many researchers transformed the extracted features into audio files, directed graphs, grayscale images, color images, etc. to take the advantage of the neural networks to process these kinds of data.

**Table 2.** Data preprocessing of Android raw features.

Reference	Description	Type
(Bai et al. 2020)	FAMD combined filter, wrapper, and embedded based feature selection method.	Data cleaning
(Xu et al. 2018)	The N-Gram technique was used to remove irrelevant API subsequences.	Data cleaning
(Pang et al.2019)	This paper removed redundant features based on feature weighting	Data cleaning
(Bai et al. 2020)	FAMD used integration of permissions and Dalvik opcode sequences.	Data integration
(Qiu et al. 2019)	A3CM combined API calls and network addresses.	Data integration
(Nicheporuk et al. 2020)	API method calls and permissions were the input of a convolutional neural network.	Data integration
(Xu et al. 2018)	Static features and dynamic features were combined.	Data integration
(Yerima and Sezer 2019)	Android permissions and API calls were used as multi-source data using static analysis.	Data integration
(Naway and Li 2019)	The integration of permission, intention filter, API call, and invalid certificate was used.	Data integration
(Bai et al. 2020)	The N-Gram technique and the FCBF algorithm were used to reduce dimensionality.	Data reduction
(Alam, Alharbi and Yildirim 2020)	This paper reduced data by DroidDomTree that mines the dominance tree of API calls.	Data reduction
(Lu et al. 2019)	Redundant N-Gram subsequences were removed using Information Gain.	Data reduction
(Li et al. 2018)	Attribute Subset Selection and Principal Component Analysis were used to reduce dimensionality.	Data reduction
(Li et al. 2018)	Three levels of permission feature pruning methods were presented	Data reduction
(Fatima et al.2019)	Evolutionary genetic algorithm was applied for feature selection.	Data reduction
(Cai, Li and Xiong 2021)	Feature weighting was employed to reduce data.	Data reduction
(Kim et al. 2018)	Image embedding method was used to reduce dimension of different graphs.	Data reduction
(Li et al. 2017)	Android malware clustering system was adopted through iterative mining of malicious payload.	Data reduction
(Bakour and Ünver 2021)	The source of APK was converted into grayscale images and processed by deep learning.	Data transformation
(Cai et al. 2021)	API sequence was converted to the enhanced function call graphs.	Data transformation
(Fan et al. 2018)	Raw data of API call sequence and sensitive data were converted to frequent subgraphs.	Data transformation
(Yen and Sun 2019)	This paper digitized the importance of the word and converted them to images.	Data transformation
(Meraldo and Santone 2021)	Features were presented in a form of audio files.	Data transformation
(Ünver and Bakour 2020)	The AndroidManifest.xml in samples was used to constructed grayscale image datasets.	Data transformation
(Vasan et al. 2020)	Raw malware binaries were converted into color images.	Data transformation

## Valid Feature Subsets Selection

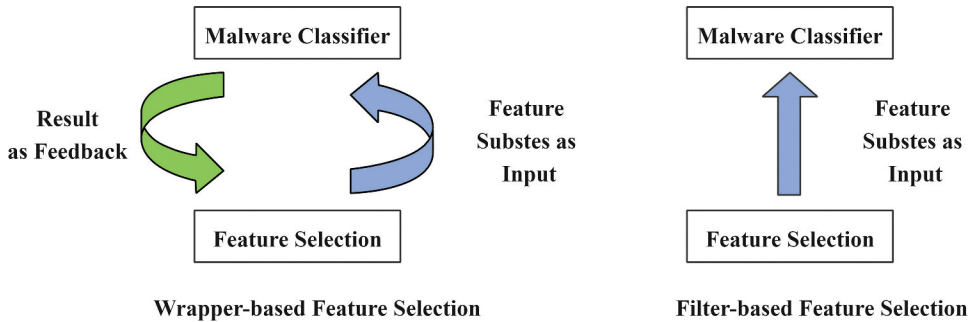
### Categories of Feature Selection

Android feature subsets selection refers to the procedure of choosing a valid subset from the existing features. According to whether the execution process of the feature selection algorithm is independent of the accuracy of the classifier or not, it can be divided into two categories. [Table 3](#) depicts the traits of each category and [Figure 3](#) illustrates the difference between them.

(1) **Filter-based Approach.** Independent of the result of the malware classifier, filter models select valid feature subsets by the general characteristics of the features. Due to the independence from classification results and high time efficiency, it has been widely used in feature subset selection.

**Table 3.** Feature subsets selection in Android.

Method	Description	characteristics	Algorithms
Filter	The model is independent of any result of classifier	Fast, lower computational cost but without considering feature relevance	Correlation-based Feature Selection (CFS), Consistency-based Filter (Dash and Liu 2003), Information Gain (Yerima, Sezer and Sezer 2014), ReliefF (Spolar et al. 2013)
Wrapper	The model uses the accuracy of the classifier to evaluate the quality of the generated feature subset	Capture feature relevance, optimize a predictor but high computational cost	FFSR (Ye and Gong 2010), WrapperSubsetEval (Witten and Frank 2011)

**Figure 3.** Categories of feature selection.

(Salah, Shalabi and Khedr 2020) proposed a lightweight Android malware classifier with a novel feature selection method inspired by TF-IDF. (Yildiz and Doğru 2019) selected three Android permission feature subsets by the genetic algorithm and evaluated by SVM and NB. (Zhang et al. 2021) used automatic word-based sensitive Android feature engineering based on text classification.

The typical filter selection algorithm is ranking-based. Each feature is assigned to a score according to its importance calculated by the algorithm, and then select the top N features as input of the classification stage. W. (Wang et al. 2014) applied feature ranking methods to rank individual permissions based on the risk of single permission and the group of permissions. (Mahindru and Sangal 2020, 2021) applied six different feature ranking approaches to select significant features, including Gain-ratio feature selection, Chi-Square, Information-gain, and logistic regression analysis.

(2) **Wrapper-based Approach.** Wrapper-based approach uses the accuracy of the malware classification to estimate the efficiency of generated feature subset. Evaluating the efficiency of generated feature subset by the accuracy of the malware classification, the wrapper method can obtain a better performance but is more complex and computationally costly compared with the filter-based method. And it can be combined with the filter-based algorithm to select Android features (Huda et al. 2016).

Recently, little research utilized the feedback from the accuracy of the classifier in Android malware detection. Though less computational overhead in the filtered-based algorithms, the valuable relevance information obtained from the classifier between different features is ignored, which can consequently choose a large number of redundant features while processing high dimension feature vectors. To make a breakthrough in the efficiency of the malware detection classifier, the problem of inexhaustible feature combination in selected valid subsets in the previous wrapper-based method should be tackled.

### *Machine Learning in Feature Subsets Selection*

Due to the vulnerability of the syntax integrity of multi-source data during the process of manual feature selection, more research has focused on feature selection machine learning algorithm based as following summarized.

Traditional machine learning models can be optimized to select the valid feature subsets. (Priya and Visalakshi 2020) proposed KNN-based relief algorithm for feature selection, and the optimized SVM algorithm was applied for malware detection with the result showing that it was equivalent to the performance of the neural network. (Wang et al. 2020) presented a multi-view neural network that can automatically generate multiple views of input and assign soft attention weights to different Android features. Multi-view preserved the rich semantic information of input without complex feature engineering.

Besides, unsupervised learning and reinforcement learning are also utilized in Android feature subsets selection. (Liu et al. 2021) proposed SRBM (Subspace-based Restricted Boltzmann Machines) by introducing the concept of subspace to optimize the model. Each RBM model in SRBM was used for unsupervised learning to learn the features of each particular subspace, and the lower dimension features are used to represent the original dataset. (Fang et al. 2019) used deep reinforcement learning to automatically select optimal feature subsets by encouraging the agent to maximize the expected accuracy from the malware classifiers in sequential interaction with the features space.

From the above discussion, the conclusion can be reached that the key to using machine learning models to select Android features is to use its prediction ability to calculate the weight of the feature or obtain the correlation between features based on an evaluation metric. Additionally, wrapper-based feature selection can also apply machine learning models to score the valid feature subset selected by the optimization algorithms.

For machine learning models that perform well on classification tasks, like SVM and DT, these models have a suitable separation capability in that they maintain the largest distance from the points in either class. For neural networks, the score derived from the sum of the softmax weights of the input features can be adopted as an evaluation indicator to select valid feature subsets.

