Deep Learning for Android Malware Defenses: a Systematic Literature Review

YUE LIU, CHAKKRIT TANTITHAMTHAVORN, and LI LI, Monash University, Australia
YEPANG LIU, Southern University of Science and Technology, China

Malicious applications (particularly those targeting the Android platform) pose a serious threat to developers and end-users. Numerous research efforts have been devoted to developing effective approaches to defend against Android malware. However, given the explosive growth of Android malware and the continuous advancement of malicious evasion technologies like obfuscation and reflection, Android malware defense approaches based on manual rules or traditional machine learning may not be effective. In recent years, a dominant research field called deep learning (DL), which provides a powerful feature abstraction ability, has demonstrated a compelling and promising performance in a variety of areas, like natural language processing and computer vision. To this end, employing deep learning techniques to thwart Android malware attacks has recently garnered considerable research attention. Yet, no systematic literature review focusing on deep learning approaches for Android malware defenses exists. In this paper, we conducted a systematic literature review to search and analyze how deep learning approaches have been applied in the context of malware defenses in the Android environment. As a result, a total of 132 studies covering the period 2014-2021 were identified. Our investigation reveals that, while the majority of these sources mainly consider DL-based Android malware detection, 53 primary studies (40.1%) design defense approaches based on other scenarios. This review also discusses research trends, research focuses, challenges, and future research directions in DL-based Android malware defenses.

CCS Concepts: • **Security and privacy** → **Malware and its mitigation**; **Software security engineering**; • **General and reference** → **Surveys and overviews**.

Additional Key Words and Phrases: Android, malware defenses, malware analysis, malware detection, deep learning, reviews, mobile security

1 INTRODUCTION

Android is one of the most popular smartphone operating systems (OS), having dominated more than 70% of the mobile OS market share since October 2016, according to a Statista report [138]. Due to its openness and popularity, Android has become one of the primary targets of cyber-attacks [38]. Developers may take advantage of crafted malicious applications to divulge mobile user privacy or perform other dangerous operations on users' mobiles, which is extremely harmful to mobile users. On the other hand, there is a large scale of Android apps in the real world, with over 3 million Android apps available through the official store, Google Play. Although Google is constantly upgrading its protection against malicious attacks and has developed Google Play Protection (GPP) [52], it is not reliable and researchers have proved that crafted dangerous apps can easily bypass the GPP's detection [33, 68, 93, 100]. Apart from the official market, there are hundreds of unofficial and third-party markets, where the security of Android apps is highly unpredictable [96, 97, 186, 189]. Therefore, it is a pressing demand to propose an available and reliable approach to defend against malware attacks on the Android platform.

Authors' addresses: Yue Liu, yue.liu1@monash.edu; Chakkrit Tantithamthavorn, chakkrit@monash.edu; Li Li, Li.Li@monash.edu, Monash University, Melbourne, Australia; Yepang Liu, liuyp1@sustech.edu.cn, Southern University of Science and Technology, Shenzhen, China.

0360-0300/2022/6-ART $15.00
https://doi.org/10.1145/3544968

Table 1. Summary of Related work

Paper	Ref. size	Newest Ref.	Scope	Review Approach	Research trend analysis
Alqahtani et al. [3]	9	2019	Malware detection	Informal	No
Souri et al. [137]	47	2018	Malware detection	Informal	No
Qiu et al. [127]	46	2019	Malware detection	Informal	No
Naway et al. [110]	25	2018	Malware detection	Systematic search(method details not described)	No
Liu et al. [99]	113	2019	Malware detection	Systematic search	No
Wang et al. [162]	54	2020	Malware detection	Informal	No
This work	132	2021	Malware defenses	Systematic search + snowballing + quality analysis	Yes

Android malware defenses are a critical research topic in computer security. Manually analyzing malware, by formulating corresponding rules and inspecting the behaviors and source code of suspicious Android apps, is a time-consuming process—i.e., it does not scale to a large amount of Android software. Besides, with malware techniques constantly evolving, manual malware analysis couldn't keep pace with the evolving attack strategies. In recent years, a large volume of research related to automatic Android malware analysis has been proposed, utilizing data mining and machine learning approaches to achieve acceptable malware detection performance. These approaches employ a series of machine learning algorithms (e.g., support vector machine, random forest) to build a prediction model based on feature vectors extracted from the Android application package (APK) [13, 133, 168]. However, traditional machine learning algorithms are limited in their ability to learn complicated representations in high-dimensional spaces [83]. In addition, the performance of machine learning models heavily relies on the training data, and these trained models are likely to become obsolete as the Android apps evolve and software engineering advances. What's more, attackers continue to update their fraud techniques to bypass protection software as well as well-trained machine learning models in order to victimize users and businesses. In front of the increasing difficulty of Android malware defenses, it is non-trivial to construct a robust and transparent defense model or system only by traditional machine learning techniques [191].

Deep learning has emerged as the dominant research field of machine learning over the last decade, with notable achievements in many domains like speech recognition [8, 60] and image processing [142, 173]. In contrast to conventional machine learning techniques, feature extraction can be performed automatically when deep learning methods are fed with raw data. Deep learning can learn feature representation from the inputted raw data with little prior knowledge, which is the key advantage of deep learning. In 2014, deep learning tools were applied to Android malware defenses and demonstrated superior performance [184]. Subsequently, an increasing number of researchers have developed Android malware defenses models or frameworks based on a variety of deep learning techniques. As a result, an up-to-date comprehensive survey of DL-based Android malware defenses is urgently required.

The domain of Android malware defenses has been widely researched in recent years, and we present related contributions of other researchers in Table 1. Several early studies [38, 95, 145] have comprehensively reviewed Android malware techniques and traditional defensive approaches. With the wide use of advanced machine learning techniques, many researchers have reviewed relevant studies on Android malware defenses with machine learning or deep learning [3, 110, 127, 137, 162, 169]. However, these previous works couldn't provide a complete picture of current research interests and trends on DL-based Android malware defenses though they analyze all possible available methods. First, these previous studies focus only on one aspect of Android malware defenses, using machine learning/deep learning techniques to detect Android malware (ML/DL-based malware detection), but neglect other critical aspects of using deep learning to prevent/defend against malicious behaviors (e.g., malware evolution, adversarial malware detection, deployment, malware families). While distinguishing malware from benign apps is critical, enhancing Android software security is not a straightforward binary classification task. Indeed, it requires not only locating malicious applications but also comprehending malicious behaviors, to which many researchers have contributed. However, these research studies are overlooked from previous review work, making it difficult for future researchers to comprehend the state of the art of this research field. More importantly, these early surveys are not based on completed systematic approaches, and thus they could not provide a comprehensive overview of the research trends and open issues in this domain. Thus, a number of unanswered questions remain regarding the development of deep learning-based Android malware defenses. For example, the prior works still could not answer what are the state-of-the-art DL-based malware defense approaches (e.g., deep learning models and feature processing approaches) and what aspects require more research efforts in the future. Furthermore, most previous works focused on relevant studies published before 2019. However, DL-based Android malware defenses have attracted significant research attention in recent two years, which means it is necessary to conclude the significant recent research achievements. As a result, this article fills the research gap in this field by conducting a systematic and organized literature review, summarizing previous research and presenting research trends on Android malware defenses related to deep learning.

This survey aims to shape the research area of using deep learning techniques to defend against Android malware, and position existing works and current progress. Specifically, this paper makes the following contributions:

- We systematically collect and review 132 primary studies published between 2014 and 2021 on DL-based Android malware defenses.
- We present a comprehensive qualitative and quantitative synthesis based on the collected studies. Our synthesis covers the following themes: research objectives, APK characterization, deep learning techniques, deployment, and model evaluation.
- We further enumerate current issues of the existing works from different aspects and provide recommendations based on findings to support further research in this domain.
- We provide trend analysis to identify potential future trends for the research community.

The remainder of this paper is structured as follows: Section 2 presents the review methodology used in this paper. Section 3 discusses the reviewed results and open issues for the proposed research questions. Section 4 and 5 discuss potential implications and possible threats to validity of this study respectively. Finally, Section 6 concludes the paper.

2 REVIEW METHODOLOGY

In this paper, we followed the methodology suggested by Kitchenham [79] to conduct a systematic review. The main steps of the Systematic Literature Review (SLR) can be summarized as follows: (1) planning the review and developing a review protocol, (2) identifying research questions, (3) designing search strategies, proposing exclusion criteria, (4) data extraction, and (5) data synthesis. The following subsections discuss the review protocol used in this paper. Due to page limitations, we detailed the systematic review process and results online as supplementary materials [1].

2.1 Research Question

In this paper, we seek to investigate the following research questions:

- **RQ1**: What are the research objectives of the DL-based Android malware defense solutions?
- **RQ2**: What approaches have been developed for malware defenses?
  - **RQ2.1**: How are features processed for model training?
  - **RQ2.2**: What deep learning architectures are used?