## Classification Tech of Android Malware

### *Categories of Classification Technology*

This section outlines the process of Android malware classification based on the features obtained from valid feature subsets selection. The Android malware detection methods can be categorized into signature, behavior, and machine learning based, as summarized in [Table 4](#), among which the most mature method is signature-based detection. The following are introductions of several detection methods.

#### **(1) Signature-based detection**

Based on pattern matching, signature-based detection maintains a malware signature library containing the unique signature for each known Android malware. Malware signature library includes different attributes like file names, content strings, or bytes, that are manually identified by experts or generated automatically. It detects an Android sample by testing whether there is a matching malware signature in the library.

This technology is the most convenient and universally used due to its fast detection speed and high accuracy. All the Android malware recorded in the malware library can be detected correctly. However, the disadvantage is that the maintenance of the malware signature library is time-consuming and is not applicable to detecting new malware.

#### **(2) Heuristic-based detection**

Heuristic detection, also known as anomaly based and behavior-based detection, emphasizes the ability to identify unknown malicious software. This method compares the characteristics of unknown samples with known malware families, and each malware family is represented by a set of rules defined to mine the common experience and knowledge of the software. It is considered malware when the characteristics of the detected sample conform to the rules of one malware family. Known rule sets include attributes like software structure features, the API calls, operation code sequences, and multiple views integration rules, etc.

Heuristic-based detection techniques have the ability to self-discover of unknown malicious software and advocate the use of multiple methods to determine the difference between malicious and benign software. It makes up for the deficiency of traditional detection and can also identify unknown malicious software but with the disadvantage of a higher error rate for zero-day malware.

#### **(3) Machine learning-based detection**

Machine learning trains a learner by adjusting the parameters to make the best predictions. Existing research demonstrated that machine learning is an effective and promising method to detect Android malware. In recent years, many malware detection works have attempted to harness machine learning to seek a breakthrough in unknown Android malware detection. The following subsections will introduce the detection technology based on machine learning in detail.

**Table 4.** Summary of Android malware detection technology.

Year	Reference	Detection technology	Description
2018	(Saracino et al. 2018)	Signature and heuristics	Host-based malware detection system
2018	(Sihwail, Omar and Ariffin 2018)	Signature and heuristics	A review of signature and heuristics method
2018	(Rehman et al. 2018)	Signature and heuristics	A hybrid framework using signature and heuristics method
2021	(Li et al., 2021)	Heuristics	A review of heuristic malware detection
2010	(Firdausi et al. 2010)	Heuristics	Five classifiers were compared to verify the effectiveness of heuristics analysis
2020	(Alazab et al. 2020)	Heuristics	Integrate permissions with API calls based on heuristic analysis
2015	(Talha, Alper and Aydin 2015)	Heuristics	A permission-based Android malware detection system using heuristic analysis
2020	(Priya and Visalakshi 2020)	Machine learning	The KNN-based Relief algorithm and the optimized SVM were adopted to detect Android malware.
2018	(Chen et al. 2018)	Machine learning	Generate adversarial samples to evade the detection of current machine learning based detectors.
2020	(Pektaş and Acarman, 2020a)	Machine learning	Employ deep neural network as malware classification.
2019	(Kumar et al. 2019)	Machine learning	Combine dynamic analysis and static analysis machine learning.
2020	(Pektaş and Acarman, 2020b)	Machine learning	Deep learning was applied using features extracted from instruction call graphs
2018	(Hasegawa and Iyatomi 2018)	Machine learning	One-dimensional convolutional neural networks was applied for Android malware detection.
2019	(Yen and Sun 2019)	Machine learning	Utilize CNN to process images generated from the importance of words

### ***Machine Learning Based Android Malware Classification***

In an attempt to deal with the lack of ability to identify unknown malware or zero-day malware for the traditional methods, machine learning is employed in Android malware detection universally in recent years' research. Machine learning can be roughly divided into five categories: symbolism, bayesianism, connectionism, evolutionism, and behavioral analogism, according to the basic concept (Zhou 2016).

Firstly, according to learning type, machine learning used in Android malware detection can be divided into four categories depicted in Table 5.

(1) **Supervised learning.** The training data labeled with the category is the input into machine learning models in supervised learning. It is a classification task when supervised learning makes discrete predictions about various things, a regression task when supervised learning makes predictions about continuous values.

(2) **Unsupervised learning.** The prediction model is trained through unlabeled data sets, with the subject to explore and infer potential connections from unlabeled data in unsupervised learning. The typical tasks are clustering and dimensionality reduction.



**Table 5.** Machine learning models applied in Android malware detection categorized by learning type.

Learning type	Models	Reference
<b>Supervised learning</b>	1.DT	(Mantoo 2020)
	2.NB	(Firdaus et al. 2018)
	3.KNN	(Mantoo 2020)
	4.SVM	(Mantoo 2020)
<b>Semi-supervised learning</b>	1.SSL-NNR	(Taheri et al. 2020) (Chen et al. 2017)
	2.GIL	(Atzeni et al. 2018)
	3.Laplace SVM	(Mahdavifar et al. 2020) (Ribeiro et al. 2020)
<b>Unsupervised learning</b>	1.K-means	(Fan et al. 2019)
	2.PCA	(Wen and Yu 2017) (Tiwari and Shukla 2018)
	3.SVD	(Singh et al. 2021)
	4.ICA	(Zhang et al. 2015)
<b>Reinforcement learning</b>	1.Q-Learning	(Fang et al. 2019)
	2.DQN, DDQN	(Rathore et al. 2020)
	3.A2C, A3C	(Khowaja and Khowaja 2021)

(3) **Semi-supervised learning.** Combined with supervised learning and unsupervised learning, only some parts of training data are labeled in semi-supervised learning (Zhu and Goldberg 2009). It learns the internal structure of the data and then reasonably organizes the data for prediction with only a few marked data sets (Engelen and Hoos 2020).

(4) **Reinforcement learning.** Reinforcement learning can be applied to select Android features, using the classification result of the input data as feedback to the classification model, with the principle that the agent optimizes its next action to maximize the reward value.

Secondly, according to learning tasks, machine learning models can be categorized into classification model, regression model, clustering model and dimension reduction model. The training samples are to be classified into the given category in classification task, but without acknowledging of given categories in the clustering task. In the regression task, the input data need to fit a set of points using a function. As the machine learning models can be applied to solve different problems, it is difficult to categorize all machine learning algorithms from the same perspective. For example, decision tree can be utilized both in classification and regression tasks. There is no absolute boundary between different categories. Therefore, one model may belong to multiple categories.

With the basic knowledge of the taxonomy of machine learning methods, the commonly used models in Android malware detection were summarized as follows. Traditional machine learning and other current state-of-the-art detection models are distinguished, with a detailed summary as shown in Table 6. Three main types of models and algorithms used for Android malware detection are as follows: the first (1)-(6) is traditional machine learning models, the second are neural network and deep learning (7)-(8), and the third uses ensemble learning (9) which combines multiple classifiers to detect Android malware.



(1) **Linear Model.** Simple and highly interpretable, linear functions using Android features as input are applied to give malware prediction. The typical linear model includes logistic regression and linear regression, with the difference that logistic regression is to solve the classification problem while linear regression deals with regression problems. (Zhang et al. 2019) provided indirect methods for diagnosing anomalies by building specialized linear models to locally approximate the anomaly scores generated by black-box models.

(2) **Support Vector Machine.** It shows significant improvement to effectively monitor the resources consumption of running Android malware with Support Vector Machine (SVM). SVM is to find a hyperplane (Boswell 2002) that perfectly divides n-dimensional data into two categories. (Faiz, Hussain and Marchang 2020) applied SVM using features extracted from Android permissions, broadcast receivers, and APIs to detect Android malware, with the highest classification accuracy of 98.55% achieved by personaCateg-SVM.

(3) **Naive Bayes.** Based on Bayes' theorem, Naive Bayes (NB) assumes that the effect of an attribute value on a given class is independent of the values of other attributes (Leung 2007). (Alqahtani, Zagrouba and Almuhaideb 2019) provided a review of machine learning detectors, summarizing NB, SVM, and DNN applied in Android malware detection in detail.