[1]https://github.com/yueyueL/DL-based-Android-Malware-Defenses-review

Table 2. Search Keywords

Group	Keywords
1	Android; Mobile; Smartphone*; Phone*
2	Malware; Malicious; Malice
3	"Deep learning"; "Deep neural network*"; DNN; "Convolutional neural network*"; CNN; "Deep belief network*"; DBN; "Recurrent neural network*"; RNN; "Long short-term memory"; LSTM

Note: * means the plural form. For example, "Phone*" refers to "Phone" or "Phones".

  - **RQ2.3**: How are DL-based Android malware defenses approaches deployed in practice?
  - **RQ2.4**: How are DL-based Android malware defenses approaches evaluated?
- **RQ3**: What are the emerging and potential research trends for DL-based Android malware defenses?

2.2 Search Strategy

After identifying the research questions, the next step is searching for relevant primary studies. To this end, five popular digital libraries, including IEEE, ACM Digital Library, Springer, Science Direct, and Wiley Online Library, are identified and the searching string is constructed based on the proposed searching items proposed in Table 2. To ensure that we did not overlook any significant relevant work, we conducted further searching processes on two of the most popular research citation engines, including Web of Knowledge[2] and Google Scholar.[3] In addition, we also performed a lightweight backward snowballing [80], which means that we only carried out snowballing once, before we identified the final review list.

2.3 Data Selection Process

Only those studies related to deep learning-based Android malware defenses should be considered for further review; therefore, any primary studies that meets any of the proposed exclusion criteria would be deemed irrelevant and would be excluded from the preliminary result set. On the other hand, obtaining all relevant studies doesn't guarantee that we are able to identify the final list of papers, as it is impossible that the quality of all selected studies is desirable. For this reason, we defined a quality appraisal criterion and evaluated the quality of each paper by reading its full text. The complete list of exclusion criteria and quality appraisal criterion is available at our online supplementary materials. After these steps, we finally obtained 132 primary studies. Table 3 and in Figure 1 provided a summary for our examined papers.

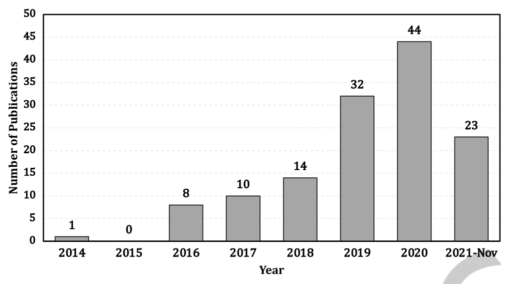
Fig. 1(a) shows the distribution of the amount of chosen studies over time. Intuitively, the number of publications related to DL-based Android malware defenses has seen a continued increase since 2014. Although we only included the public articles before November 30, 2021, in this review, the number of selected publications in 2021 is still large. These facts demonstrate that the field of Android malware defenses using deep learning is attracting growing attention, illustrating the critical need for systematic and comprehensive review work to summarize the prior work and current research trends.

On the other hand, we examined the distribution of venue domain and type for these 132 articles respectively. The results showed that over 35% of primary studies are from Security (SEC) venues, accounting for the most proportion. Both the proportion of Artificial Intelligence (AI) and Software Engineering/Programming Languages (SE/PL) is more than 10%. As for the type of venues, we found the percentage of collected studies published in conferences and journals is quite close, at about 50%. In addition, we counted the frequency of all major venues where our selected studies were published (see Fig. 1(b)). The results indicated that these primary studies were mainly collected at top venues, especially the venues in SEC domain (e.g., CCS, USENIX Security, TIFS) and more

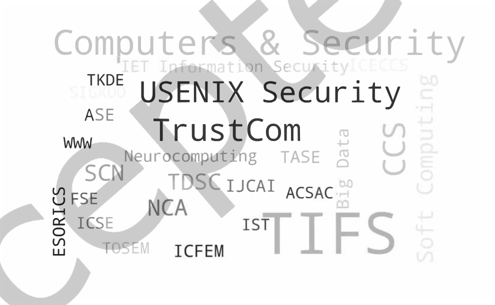
[2]https://webofknowledge.com
[3]https://scholar.google.com

Table 3. Summary of the process of data search and selection

Data source	Number of search results	Number of Candidate studies (After selection)
IEEE	201	108
ACM	1182	35
Springer	2404	36
Science Direct	1031	19
Wiley	457	8
	Merge	206
	Further Searching	328
	After Quality Assessment	132
	After Backward Snowballing	132
	Final result	132



(a) Publication count over time



(b) Word cloud of all major venue names of the sources

Fig. 1. Summary of the examined primary studies

and more relevant studies have started to be presented in top venues in SE domain recently (e.g., ICSE, ASE, and FSE).

3 RESULTS ANALYSIS

In order to answer the research questions presented in Section 2.3, we conducted a detailed review of the selected primary studies. Section 3.1 discusses the analysis results for RQ1; Section 3.2, 3.3, 3.4, and 3.5 presents the results for RQ2.1, RQ2.2, RQ2.3, and RQ2.4 respectively; while Section 3.6 presents the results of RQ3. To help our fellow researchers better understand the details for each primary study, we uploaded a detailed table in our online supplementary materials.

3.1 Malware Defenses Objectives

Deep learning techniques have been applied to various aspects of malware defenses to protect mobile users from severe malware attacks. After discussing among all authors and drawing on the classification scheme used in previous surveys by Faruki et al. [38] and Ucci et al. [149], we classify reviewed studies into the following categories: malware detection (binary classification), malware family attribution, repackaged/fake app detection, adversarial learning attacks and protections, malware evolution detection and defense, and malicious behavior analysis. Fig. 2 depicts the statistical trends of research objectives for the sources.

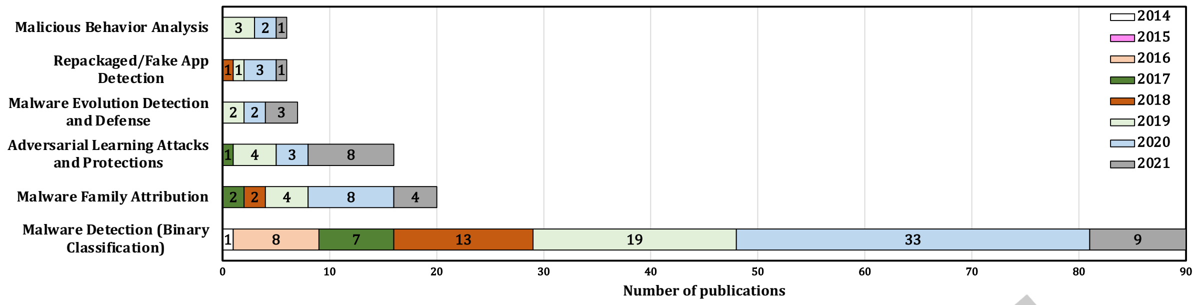


Fig. 2. Summary of the primary studies by research objectives. Some primary papers contain multiple research objectives, making the sum of percentages more than 100%.

Malware Detection (Binary Classification). As shown in Fig. 2, malware detection (binary classification), which determines whether a given application is malicious or benign, receives the most research attention (68%) and the increasing trend is expected to continue. This result is not surprising given that the most urgent task at the moment is to protect mobile users from malicious attacks by automatically distinguishing malware from goodware, which is why many previous surveys have primarily focused on this research topic. Droid-Sec [184] is the first attempt to detect Android malware using deep learning-based methods. The methodology of Droid-Sec can be summarized as three steps: (1) Android applications collection and labeling, (2) feature extraction and characterization, (3) deep learning models training and evaluation. The empirical results of Droid-Sec have demonstrated that deep learning techniques are much more effective for malware detection compared with traditional machine learning techniques like Support Vector Machine(SVM). In fact, most primary studies related to DL-based malware detection usually follow a similar methodology with Droid-Sec but explore the applicability and effectiveness of different state-of-arts deep learning techniques in more complex scenarios, which is consistent with previous literature [127].

Malware Family Attribution. Another important aspect of Android malware defenses is malware family attribution. Fig. 2 shows that 20 reviewed articles (15%) are specialized for identifying Android malware families. Given the growing number of malware variants, malware can be categorized into certain categories that are associated with different malicious objectives and behaviors, like the Adware family that displays unwanted advertisements to mobile users. In contrast to malware detection (binary classification), malware family attribution identifies which family a malware sample belongs to. Most primary studies like [192] and [141] employ multi-class classification approaches to identify existing or old malware families. As a large number of new malware variants are created, Qiu et al. [126] proposed deep learning-based approaches to detect zero-day malware families.

Repackaged/Fake App Detection. In 5% of sources, deep learning-based repackaged/fake app detection is investigated. Attackers can unpack an existing malicious/benign application, modify its contents and repackage it, depriving app developers of revenue and contributing to the spread of malware on mobile devices [94]. For this reason, identifying repackaged or fake applications and analyzing the behaviors of variants is also critical. For example, in order to locate counterfeit mobile applications in application markets, Ullah et al. [150] and Karunanayake et al. [74] propose DL-based Fake app detectors to prevent the publishing of fake apps in app stores.