(4) **Decision Tree.** As one of the most typically applied supervised learning models used in inductive reasoning, Decision Tree (DT) builds a flowchart-like tree structure from training data. (Lashkari et al. 2018) applied RF, KNN, and DT as the Android malware detection classifier for comparison, with each machine learning algorithm trained, tested, and evaluated with the same selected features.

(5) **K Nearest Neighbor.** As a supervised learning model, K Nearest Neighbor (KNN) can obtain Android malware classification results through measuring Euclidean distance in geometric space between different eigenvalues (Ray 2019).

(6) **K-means Clustering.** K-means clustering algorithm is an unsupervised learning algorithm typically applied in Android malware family classification (Ilham, Abderrahim and Abdelhakim 2018). Given a set of N data points  $R^d$  and an integer K in a real D-dimensional space, it is to find the center point in N data points, thus minimizing the mean square distance of each data point to its nearest center (Kanungo et al. 2002).

(7) **Neural Network.** Composed of a large number of connected artificial neurons, Neural Network (NN) uses neurons to reflect the received signal and the weight to present the strength of the signal (Gershenson 2003). The most typically used neural network algorithms are Perceptron Neural Network, Hopfield Neural Network, and Self-Organized Map.

**Table 6.** Summary of machine learning models in Android malware detection.

Model	Advantages	Disadvantages	Reference
Linear Model	The main algorithm of statistics; direct and fast	The premise of the algorithm is strict, and is unable to deal with high-dimensional Android features	(Ham et al. 2014)(Zhang et al. 2019)
Support Vector Machine	Easy to deal with small-scale; low dimensional and nonlinear problems	Large amount of data preprocessing; sensitive to missing values	(Mantoo 2020)(Faiz, Hussain and Marchang 2020)
Naive Bayes	Model training is simple and fast	Not available when the extracted features are correlated	(Firdaus et al. 2018)
Decision Tree	Capable to deal with large-scale and missing value model	Conspire toward overfitting; does not support online learning	(Mantoo 2020)(Lashkari et al. 2018)
Deep Learning	Address difficult problems by complex model with more parameters	Low training efficiency of complex model; easy to be overfitting	(Murtaz et al. 2018) (McLaughlin et al. 2017)
Random Forest	Easy to implement; low computational overhead; strong generalization ability	The initial performance of random forest is unsatisfactory	(Lashkari et al. 2017) (Murtaz et al. 2018)
Neural Network	High accuracy and strong fault tolerance	Requires a lot of training data; the parameters and topology are difficult to choose	(Nisa et al. 2020)(Zhu et al. 2018)
K-nearest Neighbor	Can be implemented without parameter estimation; suitable for solving multi-classification problems	Allergic to the dataset; the amount of calculation is relatively large	(Mantoo 2020) (Priya and Visalakshi 2020) (Lashkari et al. 2017)
K-means Clustering	Low computational overhead	The results are affected by the initial settings; sensitive to noise and discrete values	(Ilham, Abderrahim and Abdelhakim 2018)(Faiz, Hussain and Marchang 2020)

(8) **Deep learning.** With multiple levels of data representation obtained by composing nonlinear modules that convert a level of representation to a higher and more abstract level of representation (LeCun, Bengio and Hinton 2015), deep learning originates from NN (Du et al. 2016) as illustrated in (Qiu 2020). It is used to detect Android malware usually when features are transformed into images. A hybrid malware classification using segmentation-based fractal texture analysis and deep convolution neural network features was proposed in (Vinayakumar et al. 2018), which binarized Android APK into grayscale images generated using bytecode information. (Vinayakumar et al. 2018) used a Back-Propagation Through Time (BPTT) to train an LSTM model to detect Android malware. (Vinayakumar et al. 2018) used two different network topologies with multiple network parameters, a standard LSTM network containing only one hidden layer, and a stacked LSTM network with three hidden layers, which exhibited high Android malware detection accuracy on both static and dynamic analysis.

(9) **Ensemble learning.** Multiple classifiers were combined to improve the Android malware detection accuracy in ensemble learning (Zhao et al. 2018) (Rana and Sung 2020). More specifically, ensemble learning describes a way of combining learners. A new classifier fusion method based on the multi-level structure was proposed by (Yerima and Sezer 2019), training basic Android

classifiers at a lower level to generate models, using a set of sorting algorithms to select the final classifier and assigning the weight of the prediction results of the chosen classifier according to the prediction accuracy of the basic classifiers at a higher level. However, it is computationally costly to apply ensemble learning, for the reason that each APK file should be analyzed by multiple detectors. To tackle the problem, (Birman et al. 2019) applied deep reinforcement learning to automatically start-up and stop the base classifiers, using DNN to dynamically determine if there is adequate information to classify a given APK file.

### **Limitations and Challenges in ML Based Detection**

There are notable challenges confronting mainstream technologies especially machine learning in Android malware detection that is necessary to be considered in future work. As described in this section, these challenges can be divided into two aspects. Firstly, machine learning is vulnerable to adversarial sample attacks. Moreover, there are more serious problems caused by the upgrade of the Android ecosystem and the emergence of new malware. The machine learning-based detector suffers from degradation problems, and the feature selection algorithms are not strongly adaptable to the evolution.

### **Vulnerability to Security Attacks**

Although the enhanced performance of Android malware detection was observed in machine learning-based classifiers, a variety of countermeasures have been proposed by attackers to evade the detection. For example, they may add adversarial examples to interfere with machine learning detectors, which makes it easier to evade detection while retaining the malicious function. (Papernot et al.2016; Amodei et al. 2016) reviewed the existing work on security risk and summarized the security problems of machine learning.

As described in [Table 7](#), machine learning model security problems can be divided into three categories: training integrity threat, test integrity threat, and lack of robustness of the model. Among these different attacks, the most common situation is test integrity threat, for the little opportunities to manipulate the training dataset of the detection classifier.

Recent works have highlighted the vulnerability of many machine learning models of Android malware detection to adversarial examples, which can be used to evaluate the security and robustness of the model before it is deployed.

The existing approach to generate adversarial samples is modifying the feature vector of the Android malware, intending to be misclassified by machine learning detectors, at the same time guaranteeing the malicious functionality. (Grosse et al. 2017) used the augmented adversarial crafting

**Table 7.** Security attacks of machine learning models.

Threat types	Attack method	Description	Reference
Training integrity threat	Poison attack	The attacked model is unable to work appropriately in the test phase with poisoned data mixed into the training dataset.	(Biggio, Nelson and Laskov 2013)
	Backdoor attack	The attacked model classifies the data of the backdoor trigger into the target category due to the poisoned training set.	(Gu, Dolan-Gavitt, and Garg 2019) (Liu et al. 2018)
Test integrity threats	Adversarial sample attack	The attacked model is cheated by adversarial samples that are deliberately added with subtle interference	(Kurakin, Goodfellow and Bengio 2019) (Carlini and Wagner 2017)
Lack of robustness of the model	Not explainable	Machine learning models cannot be clearly explained how it makes decisions and why it performs well	(Papernot and Nicolas 2018)
	Insufficient training data	It destroys the adaptability of the machine learning model confronted with the changing environment for the insufficient training dataset	(Papernot and Nicolas 2018)

algorithm to mislead this classifier while adding individual features to AndroidManifest.xml to preserve semantics. (Rosenberg et al. 2017) applied a query-efficient black-box attack that generated adversarial examples by modifying the malware's API call sequences and non-sequential features.

However, there are some defects in the above attack models, which should be emphasized when building a robust detector against the adversarial samples. For example, (Grosse et al. 2017) modified AndroidManifest.xml to fool the classification model, but it fails when hybrid analysis or features contained in AndroidManifest.xml are not extracted as input to the classifier. So multi-source features extracted from different Android disassembled files, which provide more all-round analysis with comprehensive information, can be combined in future research to defense against the attack based on single feature modification.

Besides the lack of feasibility in such an adversarial attack, the impact of the mutation may also lead the Android malware to crash. In other words, the malicious behaviors could be lost or sometimes the codes cannot be compiled appropriately due to the modification of the feature vector. Therefore, to enhance the feasibility of the feature-space attacks, (Yang et al. 2017) combined malware evolution attack and malware confusion attack to preserve the critical structure of malware. Phylogenetic analysis for the Android malware family was conducted to interpret evolving malware patterns in evolution attacks, and then it was complemented by mutating permission and API features less differentiable from Android malware. Furthermore, instead of focusing on feature-space attacks, other researchers built attack models on problem-space. (Pierazzi et al. 2020) applied a problem-space attack focused on test-time evasion in the Android malware detection, through modifying real

input-space objects that correspond to an adversarial feature vector. The result of the experiment on a dataset of 170 K apps demonstrated the feasibility for an attacker to evade DREBIN (Arp et al. 2014) and its hardened version, Sec-SVM (Demontis et al. 2017).

Although the escape rate of these attacks to machine learning detectors is generally high, defenders can still build an effective classifier against the adversarial samples through some methods as follows: (i) Adversarial Training. Training a new detection model with adversarial samples. (ii) Variant Detector (Yang et al. 2017). Developing a detector in addition to the original malware detector to detect whether an app is a variant derived from existing malware. (iii) Feature Integration. Integrating more features as possible to fully extract the various information of the sample.

From the discussion, the future research focused on the Android malware classification is suggested to adopt the adversarial models to evaluate the robustness of the model, and consider the above-mentioned approach to help address the vulnerability problems of machine learning-based detectors.

### ***Deterioration Issues***

The upgrade of the Android ecosystem proposed difficulties in feature subsets selection and malware classification stage. Despite numerous malware family classification approaches being available, there remains a valuable topic since it has not been well solved. One of these challenges is how to pick features and build robust detectors that stand the test of time without frequent retraining, since a key issue is the problem caused by the evolution of the Android ecosystem.

For the Android feature selection stage, it is vitally essential to select valid Android features to build anti-malware tools that are resilient to the evolution. (Suarez-Tangil and Stringhini 2018) tracked massive amounts of malware from 2010 to 2017 and explored how the repackaging malware evolved by using differential analysis. They discussed some areas that should be specially paid attention to when extracting Android features to detect malware. Building an infrastructure able to mine a mobile software ecosystem, (Cai 2020a) depicted how the behavior of Android software has changed over time by focusing on three ecosystem elements' ecological interaction and behavioral evolution patterns. These changes in Android software tracked by the above researchers proposed challenges in the future work focused on the Android feature engineering.

For the malware classification stage, machine learning-based Android detectors have been noted that they suffer from sustainability issues. Machine learning-based detectors deteriorate due to the constant evolution

of the Android ecosystem and the new malware. The aging problem in Android malware classifier was emphasized in (Fu and Cai 2019) and identified by the framework proposed in (Jordaney et al. 2017).

Some researchers (Kantchelian et al. 2013) (Maggi et al. 2009) attempted to address the problem by frequent retraining of the malware classification model, but consequently the performance of the classifier tends to be untrustworthy with a loose retraining frequency and it results in high cost for manually labeling all Android samples in retraining process. Therefore, a thorough solution to the sustainability issues focuses on the promotion of the rapid-aging classifier. From the reviewed literature, the approach to slow-down the aging of classification models is depicted as follows.

(1) **Present Features in Abstraction.** The key to solving the problem is developing detectors resilient to changes and achieving scalability, so the concept of abstraction is utilized to make the machine learning model more adaptable for its insensitivity to the detailed changes of the Android framework. For example, (Onwuzurike et al. 2017) used the family, package, or class information to generate abstracted API calls rather than relying on the raw API calls, and they tested the model on the dataset containing samples captured over six years to display its consistency. Similarly, (Zhang et al. 2020) also dealt with the problem by exploring the semantic similarity despite the different implementations.

(2) **Track the Evolutionary Patterns.** Another strategy (Cai, 2020b) is to understand the evolutionary patterns of extracted features in benign samples and malware and then leverage the findings to build a sustainable malware detector. (Cai 2020) studied the five-year evolution trajectory of a new behavior profile described by run-time behaviors and proposed a detection system based on observations of consistent differences between benign and malicious software over years. It showed better sustainability performance than MamaDroid (Onwuzurike et al. 2016) for the ability to maintain high accuracy.

(3) **Build Self-evolving Detector.** For this method, the detection model will be updated if identified as aging in the detection stage. (Xu et al. 2019) proposed a self-evolving Android malware detection system that maintains a different set of detection models and automatically self-updates through online learning techniques to improve sustainability and reduce deterioration.

In a conclusion, as the research trends paid more attention to the deterioration problem of machine learning classification models in Android malware detection, it should not be ignored in future experiments. The researchers can demonstrate the resilience of the proposed classification model by testing it on the datasets over years without frequent retraining.

## Android Datasets and Dissemble Tools

### Datasets

As a literature survey, it is essential to propose a separate survey dimension of the datasets and disassemble tools generally used in Android malware detection. Taking experiments on well-known and updated datasets that are sufficient for a longitudinal comparative experiment is of concern in the research. Recent typically used public Android datasets are discussed as follows.

(1) **Drebin** (Arp et al. 2014). As the most-used android malware dataset in previous studies serving as a benchmark, Drebin contains 5,560 files from 179 different malware families. The applications were captured from August 2010 to October 2012 but have been never updated since then.

(2) **RmvDroid** (Wang et al. 2019). RmvDroid is a malware dataset containing 9,133 samples collected from 2014 to 2018 that belong to 56 malware families based on Google Play's app maintenance results over several years and analyzed by VirusTotal.

(3) **AndroZoo** (Allix et al. 2016). AndroZoo is a well-known collection of Android Applications mainly captured from Google Play, AnZhi, and AppChina, with samples analyzed by tens of different AntiVirus products. It is still being updated and contains 16,941,455 different benign and malware samples at present.

(4) **AndroZooOpen** (Liu et al. 2020b). As a supplement dataset for AndroZoo that is made up of close-sourced android apps, AndroZooOpen presents a growing collection of open-source Android apps collected from several sources including Github and Google Play, having over 45,000 app repositories currently.

(5) **AndroCT** (Li, Fu and Cai 2021). AndroCT is a large-scale dataset on the run-time traces of function calls in 35,974 benign and malicious Android apps from 2010 to 2019. Each app was exercised both on an emulator and a real device, and the traces were separately curated by running each sample app against automatically generated test inputs.

(6) **AMD** (Wei et al. 2017). AMD contains 24,553 samples, categorized in 135 varieties among 71 malware families ranging from 2010 to 2016. The dataset includes detailed descriptions of each malware variety's behaviors generated based on the manual analysis result.

It is notable from the above datasets that some data have not been maintained in the last three years, such as Drebin (Arp et al. 2014) and Android Malware Genome (Zhou and Jiang 2012), so using the latest Android reverse engineering tools to disassemble the outdated samples can be problematic. More significantly, these datasets no longer represent the present Android malware landscape. It is recommended to carry out experiments on well labeled, advised studied datasets released or updated after 2019 (e.g.,



Androzoo, AndroZooOpen). Most of the existing datasets are static, including no information about the runtime behavior of apps. For dynamic analysis, AndroCT is recommended for its extracted run-time trace features.

### **Dissemble Tools**

Disassembly is the reverse process of compilation, turning executable Android machine source code into higher-level code. The disassembly of the object code can be divided into static disassembly and dynamic disassembly. Static disassembly is to get the assembly code directly by parsing the binary instructions of the object code without executing the program. Dynamic disassembly, on the other hand, tracks instructions as the program executes, so dynamic disassembly can only handle instructions that the object code executes.

At present, a lot of professional disassembly tools have been produced in domestic and foreign researches in recent years. The mainstream disassembly software in the current professional field are introduced as follows.

(1) **Smali & Baksmali** (Gruver 2021). It is a powerful APK file editing tool for the Dalvik Virtual Machine to decompile and back-compile classes.dex. The syntax is a loose Jasmin in smali and– dexer syntax in Baksmali, and it implements all the features of the.dex format.

(2) **Androguard** (Halder et al. 2020). As the reverse engineering of Android applications, it functions include: support for multiple platforms (such as Linux, Windows, OSX, etc.); mainly used for static analysis; written primarily in Python; implement visualization.

(3) **APKTool** (Wiśniewski and Tumbleton 2021). The main functions of APKTool include disassembling resource files to the original format (including Resources.arsc, classes.dex, png, XML, etc.), rebuilding decoded resources back to binary APK/JAR, and processing APKs that depend on framework resources.

(4) **AndroPyTool** (Melbshark 2019). Capable to extract static and dynamic features from the Android APK, it combines various well-known Android app analysis tools such as Droidbox, FlowDroid, Strace, Androguard, and Virustotal. A source directory is needed for AndroPyTool to implement analysis and generate json and CSV format properties files.