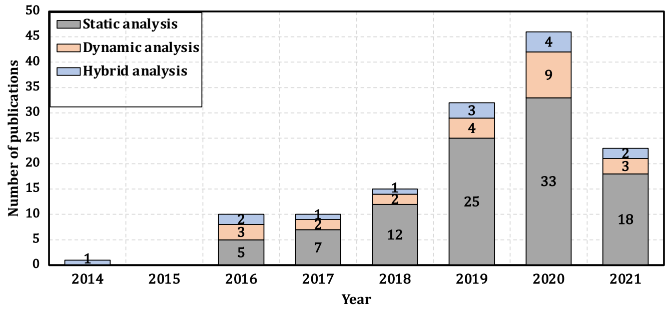
Adversarial Learning Attacks and Protections. Fig. 2 shows that 16 primary studies (12%) focus on adversarial learning attacks and protections on DL-based malware defenses. Despite the fact that numerous research studies have demonstrated that deep learning models provide promisingly high performance to identify malware, these models have been shown to be particularly vulnerable to well-designed adversarial attacks [82, 183]. Adversarial attackers could inject a small but intentional perturbation to create adversarial examples, causing the trained models to misclassify adversarial examples. For example, Chen et al. [25] performed adversarial attacks on DNN-based malware detection models, decreasing the accuracy from over 90% to 0%. Consequently, there is a corresponding increase in the attention dedicated to adversarial attacks against malware defense models, as shown in Fig. 2. Depending on when the attacks occur, adversarial attacks are split into two main categories: evasion attacks for testing samples and poisoning attacks for training samples. With respect to the two types of adversarial attacks, the majority of sources (14 studies, 87%%) discuss evasion attacks and protections for DL-based Android malware defense models, and conversely, only two recent studies focus on poisoning attacks [86, 135]. We discuss more details about this topic in Section 3.6.2.

Malware Evolution Detection and Defense. With regard to the malware evolution problem, Fig. 2 indicates that only seven papers (5%) attempt to develop solutions for malware evolution, but it is remarkable that all seven papers were published within the last three years. Due to the rapid evolution of mobile malware and the emergence of new variants and families, the performance of DL-based malware defenses models decays significantly over time. Pendlebury et al. [119] revealed that the detection performance of deep learning-based classifiers decreases drastically from almost 90% to below 30% for future malware samples. Thus, model retraining and active learning are applied to reverse and improve aged models by Pendlebury et al. [119]. However, the underlying models are still incapable of distinguishing evolved malware in this manner, as they still rely on humans to determine when models should be retrained. In the light of this issue, recent studies [37, 85, 89, 174, 178, 187] introduce a variety of approaches to slow down the aging of malware defense models, which are further discussed in Section 3.6.3.

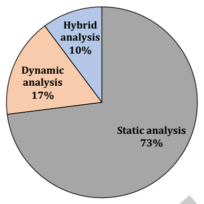
Malicious Behavior Analysis. There are six primary studies (5%) related to malicious behavior analysis in collected studies. Malicious behavior analysis aims at identifying or assessing risk behaviors in unknown applications. As for Android malware, malicious behaviors have diverse types, and a malicious application often performs more than one malicious behavior, increasing the difficulty of analysis. In addition, malicious applications may utilize code obfuscation and dynamic payload to conceal malicious behaviors. Hence, it is a relatively challenging research topic to investigate. In order to prevent malicious activities while apps are running, Gronat et al. [53] and Lorenzo et al. [30] employ recurrent neural networks to visualize potential risks for Android malware samples. For Android malware, performing malicious behaviors requires using dangerous semantic features such as permissions and API calls related to users' privacy. To assist mobile users in determining the security risk before installing unknown applications or granting permissions, some researchers examine the consistency between risk permissions and metadata-based features of apps, like descriptions [39, 42] or icon widgets [170].

DISCUSSION. Despite the rapidly growing number of research studies on deep learning for Android malware defenses, it appears that previous research studies focus on relatively simple application scenarios. More than half of the sources focus on malware detection through various deep learning strategies. Additionally, most of these existing studies focus on improving malware detection performance through the use of various advanced deep learning techniques and demonstrate that the newly proposed models outperform prior models on their own experimental datasets. It is noteworthy that an increasing number of recent studies have started to address specific issues to better apply DL-based malware detection models in practice (e.g., on-device malware detection [40, 41], explainable malware detection [167, 197], malware detection on imbalanced data [16, 112]). However, the number of relevant studies remains small. How to improve the robustness, effectiveness, stability, and reliability of malware detectors with the help of deep learning is an open issue for future researchers.

Compared with Android malware detection, the number of literature focusing on other research objectives is relatively small, requiring further in-depth research. Taking malware behavior analysis as an example, defining specific malicious behaviors and associating them with the raw code of Android APKs remain challenging issues. Thus, these research objectives require more works to integrate domain knowledge and provide fundamental theoretical construction. Except that, while this review categorizes the existing literature's research objectives



(a) Trends of program analysis approaches



(b) Overall distribution

Fig. 3. Summary of the primary studies by program analysis approaches

into six categories, the scope of Android malware defenses is actually much broader. Therefore, future research should not be limited to these six categories, but should instead propose Android malware defense approaches that leverage advanced deep learning techniques in more new application scenarios.

RQ1 What are the research objectives of the DL-based Android malware defense solutions?

- The main objective is still malware detection (binary classification) using deep learning techniques;
- On the whole, 53 primary studies focus on other research topics like malware family attribution and adversarial attacks, and the number is not small and cannot be neglected;
- At the beginning, researchers only focused on the field of malware detection, but in recent years, an increasing number of primary studies have applied deep learning to analyze Android malware in more complex scenarios.

3.2 APK Characterization

As a response to RQ2.1, this section discusses the APK feature processing approaches used in the collected studies. Each Android application is packaged as an APK file, a zip archive that primarily contains the app's manifest and bytecode. Before being fed into deep learning models, the collected Android APK data needs to be transformed into a formalized representation compatible with deep learning models. These research studies usually process APK files using reverse-engineering tools (**program analysis approaches**) and then various raw characteristics (**feature categories**) are extracted. After that, **feature encoding approaches** are utilized to perform further feature embedding operations on the raw information extracted from applications. To gain a better understanding of APK characterization mechanisms in DL-based Android malware defenses, we discuss the reviewed results from three perspectives, including program analysis approaches, feature categories, and feature encoding approaches.

3.2.1 Program analysis approaches. As shown in Fig. 3, program analysis approaches to extract raw features from Android APKs can be categorized into three types: static analysis, dynamic analysis, and hybrid analysis.

Static Analysis. Fig. 3 presents that the majority of sources (73%) extract raw features using static analysis approaches. Reverse-engineering tools such as Androguard [21] and APKtool [11] are required to disassemble and/or decompile Android APK. The raw information extracted from the APK files is used for further analysis of malicious applications. The extracted information is diverse. Raw binary code and opcode sequence can be

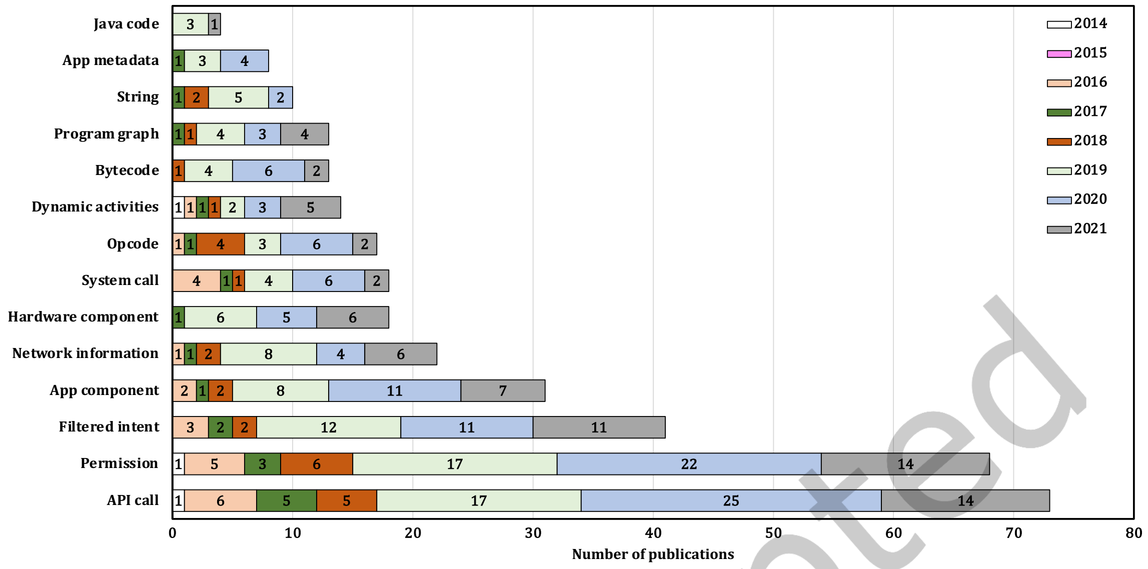


Fig. 4. Summary of the primary studies by feature categories.

fed directly to DL models [59, 66, 141, 196]. Aside from that, high-level semantic features like API calls and permissions are also widely used [54, 77, 140, 188].