(5) **FlowDroid** (Arzt et al. 2014). FlowDroid is an Android static taint analysis tool oriented on context, flow, fields, object sensitivity, and lifecycle awareness, which has higher accuracy and recall rates than other static analysis approaches. Based on IFSP framework, it can analyze all possible paths of information flow to generate CFG (Control Flow Graph) and label the taint leak path of sensitive information flow from source to sink.



## Conclusion

Relying on reviewing the captured work, this paper provides a systematic overview of the Android OS environment, feature selection, and classification technology. Also, the limitations of machine learning and the commonly applied datasets and disassemble tools are included. The main objective of this paper is to depict a full portrait in the field of Android malware detection, especially machine learning based.

Compared with other reviews, this paper not only gives a brief introduction to Android system mechanism and malware classification algorithms but also summarizes the data preprocessing approach and valid feature subsets selection models systematically, including the limitations and challenges in Android malware detection, which gives a more comprehensive look in the techs of feature selection and malware detection in recent years. This article can provide readers with a fundamental overview of Android malware detection and inspire them to pursue new research avenues.

However, after a comprehensive research of Android malware detection, there are still some challenges in future research, for example, the vulnerability of Android detectors to adversarial sample attacks, the aging classification models due to the emergence of new malware, the difficulty to build Android feature selection models resilient to the evolution of the Android system, etc. In addition, while there are a variety of machine learning methods used to classify Android malware, little research on these methods has focused on feature selection that has a fundamental impact on detection efficiency. Although the ensemble learning models that combine multiple learners are widely utilized, many models used traditional machine learning as base classifiers, instead of presenting the combination of state-of-the-art machine learning algorithms such as deep learning models. Moreover, the way to automatically select and remove the base classifiers can be exploited to solve the problem of computational expense and to find the optimal combinations of base classifiers. Therefore, future research can make full use of reinforcement learning in Android detection for ensemble learning.

Finally, little research utilizes the feedback from the accuracy of the classifier in Android malware detection, consequently the valuable relevance information obtained from the classifier between different features is ignored. However, to make a breakthrough in the efficiency of the wrapper-based feature selection procedure, the problem of inexhaustible feature combination in selected valid subsets in the previous wrapper-based method should be tackled.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 62101368, Grant U20A20161, and Grant U1836103; and in part by the Basic Research Program of China under Grant 2019-JCJQ-ZD-113.

## ORCID

Li Meijin  <http://orcid.org/0000-0001-8137-540X>

Fang Zhiyang  <http://orcid.org/0000-0001-6502-8053>

Zeng Qi  <http://orcid.org/0000-0002-6965-2969>

Yang Tao  <http://orcid.org/0000-0001-9658-4319>

## References

- Afonso, V. M., M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. de Geus. 2015. Identifying Android malware using dynamically obtained features. *Journal of Computer Virology & Hacking Techniques* 11 (1):9–17. 2015. doi:10.1007/s11416-014-0226-7.
- Alam, S., and Alharbi S. A., Yildirim, S. 2020. Mining nested flow of dominant APIs for detecting android malware. *Computer Networks* 167:107026. doi:10.1016/j.comnet.2019.107026.
- Alazab, M., Alazab, M., Shalaginov, A., Mesleh A., and Awajan A. 2020. Intelligent mobile malware detection using permission requests and API calls. *Future Generation Computer Systems* 107:509–21. doi:10.1016/j.future.2020.02.002.
- Allix, K., T. Bissyandé, J. Klein, and Y. Traon. (2016). AndroZoo: Collecting millions of Android apps for the research community. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. Association for Computing Machinery, New York, NY, USA, 468–71. DOI:10.1145/2901739.2903508
- Alqahtani, E. J., Zagrouba, R., and Almuhaideb A. (2019). A survey on android malware detection techniques using machine learning algorithms. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, Rome, 110–17. IEEE.
- Amodei, D., and C. Olah, Steinhardt, J., Christiano, P., Schulman, J., Mané, D. 2016. Concrete Problems in AI Safety
- Arp, D., Spreitzenbarth, Hübner, Gascon, and Rieck. 2014. Drebin: effective and explainable detection of android malware in your pocket Network and Distributed System Security Symposium February 23-26, 2014 San Diego, CA
- Arzt, S., S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, L. T. Yves, D. Oceau, and M. Patrick. (2014). FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. Association for Computing Machinery, New York, NY, USA, 259–69. DOI:10.1145/2594291.2594299
- Atzeni, A., Díaz, F., Marcelli, Sánchez, Squillero, and Tonda. 2018. Countering android malware: A scalable semi-supervised approach for family-signature generation. *IEEE Access* 6:59540–56. doi:10.1109/ACCESS.2018.2874502.
- Babaagba, and Adesanya. 2019. A study on the effect of feature selection on malware analysis using machine learning. *ACM International Conference Proceeding Series* F148151:51–55.

- Bai, H., Xie, N., X. Di, and Q. Ye. 2020. FAMD: A fast multifeature android malware detection framework, design, and implementation. *IEEE Access* 8:194729–40. doi:10.1109/ACCESS.2020.3033026.
- Bakour, and Ünver. 2021. DeepVisDroid: Android malware detection by hybridizing image-based features with deep learning techniques. *Neural Computing and Applications* 33(18):11499–516. 2021. doi:10.1007/s00521-021-05816-y.
- Bakour, Ünver, and Ghanem. 2019. The Android malware detection systems between hope and reality. *SN Applied Sciences* 1 (9):2019. doi:10.1007/s42452-019-1124-x.
- Biggio, Nelson, and Laskov. 2013. Poisoning attacks against support vector machines. *Computer Science* 28:331–44. doi:10.1007/s00450-013-0251-7.
- Birman, Hindi, Katz, and Shabtai. (2019). Transferable cost-aware security policy implementation for malware detection using deep reinforcement learning.
- Boswell. (2002). Introduction to support vector machines.
- Cai, H., N. Meng, B. Ryder, and D. Yao. 2018. Droidcat: effective android malware detection and categorization via app-level profiling. *IEEE Transactions on Information Forensics & Security* 14:1455- 70 .
- Cai, H., N. Meng, and B. Ryder, Yao, D. 2016. DroidCat: unified dynamic detection of android malware <https://vtechworks.lib.vt.edu/handle/10919/77523> ..
- Cai, H. 2020. A preliminary study on the sustainability of android malware detection. *ACM Transactions on Software Engineering and Methodology*, 1807.08221, 1–28. <http://dx.doi.org/10.1145/3371924> .
- Cai, H. 2020a. Assessing and improving malware detection sustainability through app evolution studies. *ACM Transactions on Software Engineering and Methodology*. 29 (2): 28. April 2020. Article 8. doi: 10.1145/3371924.
- Cai, H. (2020b). Embracing mobile app evolution via continuous ecosystem mining and characterization. *MOBILESoft '20: IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems* July 13-15, 2020. Seoul, ROK: ACM.
- Cai, L., Li, Y., and Xiong Z. 2021. JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers and Security* 100:102086. doi:10.1016/j.cose.2020.102086.
- Cai, M., Jiang, Y., C. Gao, Li, H., and Yuan W. 2021. Learning features from enhanced function call graphs for Android malware detection. *Neurocomputing* 423:301–07. doi:10.1016/j.neucom.2020.10.054.
- Carlini N., and Wagner D. 2017. Towards evaluating the robustness of neural networks 2017 IEEE Symposium on Security and Privacy (SP) San Jose, CA. 39–57 doi:10.1109/SP.2017.49.
- Chen, Li, Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren. 2018. Android HIV: A study of repackaging malware for evading machine-learning detection. *arXiv* 15:987–1001.
- Chen, S., M. Xue, Z. Tang, L. Xu, and H. Zhu. (2016). StormDroid: A streaminglized machine learning-based system for detecting android malware. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 377–388. Xi'an, CN: ACM.
- Chen, Shi, Xie, Yang, Fang, and Wen. 2021. Suip: an android malware detection method based on data flow features. *Journal of Physics: Conference Series* 1812 (1):12010.
- Chen, Zhang, Yang, and Sahita. (2017). Poster: semi-supervised classification for dynamic android malware detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, USA, 2479–81.
- Dash, S., G. Suarez-Tangil, S. Khan, K. Tam, and R. Holloway. 2016. DroidScribe: classifying android malware based on runtime behavior 2016 IEEE Security and Privacy Workshops (SPW) 22-26 May 2016 San Jose, CA. 252–61 doi:10.1109/SPW.2016.25.
- Dash, and Liu. 2003. Consistency-based search in feature selection. *Artificial Intelligence* 151 (1–2):155–76. doi:10.1016/S0004-3702(03)00079-1.

- Demontis, A., M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli. 2017. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable & Secure Computing* PP (99):1–1.
- Du, X., Cai, Y., Wang S., and Zhang L. (2016). Overview of deep learning. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Wuhan, CN, 159– 64. IEEE.
- Engelen, and Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* 109:02.
- Faiz, M. F. I., Hussain, M. A., and Marchang N. (2020). Android malware detection using multi-stage classification models. In *Conference on Complex, Intelligent, and Software Intensive Systems* Lodz, PL, 244–54. Springer.
- Fan, M., Liu, J., X. Luo, Chen, K., Z. Tian, Zheng, Q., and T. Liu. 2018. Android malware familial classification and representative sample selection via frequent subgraph analysis. *IEEE Transactions on Information Forensics and Security* 13 (8):1890–905. doi:10.1109/TIFS.2018.2806891.
- Fan, M., Luo, X., J. Liu, M. Wang, Nong, C., Zheng Q., and T. Liu. (2019). Graph embedding based familial analysis of android malware using unsupervised learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* Montreal, Quebec, 771–82. IEEE.
- Fang, Wang, and Geng, Kan. 2019. Feature selection for malware detection based on reinforcement learning. *IEEE Access* 7:176177–87. doi:10.1109/ACCESS.2019.2957429.
- Fatima, A., Maurya, R., M. S. Dutta, Burget, R., and Masek J. (2019). Android malware detection using genetic algorithm based optimized feature selection and machine learning. *2019 42nd International Conference on Telecommunications and Signal Processing*, Budapest, Hungary, TSP 2019, 220–23.
- Firdaus, Anuar, A., Karim N. B., and Razak. 2018. Discovering optimal features using static analysis and a genetic search based method for android malware detection. *Frontiers of Information Technology & Electronic Engineering* 19 (6):712–36. doi:10.1631/FITEE.1601491.
- Firdausi, I., Lim, C., Erwin A., and Nugroho A. S. (2010). Analysis of machine learning techniques used in behavior-based malware detection. In *Proceedings - 2010 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies*, ACT 2010, Washington, DC, 201–03.
- Fu, X., and H. Cai. (2019). On the deterioration of learning-based malware detectors for android. *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* ACM, Montreal, Quebec, 2019.
- Gao. (2019). Detecting android malicious applications based on string and function function call graph features. Master's thesis, BJTU.
- Gershenson. (2003). Artificial neural networks for beginners. *arXiv preprint cs/0308031*.
- Grosse, K., N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. 2017. Adversarial examples for malware detection ESORICS 2017: Computer Security – ESORICS Oslo, Norway (Springer International Publishing)62–79 .
- Gruver, B. (2021). JesusFreke/smali: Smali/baksmali. <https://github.com/JesusFreke/smali>
- Gu T., Dolan-Gavitt B., and Garg S. 2019. Badnets: Identifying vulnerabilities in the machine learning model supply chain. . <https://arxiv.org/abs/1708.06733>
- Halder, Reox, S., A. Desnos, and A. Desnos. (2020). androguard/androguard: Reverse engineering, Malware and goodware analysis of Android applications ... and more (ninja !). <https://github.com/androguard/androguard>
- Ham H.-S., -H.-H. Kim, M.-S. Kim, M.-J. Choi, and . 2014. Linear svm-based android malware detection for reliable iot services. *Journal of Applied Mathematics* 2014:1–10. doi:10.1155/2014/594501.

- Hasegawa, C. Iyatomi, H. (2018). One-dimensional convolutional neural networks for Android malware detection. In *Proceedings - 2018 IEEE 14th International Colloquium on Signal Processing and its Application* Penang, Malaysia, CSPA 2018.
- Huang, M. (2020). Yearly 5G smartphone production projected to exceed 200 million units thanks to smartphone brands' proactive push in 2H20, Says TrendForce. <https://www.trendforce.com/presscenter/news/20200722-10398.html>
- Huda, Abawajy, Alazab, Abdollahian, and Islam, John. 2016. Hybrids of support vector machine wrapper and filter based framework for malware detection. *Future Generation Computer Systems* 55:376–90. doi:10.1016/j.future.2014.06.001.
- Ilham, S., Abderrahim, G., and Abdelhakim B. A. (2018). Clustering android applications using k-means algorithm using permissions. In *The Proceedings of the Third International Conference on Smart City Applications* Tétouan, Morocco, 678–90. Springer.
- Internet Security Center. (2020). Android malware special report. <https://cert.360.cn/report/detail?id=0d66c8ba239680d6674f2dba9f2be5f7>
- Jordaney, R., Sharad, K., S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro. (2017). Transcend: Detecting concept drift in malware classification models. In *Proceedings of the 26th USENIX Conference on Security Symposium (SEC'17)*. USENIX Association, USA, 625–42.
- Kantchelian, A., Afroz, S., L. Huang, A. C. Islam, B. Miller, M. C. Tschantz, R. Greenstadt, A. D. Joseph, and J. D. Tygar. (2013). Approaches to adversarial drift. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security (AISeC '13)*. Association for Computing Machinery, New York, NY, USA, 99–110. DOI:10.1145/2517312.2517320
- Kanungo, Mount, Netanyahu, Piatko, Silverman, and A. Y. Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (7):881–92. doi:10.1109/TPAMI.2002.1017616.
- Khowaja, S., and P. Khuwaja. 2021. Q-learning and lstm based deep active learning strategy for malware defense in industrial iot applications. *Multimedia Tools and Applications* 1–27. doi:10.1007/s11042-020-10371-0.
- Kim, T., Kang, B., M. Rho, Sezer, S., and E. G. Im. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* PP:1–1, 08.
- Kumar, R., X. Zhang, Wang W., R. U. Khan, , J. Kumar, and A. Sharif. 2019. A multimodal malware detection technique for android iot devices using various features. *IEEE Access* 7:64411–30. doi:10.1109/ACCESS.2019.2916886.
- Kurakin, A., Goodfellow, I., and Bengio S. (2019). Adversarial examples in the physical world. . <https://arxiv.org/abs/1607.02533v4>
- Lashkari, A. H., Kadir, Taheri, and Ghorbani. (2018). Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)* Montreal, QC, 1–7. IEEE.
- Lashkari, Kadir, Gonzalez, Mbah, and Ghorbani. (2017). Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual conference on privacy, security and trust (PST)* Calgary, AB, 233–23309. IEEE.
- LeCun, Y., Bengio, Y., and Hinton G. 2015. Deep learning. *nature* 521 (7553):436–44. doi:10.1038/nature14539.
- Leung. 2007. Naive Bayesian classifier. *Polytechnic University Department of Computer Science/ Finance and Risk Engineering* (2007:123–56.
- Li, J., Wang, Z., T. Wang, J. Tang, Yang, Y., and Zhou Y. 2018. An android malware detection system based on feature fusion. *Chinese Journal of Electronics* 27 (6):1206–13. doi:10.1049/cje.2018.09.008.