Dynamic Analysis. Only 17% of primary studies use dynamic analysis approaches to collect raw features from Android APK files. This finding is not surprising given that dynamic analysis requires executing apps in a protected environment and dynamic analysis can only provide a partial picture of applications (i.e., it is challenging to cover all code) [38, 95]. However, dynamic analysis works by running samples to examine the runtime behaviors and system metrics of Android applications, which is more resilient to malware evasion techniques like obfuscation [63]. Representative dynamic analysis tools include TaintDroid [35], CopperDroid [146], etc. Dynamic features are obtained by dynamically executing collected app samples in a controlled environment, such as an Android emulator or a real mobile device. Thirteen primary studies employ emulators such as Genymotion to monitor the application's dynamic behaviors. However, various anti-emulator techniques are developed to conceal malicious activities. Thus, we also discovered that there are seven primary studies focusing on dynamic analysis on real mobile devices. For example, Alzaylaee et al. [5] demonstrated that on-device dynamic analysis performed much better than on-simulator dynamic analysis concerning stability and detecting ability.

Hybrid Analysis. Fig. 3 presents that 10% of primary studies involve hybrid program analysis (which combines static and dynamic analysis). Static program analysis has the advantage of providing full code coverage at a lower computational cost but it is vulnerable to evasion techniques like obfuscation, while dynamic program analysis allows for the analysis of run-time behaviors in a controlled environment but the code coverage may be limited [23, 154]. Although hybrid analysis leverage the complementary strengths of both types of program analyses, it is still computationally intensive, which may explain why the number of related studies is small.

3.2.2 Feature categories. As illustrated in Fig. 4, extracted features can be summarized into 13 categories, indicating the diversity of raw feature types. Note that many studies may combine multiple types of features in order to accurately represent a malicious application.

As can be observed from Fig. 4, semantic features are the most common. API calls (55.3%) and permissions (51.5%) have been the most frequently used feature types, accounting for well over half of primary studies. A possible explanation for this might be that API calls and permissions carry sufficient semantics and that the risk API calls and permissions usually result in dangerous or malicious behavior. Other types of semantic information extracted from the decompiled code such as filtered intents and app components are also used by a large number of primary studies. There are also 13 primary studies (10%) using program graphs like Control Flow Graph (CFG) and Data Flow Graph (DFG) to represent an application when analyzing Android malware. Apart from the semantic information extracted from decompressed APK, we find eight recent studies leverage app metadata such as icons and app descriptions for the subsequent analysis.

Although the aforementioned features are usually extracted via static analysis, we discover two distinct dynamic features. 18 primary studies employ Linux kernel system calls as extracted features to capture malicious behaviors. Unlike API calls, Linux kernel system calls are not dependent on the Android OS version, making them more resilient to malware evasion strategies [63]. Additionally, 14 primary studies examine characteristics associated with dynamic activities such as network access and memory dump. These observations from Fig. 4 corroborate those from Fig. 3, indicating that static analysis is the most frequently occurring approach for program analysis.

Although high-level semantic features such as API calls remain the most commonly used, there is an increasing number of primary studies using raw code sequences to construct feature vectors. Fig. 4 indicates that the most frequently occurring raw code feature is raw opcode sequences from disassembled Android apps (with 22 studies). The raw opcode sequences are fed into deep neural networks to learn high-level semantic feature representation automatically [127]. Notice that four primary sources convert disassembled code to Java source code to construct feature vectors. On the other hand, we find that 13 primary studies fed the deep neural models with the raw classes.dex bytecode. For example, R2-D2 [67] converts bytecode into a color image by mapping the bytecode's hexadecimal value to the RGB color code.

3.2.3 Feature encoding approaches. Fig. 5 provides a summary of examined sources based on feature encoding approaches. Following program analysis, the extracted information is further encoded into feature vectors and then fed into deep learning models. There are numerous ways to represent extracted features in primary studies, as extracted data from Android applications take on a variety of categories. Thus, we classify feature encoding approaches into the following five categories:

Categorical encoding. Fig. 5 indicates that categorical encoding approaches are most frequently occurring at 47% of sources (62 primary studies). This result appears to be consistent with Section 3.2.2 which indicates that categorical semantic features like API calls and permissions are the most frequently used. Typically, a numerical vector is constructed to indicate the presence of each categorical feature. It is noteworthy that we discovered that 55 out of 62 primary studies adopt one-hot encoding to record the information of the presence of each possible feature value for applications. For instance, DroidDetector [185] considers a total of 192 features through hybrid analysis, and constructs a 192-dimensional vector for each app where each feature is assigned a value of 1 if it occurs in the app; otherwise, it is assigned a value of 0. Besides, we find that seven sources assign each feature a discriminative integer and store the used features in a numerical vector. Although categorical encoding is the most prevalent strategy because of its simplicity, it has two significant drawbacks: (1).high dimensional generation, (2).embedding in isolation between distinct patterns [98].

Text-based encoding. It is quite common to employ approaches from natural language processing to encoding sequential features. Fig. 5 indicates that 26 primary studies (20%) attempt to utilize text-based feature encoding approaches. Numerous state-of-the-art text encoding approaches have been introduced to process sequential data. In fact, one-hot encoding is the simplest method of text encoding but one of its disadvantages is high dimensional problem that we discussed before. In addition, some researchers employ discrete encoding approaches like Bag of Words (BOW), Term Frequency–Inverse Document Frequency (TF-IDF), and N-Gram [9, 122, 126, 147, 150, 153, 176, 181]. These methods, however, are still limited by data sparsity and high dimensionality issues [160].

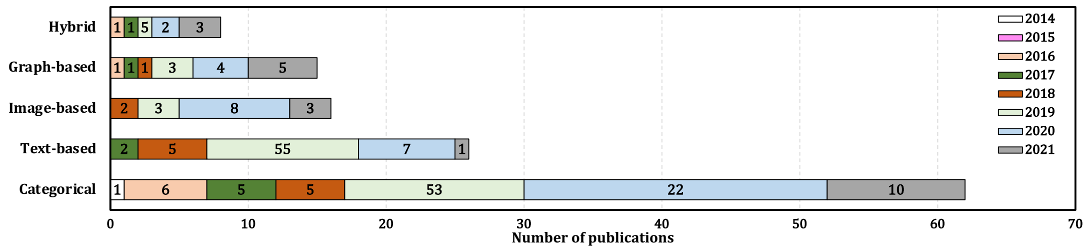


Fig. 5. Summary of the primary studies by feature encoding approaches.

Therefore, many primary studies further investigate the effectiveness of pre-trained word embedding models, such as Continuous Word2vec [18, 23, 42, 73, 158, 188, 198] and GloVe [73].

Graph-based encoding. We find that 15 primary studies (11%) employ graph-based representation approaches. Deep4MalDroid [63] obtains system calls through dynamic analysis tools to construct a weighted directed graph, and graph structure information including weights of each edge and in-degree and out-degree of each node is stored in vectors as inputs. Xu et al. [192] encode CFG and DFG into adjacency metrics respectively and combine them into a single metric in embedding layers. In [117], the authors investigate several state-of-art graph embedding approaches to encode API call graphs, including DeepWalk [121], Node2vec [55], HOPE [114], etc.

Image-based encoding. Image-based representation, employed in 16 primary studies (12%), usually transforms extracted features into a grayscale or color image. The most common scenario is directly transforming bytecode into images. For instance, IMCFN [151] reads an Android binary as a vector of 8-bit unsigned integers and then converts it into a two-dimensional array. Following that, the Android bytecode is visualized as a color image based on the RGB color map. Numerous research studies used similar approaches to encode Android bytecode [19, 59, 66, 107, 130, 141, 171].

Hybrid encoding. Combining distinct feature encoding approaches to process richer features is also common in collected research (6%). Take Kim et al. [77] as an example. The authors construct one-hot vectors to record the existence of categorical features like permission, string and app components. At the same time, in order to alleviate the impacts of obfuscation techniques, a similarity-based feature vector generation process is introduced to encode sequential features like opcode and API calls. In [74, 116, 170], since these studies also consider icons or pictures of Android applications, both image embedding approaches and text embedding algorithms are used to encoding features.

DISCUSSION. According to our reviewed results, the majority of research constructs feature vectors by recording the existence of various categorical features of Android applications. Many studies create a look-up table to list all the potential features based on prior knowledge or feature selection approaches, and then build a fixed-size one-hot feature vector to represent each application [16, 17, 36, 40, 41, 43, 50, 53, 64, 65, 91, 92, 105, 112, 129, 157, 159, 161, 167, 184, 185]. For instance, Wu et al. [167] identified 158 high-risk features to construct feature vectors (including 97 API calls and 61 permissions). However, there are several issues to process features in this way. One of these is that it is pretty difficult to define a robust malicious feature list using either humans' experience or traditional feature selection approaches. The built feature lists can't encompass all potential malicious characteristics, resulting in poor performance in the practical application. Even when all features in the training data are used, concept drift caused by Android malware evolution is a serious problem that cannot be ignored [187]. Android malware continues to evolve with similar functionality but a completely different implementation, easily evading detection by Android malware defense models. As a result, how to design effective and practical feature lists is a challenging issue.