- Li, L., T.F. Bissyandé, Papadakis, M., S. Rasthofer, A. Bartel, D. Ocateau, J. Klein, and L. Traon. 2017. Static analysis of android apps: A systematic literature review. *Information and Software Technology* 88(2017):67–95. 0950-5849. doi:10.1016/j.infsof.2017.04.001.
- Li, W., X. Fu, and H. Cai. (2021). AndroCT: ten years of app call traces in android. 570–74. 10.1109/MSR52588.2021.00076.
- Li, Y., Jang, J., Hu, X., Ou, X. (2017). Android malware clustering through malicious payload mining. *International symposium on research in attacks, intrusions, and defenses* Atlanta, GA. Springer, Cham, 2017. 192–214.
- Li, Zhang, Che, Guo, and Cai. 2021. A survey on feature extraction methods of heuristic malware detection. *Journal of Physics: Conference Series* 1757 (1):012071.
- Liu Y., S. Ma, Y. AaFer, W. C. Lee, and X. Zhang. 2018. Trojanning attack on neural networks Network and Distributed System Security Symposium San Diego, CA.
- Liu, K., S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu. 2020a. A review of android malware detection approaches based on machine learning. *IEEE Access* 8:124579–607. doi:10.1109/ACCESS.2020.3006143.
- Liu, P., L. Li, Y. Zhao, X. Sun, and J. Grundy. (2020b). AndroZooOpen: collecting large-scale open source android apps for the research community. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20)*. Association for Computing Machinery, New York, NY, USA, 548–52. DOI:10.1145/3379597.3387503
- Liu, Wang, Japkowicz, Tang, and Zhao. 2021. Research on unsupervised feature learning for Android malware detection based on restricted boltzmann machines. *Future Generation Computer Systems* 120:91–108. doi:10.1016/j.future.2021.02.015.
- Lu, X., , , and , Jiang, F., Zhou, X., Yi, S., Sha, J., Pietro, L. 2019. ASSCA: API sequence and statistics features combined architecture for malware detection. *Computer Networks* 157:99–111. doi:10.1016/j.comnet.2019.04.007.
- Maggi, F., Robertson, W., C. Kruegel, and G. Vigna. (2009). Protecting a Moving Target: Addressing Web Application Concept Drift. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID '09)*. Springer-Verlag, Berlin, Heidelberg, 21–40. DOI:10.1007/978-3-642-04342-0\_2
- MahdaviFar, S., Kadir, A. F. A., R. Fatemi, Alhadidi, D., and Ghorbani A. A. (2020). Dynamic android malware category classification using semi-supervised deep learning. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress* Calgary, AB, 515–22. IEEE.
- Mahindru A., and Sangal A. L. 2020. SOMDROID: Android malware detection by artificial neural network trained using unsupervised learning. Springer Berlin Heidelberg doi:10.1007/s12065-020-00518-1.
- Mahindru, and Sangal. 2021. FSDroid: A feature selection technique to detect malware from android using machine learning techniques: FSDroid. *Multimedia Tools and Applications* 80 (9):13271–323. doi:10.1007/s11042-020-10367-w.
- Mantoo B. A. (2020). A hybrid approach with intrinsic feature-based android malware detection using lda and machine learning. In *The International Conference on Recent Innovations in Computing* Jammu, India, 295–306. Springer.
- Martin, A., V. Rodriguez-Fernandez, Camacho D., and . 2018. Candyman: Classifying android malware families by modelling dynamic traces with markov chains. *Engineering Applications of Artificial Intelligence: The International Journal of Intelligent Real-Time Automation* 74:121–33. doi:10.1016/j.engappai.2018.06.006.



- McLaughlin, N., Rincon, J. M., B. Kang, Yerima, S., P. Miller, Sezer, S., Y. Safaei, Trickel, E., Z. Zhao, Doupé, A. *et al*, et al. (2017). Deep android malware detection. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy* Scottsdale, AZ, 301–08.
- Melbshark. (2019). melbshark/AndroPyTool: A framework for automated extraction of static and dynamic features from Android applications. <https://github.com/melbshark/AndroPyTool>
- Mercaldo, and Santone. 2021. Audio signal processing for Android malware detection and family identification. *Journal of Computer Virology and Hacking Techniques* 17 (2):139–52. doi:10.1007/s11416-020-00376-6.
- Murtaz, M., Azwar, H., Ali S. B., and Rehman S. A framework for android malware detection and classification. (2018). In *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)* Bangkok, Thailand, 1–5. IEEE.
- Naway, A., and W. Li. (2019). Android malware detection using autoencoder.
- Nicheporuk, A., Savenko, O., A. Nicheporuk, and Y. Nicheporuk. 2020. An android malware detection method based on CNN mixed-data model CEUR Workshop Proceedings Kharkiv, Ukraine. 2732:198–213.
- Nisa, M., Shah, J. H., S. Kanwal, Raza, M., M. A. Khan, Damaševičius, R., and Blažauskas T. 2020. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Applied Sciences* 10 (14):4966. doi:10.3390/app10144966.
- Onwuzurike, L., E. Mariconti, P. Andriotis, E. Cristofaro, G. Ross, and G. Stringhini. 2016. Mamadroid: detecting android malware by building markov chains of behavioral models. . . <https://arxiv.org/abs/1711.07477>
- Onwuzurike, L., E. Mariconti, P. Andriotis, E. Cristofaro, G. Ross, and G. Stringhini. 2017. MaMaDroid: detecting android malware by building markov chains of behavioral models (Extended version). *ACM Transactions on Privacy and Security (TOPS)* 22 (2):14. 2019.
- Onwuzurike, L., M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and Cristofaro E. D. (2018). A family of droids – Android malware detection via behavioral modeling: Static vs dynamic analysis. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)* Belfast, Ireland, (pp. 1–10). IEEE.
- Pang, Y., Peng, L., Z. Chen, Yang, B., and Zhang H. 2019. Imbalanced learning based on adaptive weighting and Gaussian function synthesizing with an application on Android malware detection. *Information Sciences* 484:95–112. doi:10.1016/j.ins.2019.01.065.
- Papernot, N., Mcdaniel, P., Sinha A., and Wellman M. 2016. Towards the science of security and privacy in machine learning. . <https://arxiv.org/abs/1611.03814>
- Papernot, and Nicolas. (2018). Characterizing the limits and defenses of machine learning in adversarial settings. PhD thesis, The Pennsylvania State University.
- Pektaş A., and Acarman T. 2020a. Deep learning for effective Android malware detection using API call graph embeddings. *Soft Computing* 24 (2):1027–43. doi:10.1007/s00500-019-03940-5.
- Pektaş A., and Acarman T. 2020b. Learning to detect Android malware via opcode sequences. *Neurocomputing* 396:599–608. 2020. doi:10.1016/j.neucom.2018.09.102.
- Pierazzi, F., Pendlebury, F., Cortellazzi J., and Cavallaro L. 2020. Intriguing properties of adversarial ml attacks in the problem space 2020 IEEE Symposium on Security and Privacy San Francisco, CA. 1332–49.
- Priya P. D., and Visalakshi P. 2020. Detecting android malware using an improved filter based technique in embedded software. *Microprocessors and Microsystems* 76 doi:10.1016/j.micpro.2020.103115.

- Qiu J., Zhang J., Luo W., L. Pan, S. Nepal, Y. Wang, and Y. Xiang. 2019. A3CM: automatic capability annotation for android malware. *IEEE Access* 7 147156–168 doi:10.1109/ACCESS.2019.2946392 .
- Qiu. 2020. *Neural network and deep learning*. Beijing: China Machine Press.
- Rana, and Sung. 2020. Evaluation of advanced ensemble learning techniques for android malware detection. *Vietnam Journal of Computer Science* 7 (2):145–59. doi:10.1142/S2196888820500086.
- Rathore, H., Sahay, S. K., Nikam P., and Sewak M. 2020. Robust android malware detection system against adversarial attacks using q-learning. *Information Systems Frontiers* 1–16. doi:10.1007/s10796-020-10083-8.
- Ray S. (2019). A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)* Faridabad, India, 35–39. IEEE.
- Rehman, Z.-U., Khan, S. N., K. Muhammad, Lee, J. W., Z. Lv, S. W. Baik, P. A. Shah, K. Awan, and I. Mehmood. 2018. Machine learning-assisted signature and heuristic-based detection of malwares in Android devices. *Computers and Electrical Engineering* 69:828–41. doi:10.1016/j.compeleceng.2017.11.028.
- Ribeiro, J., Saghezchi, F. B., G. Mantas, Rodriguez, J., Shepherd, and Abd-Alhameed R. A. 2020. An autonomous host-based intrusion detection system for android mobile devices. *Mobile Networks and Applications* 25 (1):164–72. doi:10.1007/s11036-019-01220-y.
- Rosenberg, I., A. Shabtai, L. Rokach, and Y. Elovici. (2017). Generic black-box end-to-end attack against state of the art API call based malware classifiers. <https://arxiv.org/abs/1707.05970>
- Salah, A., Shalabi, E., and Khedr W. 2020. A lightweight android malware classifier using novel feature selection methods. *Symmetry* 12 (5):858. doi:10.3390/sym12050858.
- Saracino, A., Sgandurra, D., Dini G., and Martinelli F. 2018. MADAM: effective and efficient behavior-based android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing* 15 (1):83–97. doi:10.1109/TDSC.2016.2536605.
- Sihwail, R., Omar, K., and Ariffin K. A. Z. 2018. A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. *International Journal on Advanced Science, Engineering and Information Technology* 8 (4–2):1662–71. doi:10.18517/ijaseit.8.4-2.6827.
- Singh, S., Gera, T., F. Ali, D. Thakur, K. Singh, and K.-S. Kwak. 2021. Understanding research trends in android malware research using information modelling techniques. *CMC-COMPUTERS MATERIALS & CONTINUA* 66 (3):2655–70. doi:10.32604/cmc.2021.014504.
- Spolar, N., Cherman, E. A., Monard M. C., and Lee H. D. (2013). Relieff for multi-label feature selection. In *Proceedings of the 2013 Brazilian Conference on Intelligent Systems* Fortaleza, Brazil.
- Suarez-Tangil G., and Stringhini G. 2018. Eight years of rider measurement in the android malware ecosystem: evolution and lessons learned. *arXiv preprint arXiv:1801.08115*. 2018.
- Suarez-Tangil, G., S. K. Dash, M. Ahmadi, J. Kinder, and L. Cavallaro. (2017). DroidSieve: fast and accurate classification of obfuscated android malware. In *Conference on Data and Application Security and Privacy (CODASPY)* Scottsdale, AZ,2017 doi:10.1145/3029806.3029825.
- Taheri, R., Javidan, R., M. Shojafar, Pooranian, Z., A. Miri, and M. Conti. 2020. On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications* 32 14781–800 doi:10.1007/s00521-020-04831-9.
- Talha, K. A., Alper, D. I., and Aydin C. 2015. APK auditor: permission-based android malware detection system. *Digital Investigation* 13:1–14. doi:10.1016/j.diin.2015.01.001.