Table 4. A summary of learning paradigms

	Supervised learning	Unsupervised & supervised	Unsupervised learning	Reinforcement learning
Counts	108	20	1	3
Percentage	81.8%	15.2%	0.8%	2.3%

As shown in Fig. 3, static program analysis is the most common approach (73%). Furthermore, our results in Section 3.2.2 show that the majority of reviewed studies extract static semantic features from disassembled files. A significant drawback of this approach is its weak ability to handle obfuscation problems. Obfuscation techniques (e.g., polymorphic code, encryption) transform malware binaries into self-compressed and uniquely structured binary files that are resistant to reverse-engineering approaches [47, 113]. Obfuscation techniques improve code protection for Android apps, but create significant barriers to malware analysis. For example, code reordering aims to modify the order of instructions in smali code but preserve the original run-time execution trace, thereby evading detection by malware defense tools [15]. By using a variety of obfuscation techniques, malware attackers can produce multiple variants of a single malicious sample, complicating malware defenses. Although some studies have shown that the proposed DL-based approaches are slightly affected by some simple obfuscation approaches [77, 84, 108, 175], we cannot ignore the fact that the real-world obfuscation techniques constantly update and evolve against anti-malware approaches [127]. Investigating obfuscated apps using deep learning techniques is a potential future research topic, and we outline some potential research trends: (1). using deep learning techniques to detect and analyze obfuscation approaches; (2). analyzing malware based on bytecode-level rather than capturing semantic features.

RQ2.1 How are features processed for model training?

- Static analysis is mostly used to obtain features, and static semantic features like API calls and permissions remain the most frequently utilized.
- The number of primary studies devoted to dynamic analysis is rising and many generally applicable methodologies/frameworks are proposed.
- One-hot encoding and text encoding are mostly used to represent features.
- 13 primary studies encode raw bytecode into feature vectors.

3.3 Deep Learning Techniques

Responding to RQ2.2, this section provides a detailed review of the primary studies according to deep learning techniques. To comprehend this section, readers are expected to be relatively familiar with deep learning. For more details on the patterns described, readers are referred to the deep learning textbook by Goodfellow et al. [51].

3.3.1 Learning paradigms. Regarding the type of deep learning paradigms, Table 4 indicates that supervised learning-based Android malware defenses appear most frequently (81.8%). It is worth noting that only one primary source employs unsupervised learning. Specifically, CADE [178] proposes an unsupervised representation learning approach to combat concept drift for security applications. Twenty primary studies (15.2%) develop Android malware defense approaches based on unsupervised & supervised scenarios. Specifically, unsupervised DNN models such as Auto-Encoders are usually employed to initialize a neural network's weights. Then, the pre-trained model can be fine-tuned using labeled samples using a standard supervised back-propagation algorithm [26, 43, 49, 63–65, 76, 101, 139, 140, 157, 159, 161, 176, 184, 185, 195, 199]. Besides, three primary studies rely on reinforcement learning to conduct the research [129, 156, 190]. These observations indicate that supervised

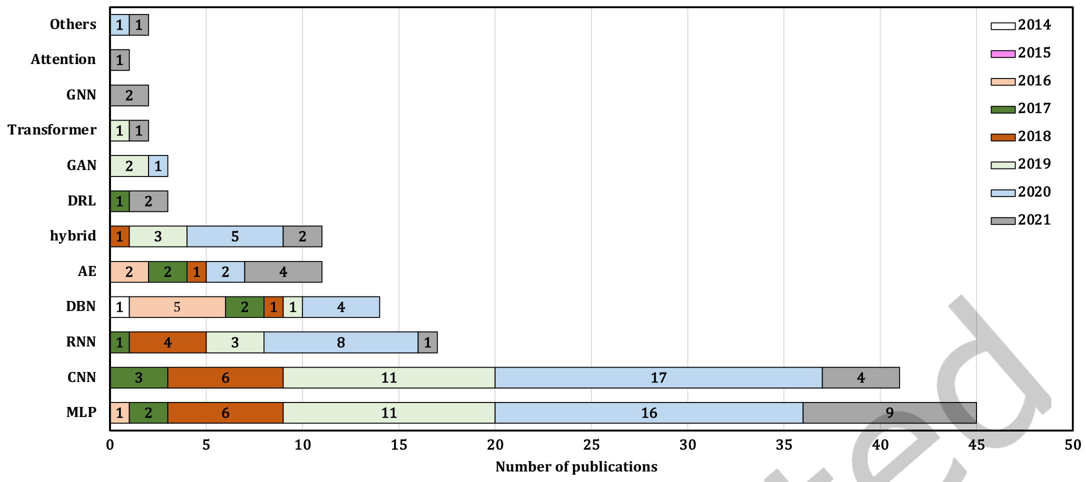


Fig. 6. Summary of the primary studies by deep learning models.

learning techniques that require sufficient labeled data occupy an absolutely dominant position in this research domain currently.

3.3.2 Deep learning models. The reviewed sources consider a variety of deep learning models. We categorize these DL models according to their model architectures and summarize the primary studies regarding DL models in Fig. 6.

Multilayer Perceptrons (MLPs), also known as deep feedforward networks or feedforward neural networks, are among the simplest deep learning models for Android malware defenses. Building an MLP is straightforward and MLP can learn hierarchical feature representation of inputs. MLPs are demonstrated to be universal approximators, capable of approximating any measurable function to any designed degree of accuracy [62]. As a result, MLPs serve as the basis of many advanced deep learning models and are widely used in a variety of research areas [51]. We thus observe from Fig. 6 that MLPs are the most frequently occurring DNN models, referring to 45 primary studies (34.1%).

Convolutional neural networks (CNNs, ConvNets) improve traditional MLPs by introducing convolution and pooling (or subsampling) operations to learn high-level features from low patterns with higher efficiency and accuracy. It is remarkable that CNNs have also been among the most popular deep learning models in Android malware defenses, accounting for 41 sources (31.1%) since 2017. We observe that numerous sources employ CNNs to learn feature representations for opcode, bytecode, and API call sequences [18, 19, 32, 59, 66, 69, 73, 90, 106, 111, 117, 141, 151, 179, 182]. This observation is unsurprising given that CNNs can automatically learn useful context structural information of Android apps in comparison to MLPs.

Recurrent Neural Networks (RNNs) have emerged as a successful paradigm for modeling sequential data since RNNs incorporate hidden units that implicitly maintain the history of all past elements in the sequence [34]. As such, it becomes a powerful tool in Natural Language Processing (NLP) and speech processing [83]. The programming pattern of Android applications is sequential and logical, and capturing sequential and semantic properties from the decompiled code is essential for improving malware defense models' performance. Fig. 6 shows that 17 primary studies (12.9%) use RNNs to defend against Android malware attacks. Except for standard RNNs, advanced variants such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) are often deployed to overcome the vanishing gradient problem, thereby enhancing model performance [23, 30, 70, 153, 154, 172, 177, 193].

Autoencoders (AEs) are unsupervised neural networks that learn the latent space of inputs in an unsupervised manner. AEs have been successfully used in dimensionality reduction and information retrieval tasks [20, 51]. The typical structure of AEs consists of two components: an encoder mapping inputs to a hidden representation, and a decoder mapping the hidden representation back. AE and its variants, such as Denoising Autoencoders (DAEs) and Variational Autoencoders (VAEs), have been widely applied to Android malware defenses, referring to 11 sources (8.3%).

Deep Belief Networks (DBNs) belong to probabilistic generative models, and DBNs are hierarchically constructed by multiple layers of stochastic hidden variables [61]. DBNs learn multiple layers of features from unlabeled inputs in an unsupervised manner, and these features can then be used to optimize discrimination in a supervised manner to perform the classification tasks. Deep belief networks are one of the first non-traditional models to admit deep architecture training successfully, but they are currently rarely used compared with other advanced deep learning networks [51]. Fig.6 presents a consistent result that DBNs were also the first DNN network to build Android malware defense models, with the highest proportion between 2014 and 2016.

Generative Adversarial Networks (GANs) are composed of a generative model and a discriminative model. The generator is trained to generate new samples in order to fool the discriminative model, while the discriminator tries to distinguish the generated samples from the real samples, liking in a game of cat and mouse to compete with each other. GANs were even described as the most interesting machine learning idea in the last ten years by AI pioneer Yann LeCun in 2016. Nevertheless, only three primary studies render GAN-based Android malware defense architectures [7, 91, 108], showing that further explorations to the application of GANs in malware defenses are still needed.

Graph Neural Networks (GNNs) are designed by extending deep learning techniques to graph data. AI researchers have developed a variety of GNN architectures, such as Graph Convolutional Network (GCN) [78] and Graph Attention Network (GAT) [152]. In Fig. 6, we can observe that two recent studies propose GNNs-based heterogeneous graph representation learning approaches in Android malware defenses [37, 48].