- Tiwari S. R., and Shukla R. U. (2018). An android malware detection technique using optimized permission and api with pca. In *2018 Second International Conference on Intelligent Computing and Control Systems* Madurai, India, 2611–16 doi:10.1109/ICCONS.2018.8662939. IEEE.
- Ünver, H. M., and K. Bakour. 2020. Android malware detection based on image-based features and machine learning techniques. *SN Applied Sciences* 2 (7):1299. doi:10.1007/s42452-020-3132-2.
- Vasan, D., Alazab, M., S. Wassan, Naeem, H., Safaei B., and Zheng Q. 2020. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks* 171 (April 2019):107138. doi:10.1016/j.comnet.2020.107138.
- Vinayakumar, R., Soman, K. P., Poornachandran, and Kumar. 2018. Detecting android malware using long short-term memory (Lstm). *Journal of Intelligent & Fuzzy Systems* 34 (3):1277–88. doi:10.3233/JIFS-169424.
- Wang, H., J. Si, H. Li, and Y. Guo. (2019). RmvDroid: Towards a reliable Android malware dataset with app metadata. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19)* Montreal, QC. IEEE Press, 404–08. DOI:10.1109/MSR.2019.00067
- Wang, S., Chen, Z., Yan Q., Ji K., Peng L., and Yang B., Conti, M. 2020. Deep and broad URL feature mining for android malware detection *Information Sciences* . 513:600–13.
- Wang, W., X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang. 2014. Exploring permission-induced risk in android applications for malicious application detection. *IEEE Transactions on Information Forensics and Security* 9 (11):1869–82. doi:10.1109/TIFS.2014.2353996.
- Wei, F., Y. Li, S. Roy, X. Ou, and Z. Wu. (2017). Deep ground truth analysis of current android malware. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Cham, 2017.
- Wen L., and Yu H. 2017. An android malware detection system based on machine learning. *AIP Conference Proceedings* 1864:020136 doi:10.1063/1.4992953. AIP Publishing LLC.
- Wiśniewski, and Tumbleson. (2021). Apktool - A tool for reverse engineering 3rd party, closed, binary Android apps. <https://ibotpeaches.github.io/Apktool/>
- Witten I. H., and Frank E. 2011. Data mining: Practical machine learning tools and techniques. *Acm Sigmod Record* 31 (1):76–77. doi:10.1145/507338.507355.
- Xu, K., Y. Li, R. Deng, K. Chen, and J. Xu. (2019). DroidEvolver: self-evolving Android malware detection system. *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* Stockholm, Sweden, 47–62 doi:10.1109/EuroSP.2019.00014.
- Xu, L., Zhang, D., Jayasena N., and J. Cavazos. 2018. HADM: hybrid analysis for detection of malware *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016* London, UK. 16:702–24.
- Yang, W., D. Kong, T. Xie, and C. A. Gunter. (2017). Malware detection in adversarial settings: exploiting feature evolutions and confusions in Android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 2017)*. Association for Computing Machinery, New York, NY, USA, 288–302. DOI:10.1145/3134600.3134642
- Ye J.-X., and Gong X.-L. 2010. A novel fast wrapper for feature subset selection. *Journal of Changsha University of Science & Technology(Natural Science)* 7 69–73 .
- Yen Y.-S., and Sun H.-M. 2019. An Android mutation malware detection based on deep learning using visualization of importance from codes. *Microelectronics Reliability* 93 (October 2018):109–14. doi:10.1016/j.microrel.2019.01.007.
- Yerima S. Y., and Sezer S. 2019. DroidFusion: A novel multilevel classifier fusion approach for Android malware detection. *IEEE Transactions on Cybernetics* 49 (2):453–66. doi:10.1109/TCYB.2017.2777960.

- Yerima, S. Y., S. Sezer, and G. McWilliams. 2014. Analysis of Bayesian classification-based approaches for android malware detection. *IET Information Security* 8 (1):25–36. doi:10.1049/iet-ifs.2013.0095.
- Yildiz O., and Doğru I. A. 2019. Permission-based android malware detection system using feature selection with genetic algorithm. *International Journal of Software Engineering and Knowledge Engineering* 29 (2):245–62. doi:10.1142/S0218194019500116.
- Yung C.-H., and Juang W.-S. 2017. Static and dynamic integrated analysis scheme for android malware. *Journal of Electronic Science and Technology*. doi:10.11989/JEST.1674-862X.60804053.
- Zhang, N., Tan, Y., Yang C., and Li Y. 2021. Deep learning feature exploration for android malware detection. *Applied Soft Computing* 102:107069. doi:10.1016/j.asoc.2020.107069.
- Zhang, W., Ren, H., Jiang Q., and K. Zhang. (2015). Exploring feature extraction and elm in malware detection for android devices. In *International Symposium on Neural Networks Jeju, KOR*, 489–98 doi:10.1007/978-3-319-25393-0\_54. Springer.
- Zhang, X., Marwah, M., Lee I., M. Arlitt, and Goldwasser. 2019. Arlitt. An anomaly contribution explainer for cyber-security applications. . <https://arxiv.org/abs/1912.00314v1>
- Zhang, X., Y. Zhang, M. Zhong, D. Ding, Y. Cao, Y. Zhang, M. Zhang, and M. Yang. (2020). Enhancing state-of-the-art classifiers with API semantics to detect evolved Android malware. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20) Virtual Event, USA*, 757–70 doi:10.1145/3372297.3417291.
- Zhao, C., Zheng, W., L. Gong, M. Zhang, and C. Wang. (2018). Quick and accurate android malware detection based on sensitive apis. In *2018 IEEE International Conference on Smart Internet of Things Xi'an, China*, 143–48.
- Zhao, Y., Hu, L., Xiong, H., Qin, Z.-G. 2014. Dynamic analysis scheme of android malware based on sandbox. *Netinfo Security* 12 21–26 doi:10.3969/j.issn.1671-1122.2014.12.005.
- Zhou Z. 2016. *Machine Learning*. Beijing: Tsinghua University Press.
- Zhou, Y., and X. Jiang. 2012. Dissecting Android malware: characterization and evolution 2012 IEEE Symposium on Security and Privacy San Francisco, CA. 2012:95–109. doi:10.1109/SP.2012.16.
- Zhu, H.-J., You, Z.-H., Z.-X. Zhu, Shi, W.-L., X. Chen, and L. Cheng. 2018. Droiddet: Effective and robust detection of android malware using static analysis along with rotation forest model. *Neurocomputing* 272:638–46. doi:10.1016/j.neucom.2017.07.030.
- Zhu, and Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3 (1):1–130. doi:10.2200/S00196ED1V01Y200906AIM006.