Attention-based neural networks are capable of learning the dependencies between inputs and target sequences, bringing a huge improvement in Machine Translation [166]. As such, one source [167] proposed an explainable attention-based Android malware detection model since attention mechanisms can provide information about the elements' relevance to their targets. Although there is currently only one primary study employing attention mechanisms, a variety of popular attention mechanisms have been proposed and demonstrated to perform well in NLP or CV, such as self-attention, soft/hard attention, local and glocal attention, co-attention, etc. As a result, we suggest that future researchers make more efforts to apply attention-based models to more specific issues in Android malware defenses.

Deep Reinforcement Learning (DRL) operates on a trial-and-error paradigm to teach an autonomous agent how to perform a task without human guidance. DRL has demonstrated its significant performance in the fields of games, robotics, and self-driving cars [14, 45]. In Android malware defenses, we also find three DRL-based primary studies [17, 129, 190]. For example, Zhao et al. [190] and Rathore et al. [129] examine the effectiveness of reinforcement learning for evading adversarial attacks and relevant protection strategies.

Transformers enable much more parallelization than CNNs and RNNs by leveraging the self-attention mechanism. This capability makes it possible to efficiently (pre-)train extremely large language models on GPUs. Bidirectional EncoderRepresentations from Transformers (BERT) is among the most widely used transformer-based pre-trained language models [31]. We found one research study that employed BRET to model sequential features on highly imbalanced malware data.

Hybrid-based models integrate several basic DNN blocks to formalize more robust and effective models. As shown in Fig. 6, hybrid-based models have been leveraged in 11 primary studies (8.3%). Numerous deep learning framework combinations have been considered, like AE and CNN [72, 159], RNN and CNN [116, 118], MLP and LSTM [175], etc.

Table 5. A summary of explanation approaches used in DL-based Android malware defenses

Tool/System.Ref	Year	DNN models	Explanation approach	Explanation type
Zhu et al. [197]	2019	CNN	Global surrogate (training a simple CNN)	Global
Pierazzi et al. [122]	2020	MLP, DBN ,CNN	Mean Decreased Impurity	Global
Fan et al. [36]	2020	MLP	LIME, SHAP, LEMNE, Anchor, LORE	Local
DENAS [24]	2020	MLP	DENAS (approximating the non-linear decision boundary of DNNs using an iterative process)	Global
Warnecke et al. [163]	2020	MLP, CNN	Gradients and Integrated Gradients, Layer-wise relevance propagation (LRP), LIME, SHAP, LEMNA	Local
Feichtner et al. [39]	2020	CNN	LIME	Local
XMal [167]	2021	Attention-based	Attention mechnism	Local
Severi et al. [135]	2021	MLP	SHAP	Local
Iadarola et al. [69]	2021	CNN	Grad-CAM	Local
CADE [178]	2021	AE	A distance-based approach (contrastive learning)	Local

Others. Recent years have seen rapid advancements in deep learning, with new deep learning techniques being proposed constantly. Two sources leverage deep learning models that fall outside of the aforementioned categories. Bai et al. [16] perform Android malware family classification through siamese neural networks. Ma et al. [103] adopt deep residual learning to detect sensitive behaviors.

Fig. 6 demonstrates that multiple types of deep learning models have been adopted to defend against Android malware. In the earlier years, unsupervised DNN models like AE and DBN drew the most research attention. However, beginning in 2017, popular supervised DNN architectures such as MLP, CNN, and RNN garnered increased attention. Another intriguing finding is that a variety of advanced approaches have emerged over the past three years, including GAN, GNN, attention, transformer, DRL, and hybrid models. These observations further support that deep learning has garnered growing interest in the field of Android malware defenses.

3.3.3 Model explanation. Deep learning approaches with a sophisticated architecture remain black-box models [109]. Specifically, these constructed models cannot provide evidence to interpret why a given sample is identified as malicious. The absence of sufficient transparency and trustworthiness in the proposed approaches is a significant obstacle to employing theoretical tools in practical malware analysis. As a result, it is necessary to develop explanation approaches for malware defense models. In collected studies, ten studies (7.5%) exploited interpretable deep learning techniques in Android malware defenses. Table 5 summarizes a comparative result towards interpretable tools. Interestingly, nine out of ten primary sources are proposed after 2019, indicating that explainable deep learning approaches for malware defenses are a current hot research topic.

With respect to the scope of interpretability, it is remarkable that most studies leverage local approaches (seven out of the 10 studies). Global methods describe how features affect the prediction on average, whereas local methods aim to explain individual predictions [109]. Regarding the explanation approaches used in sources, state-of-the-art model-agnostic explanation approaches, including LIME [131], SHAP [102], Anchor [132], LORE [56], and LEMNA [58], are most frequently occurring (four primary sources [36, 39, 135, 163]). These approaches treat the target classifier as a blackbox and approximate the decision boundary of any machine learning model by using a simple explainable model. Note that two primary studies employ gradient-based explanation approaches (e.g., integrated gradients and Grad-CAM) to back-propagate gradients through the DNN in order to measure the sensitivity of each feature [69, 163]. Wu et al. [167] design an interpretable approach to classify Android malware by leveraging a customized attention mechanism.

DISCUSSION. Our results indicate that supervised learning techniques occupy an absolutely dominant position in the current research. However, this kind of learning involves data labeling, which is costly and requires domain-specific knowledge. In Android malware defenses, Anti-Viruses (AVs) like VirusTotal are widely used to provide ground truth for experimental data. We cannot ignore the following significant issues. First, it may be convenient to use AVs to distinguish between malware and benign apps. However, AVs couldn't perform complex labeling tasks such as evolved malware labeling or malware behavior labeling, which still require substantial expertise. Second, most of the AVs work on the signature, heuristic, and behavior-based detection engines [180]. These approaches, however, are still time-consuming and human-dependent, and the more serious problem is that they cannot work well on future samples [180]. Furthermore, one commercial AV may produce inconsistent results over time, or distinct AVs may produce different results, causing the ground-truth unreliable [22]. As a result, reliable data labeling for Android malware defenses may be a potential research topic. We also encourage our fellow researchers to focus more on deep learning techniques requiring less human labor to annotate data, like active learning, semi-supervised learning, reinforcement learning, or unsupervised learning.

Although all studies we investigated in this review are related to deep learning, most studies employed neural networks with three to four layers. Training a very deep neural network on a small scale of data would cause severe overfitting [51]. Although shallow DNN networks have demonstrated promising performance in Android malware defenses, a deeper neural network is worth further exploring in this domain. On the other hand, pre-trained models trained on large amounts of data play an important role in CV and NLP domains, as they lower the barrier to applying these DNN models to real-world problems. With the explosive growth of the number of Android malware, it appears that it is not a good solution to train a DNN model from scratch every time a model is needed. A pre-trained DL model for Android malware can considerably bring convenience for the research in this domain.

With the exponential growth of Android applications, the requirement for massive computational resources to achieve the desired performance is becoming an increasing issue in this domain. In comparison to text and image, Android files are larger in size and have a more complex structure and feature processing by reverse-engineering tools is required, which is time-consuming. Furthermore, current deep learning frameworks involve a considerable amount of computational resources to approach state-of-the-art performance [125]. As a result, improving the computational efficiency of DL-based Android malware defense approaches is a growing need.

Interpretable or explainable deep learning-based Android malware defenses are also a future interesting topic [57, 148]. Recently, researchers have focused on conducting empirical studies to highlight the need of explainable AI/ML models for software engineering [71] and developing novel approaches for explainable AI/ML models for software engineering [71, 81, 120, 120, 124, 128, 164]. Although prior studies have attempted to employ local/global explainable approaches to provide explanations based on the Android characteristic-based features for each unknown sample [167], there are still several issues requiring further exploration. First, current studies mainly focus on simple semantic characteristic-based features extracted from APK, so richer feature types and in-depth explanations of source code should be investigated further. Specifically, deep learning techniques have been widely utilized to analyze raw code but how to transform the unreadable code into semantic interpretation is an unsolved problem. On the other hand, an effective evaluation system for explainable Android malware defenses is currently unavailable, making it significantly more difficult for researchers to measure the quality of explanation results and compare alternative explanation approaches. Indeed, it is nearly impossible for experienced malware analysts to detect all malicious behaviors in malware samples without making a mistake. Therefore, improving the reliability of explanations for malware samples is a potential challenge for future researchers.

Table 6. A summary of the deployment of malware defense tools

	Off-device	On-device	Distributed
Number of papers	123	2	7
Ratio	93.2%	1.5%	5.3%

RQ2.2: What deep learning architectures are used?

- MLPs, CNNs, and RNNs are mostly used in Android malware defenses.
- Research is primarily focused on supervised learning tasks, especially binary classification tasks.
- The applications of recent advanced DL techniques (e.g., GAN, attention and DRL) to Android malware defenses are still relatively preliminary.
- The interest in explainable DL-based malware defenses raises, and ten related studies have been published from 2019.

3.4 Deployment of Analysis

Deployment approaches for malware defenses can be grouped into three categories: (i) off-device, (ii) on-device, and (iii) distributed, i.e., a combination of (i) and (ii) [38].

In 93.2% of sources, the proposed tools are deployed off-device (see Table 6). Specifically, most studies design an off-device approach and conduct experiments on personal computers or higher-performance GPU servers. Automated malware defenses on a large volume of data require massive computational resources. Thus, the majority of sources didn't consider deploying the obtained DL models on mobile devices for real usage.

Conversely, only two studies propose on-device approaches [40, 41]. On-device Android malware defenses provide analysis results through the mobile device itself, without the need to share or upload private data. Currently, on-device Android malware defenses frameworks with deep learning techniques are usually implemented by transplanting the model trained on servers to smartphones. Feng et al. [40, 41] proposed two on-device Android malware detection systems, MobiDroid and MobiTive, which leveraged deep learning techniques to provide real-time detection on the user's mobile device. Specifically, they first maintained an effective Android malware detection model on the server side before migrating the pre-trained model to TensorFlow-lite [4] model. They demonstrated that the proposed approach could provide a reliable and fast reactive detection service on mobile devices.

We also found seven distributed approaches (5.3%) [4, 50, 66, 147, 150, 156, 179]. Distributed malware defense that performs on-the-fly analysis and/or detection on the mobile device while performing detailed and computationally expensive analysis on a remote server [38]. A good example is R2-D2 proposed by Hsien et al. [66] in which Android users scan a suspicious app on their own mobile device, and if the app is previously unrecognized, the app's classes.dex is transformed into an RGB image that needs to be uploaded to the server side. In their back-end server, the image will be fed into a CNN network and, once identified, the results will be sent to the users' phone. One of the major drawbacks is that distributed approaches reveal significant private data to the cloud as the uploading process via the Internet and the analysing server itself may not be secure. In the machine learning domain, Gâlvez et al. [46] utilize semi-supervised ensemble learning to implement a privacy-respect malware detector. However, privacy protection for distributed DL-based approaches still needs to be thoroughly studied.

[4]https://www.tensorflow.org/lite/

In commenting on these findings, we would encourage authors to propose more on-device and distributed malware defense approaches. Furthermore, we would encourage future authors to investigate possible solutions to real-world issues such as privacy protection and computation resource limitations.

DISCUSSION. Most of the surveyed studies proposed off-device Android malware defense models. They firstly collected a number of malware samples and performed model training and model evaluation on personal computers or GPU servers, which is a fairly common operation in the domain of deep learning applications. However, as malware techniques evolve and update, the problem of model aging is inevitable, resulting in significant performance degradation over time. When a new Android malware family is reported, it is relatively difficult for these obsolete models to update it in a responsive time. In addition, off-device analysis can't provide timely protection for mobile users. Distributed or on-device Android malware is one of the potential solutions, but the number of related research studies is relatively small at the moment. Furthermore, the prior approaches suffer from several critical flaws requiring further investigations.

The existing distributed/on-device frameworks for Android malware defenses are quite simple and limited. Firstly, suspicious Android applications must be uploaded to the server-side, which requires a heavy communication overhead. It is a potential option to assign parts of the computational tasks to smartphones. But there is a challenge to seeking the trade-off between the detection performance and the real-time demand. On the other hand, the communication process between clients and servers via the Internet may not be secure enough. It is not a difficult job for attackers to modify uploading data or steal private data. As a result, taking privacy protection into account is necessary for future work. Without a doubt, with the rapid development of deep learning techniques and smartphones, there will be new available DNN architectures supporting working effectively on mobile devices. Thus, we hope future researchers can propose more practical on-device approaches.

RQ2.3 How are DL-based Android malware defenses approaches deployed in practice?

- Most studies propose the Android malware defense models based off-device.
- There are seven papers focusing on distributed malware defenses.
- Only two papers propose on-device models that can perform a whole malware defense process on the mobile device.

3.5 Performance Evaluation

Assessing the performance of the proposed approach is an important process. As a response for RQ2.4, we analyzed the evaluation approaches utilized in the surveyed studies.

3.5.1 Dataset. First, we examined the experimental datasets used in collected studies. Fig. 7 indicates that the authors can collect malware and goodware samples from a variety of sources. With respect to goodware samples, the official Google Play Store is the most frequently used one (37.1%). Note that third-party markets such as Anzhi, HUAWEI app store, and APKPure are also popular sources for real-world Android samples. Conversely, public research datasets such as Drebin [13], AMD [165], and Genome [194] are more popular to gather malicious applications. It is remarkable that 61 primary studies collect malware samples from Drebin (46.2%). One potential disadvantage is that these datasets are not maintained or updated after being released, causing collected samples to be outdated. Taking Drebin as an example, the dataset includes 123,453 benign samples and 5,560 malware samples from 2011 to 2014. Although these datasets are widely used, it appears that the evaluation results may not reflect the real detection capability of recent malware samples. To overcome such a limitation, Fig. 7 shows that the authors show an increasing interest in online repositories like AndroZoo [10] and VirusShare [155] to collect recent malicious samples.

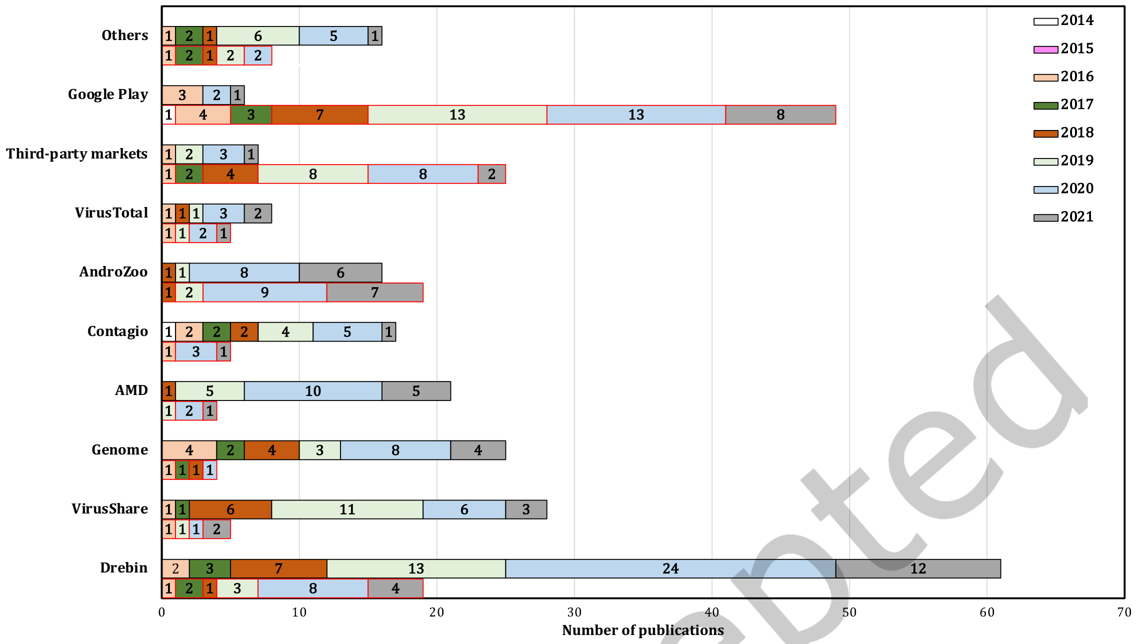
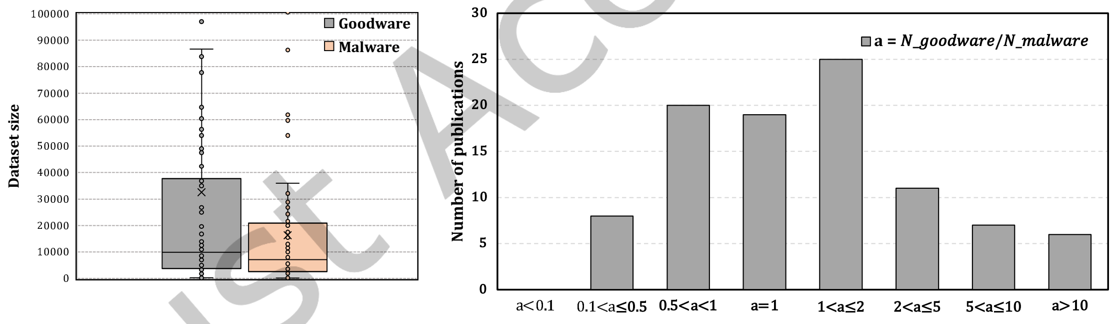


Fig. 7. Summary of the primary studies by evaluation datasets (Red frame for goodware and black frame for malware).



(a) The distribution of the number of evaluated applications

(b) The distribution of the ratio of goodware to malware

Fig. 8. Summary of the size of evaluation data

Second, for the scale of evaluation data, Fig. 8(a) represents the distribution of the number of evaluation data, while Fig. 8(b) displays the distribution of the ratio of the size of benign samples to malware samples. The median number of goodware samples used in performance evaluation is 9945, while the median number of malware samples used in performance evaluation is 7149 (see Fig. 8(a)). Fig. 8(b) shows that the goodware:malware rate of 19 primary studies is set to 1:1, constructing a balanced dataset. Pendlebury et al. [119] described that the number of goodware in the real world is much greater than the number of malware, and Android malware accounts for between 6% and 18.6% of all apps. However, it appears that only seven sources (5<a<=10) and six sources

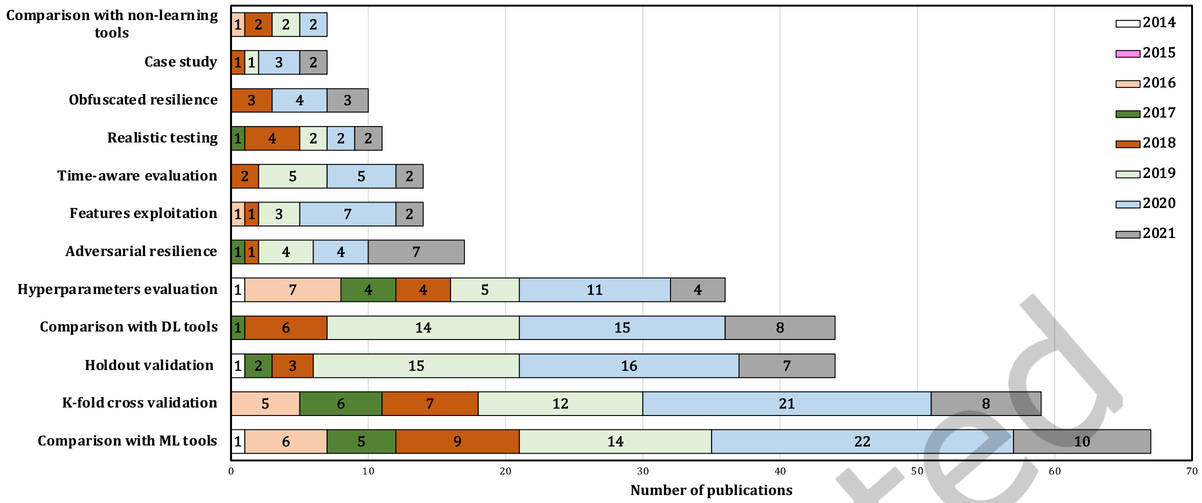


Fig. 9. Summary of the primary studies by evaluation approaches.

(a>10) appear to adhere to a realistic setting for the ratio of goodware to malware. There are even 28 primary studies constructing the evaluation dataset with more malware samples. Pendlebury et al. [119] confirmed that unrealistic assumptions about the ratio of goodware to malware cause biased performance. As such, we would encourage authors to construct evaluation data in appropriate and reliable settings.

3.5.2 Evaluation approaches. Fig. 9 summarizes the evaluation approaches used across the reviewed studies. We can observe here that 59 primary studies (44.6%) employ k-fold cross-validation and 44 primary studies (33.3%) utilize holdout validation. Due to malware evolution, 14 primary studies (10.6%) split evaluation data based on timestamps in order to perform time-aware experiments. To compare performance, we can see that 67 sources (50.7%) compare the performance with traditional machine learning approaches like Drebin [13] and MaMaDroid [104], while 44 sources (33.3%) compare the performance with other deep learning-based approaches. It is interesting to observe that seven primary studies (5.3%) compare the proposed approaches with non-learning tools such as signature-based anti-virus scanners. It is worth noting that 36 sources (27.2%) evaluate the influence of hyperparameters in deep learning models (e.g., learning rate and hidden layer size). Only seven primary studies conduct case studies to conduct an in-depth manual analysis of specific results. To prove the robustness of the proposed approaches, some studies also examine their resilience against obfuscation (7.5%) and adversarial attacks (12.8%). To demonstrate the reliability of the proposed approaches, 14 primary studies (10.6%) list the top significant features to measure the contribution of different features for the predictions. Additionally, we discover that 11 sources (8.3%) perform evaluation tests in real-world scenarios to further prove the performance.

3.5.3 Evaluation metrics. The previous results indicate that most studies focus on a classification problem. As such, evaluation metrics from traditional classification problems are directly utilized to measure the performance of proposed malware defense models in the majority of collected studies. These standard classification evaluation metrics (e.g., accuracy, precision, recall, F1-score, True Positive Rate, False Positive rate, Receiver Operating Characteristic (ROC) curve) are discussed in [99, 127] in detail. Computational efficiency is another common measure criterion in review papers. Specifically, time costs like feature processing time and hardware resource consumption like memory usages are typically considered.

To ascertain the effectiveness of these deep learning-based approaches, we also looked at the specific values of the evaluation metrics. Because accuracy and F1 score are the most frequently occurring (71.2% sources use

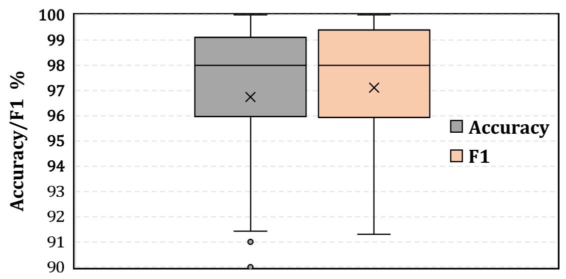


Fig. 10. The distribution of the performance metric values.

Table 7. List of publicly available tools

Tool	Year	DNN models	Open-source tool-support	Tool	Year	DNN models	Open-source tool-support
McLaughlin et al. [106]	2017	CNN	✓	Feichtner et al. [39]	2020	CNN	✓
Kim et al. [77]	2018	MLP	✓	Warnecke et al. [163]	2020	MLP, CNN	✓
R2-D2 [66]	2018	CNN		APICHECKER [50]	2020	MLP	
Vinayakumar et al. [154]	2018	LSTM	✓	DENAS [24]	2020	MLP	✓
TESSERACT [119]	2019	MLP		XMal [167]	2021	Attention	✓
Yen et al. [181]	2019	CNN	✓	Li et al. [88]	2021	AE, MLP	✓
Abderrahmane et al. [1]	2019	CNN	✓	RAMDA [92]	2021	Hybrid	✓
DeepIntent [170]	2019	Hybrid	✓	Dr.Droid [37]	2021	transformers, GNN	✓
ANDRE [188]	2019	MLP	✓	PetaDroid [72]	2021	Hybrid	✓
Karunanayake et al. [74]	2020	CNN	✓	DexRay [28]	2021	CNN	✓
Li et al. [87]	2020	MLP	✓	Li et al. [89]	2021	MLP, CNN, RNN	✓
BIHAD [116]	2020	Hybrid	✓	Severi et al. [135]	2021	MLP	✓
API-GRATH [187]	2020	MLP	✓	Iadarola et al. [69]	2021	CNN	✓
Shar et al. [136]	2020	MLP,CNN,RNN	✓	CADE [178]	2021	AE	✓
Chaulagain et al. [23]	2020	LSTM	✓	HRAT [190]	2021	DRL	

accuracy and 43.1% sources use F1 score), we record the highest accuracy and F1 score values presented in reviewed sources. Fig. 10 presents their distribution. It is remarkable that the median number of accuracy/F1 is 98%. Moreover, the accuracy/F1 values of 25% of sources are greater than 99%. Numerous research studies [12, 119] have pointed out that most studies present overly-optimistic results caused by a series of biased experimental settings (e.g., data imbalance, outdated data). However, this issue still needs to be thoroughly studied in the future. For more information about potential risks/biases that undermine the evaluation performance, readers are referred to the empirical study by Arp et al. [12].

3.5.4 Availability. The availability of proposed works in collected studies is investigated, which helps future researchers measure and certify the proposed tools. Table 7 presents a summary of the publicly available tools. In reviewed sources, there are only 30 primary studies providing publicly available tools, with just 22.7%. Among the publicly available approaches, 26 approaches are open-sourced. It is noticeable that 26 sources out of the 30 are from 2019 to 2021, indicating that the research community in DL-based Android malware defenses is showing an increasing interest in sharing their research efforts.

DISCUSSION. The first problem is that evaluation results may not reflect the real performance of the model. For example, it is quite common to combine malware datasets released a long time ago such as Drebin and Contagio, with the latest benign samples from official Android markets to build new experimental trained data [9, 30, 32, 108]. These malware datasets have not been updated in real time to include the latest malware samples, and thus contain outdated malware samples. On the other hand, as defense strategies evolve, some malicious

Table 8. Top 10 cited papers based on citations per year

Tool	Year	Cites	Cit./Tear	DNN models	Objectives
Grosse et al. [54]	2017	393	78.6	MLP	Adversarial Learning Attacks and Protections
DL-Droid [6]	2020	130	65.0	MLP	Malware Detection
McLaughlin et al. [106]	2017	313	62.6	CNN	Malware Detection
DroidDetector [185]	2016	341	56.8	DBN	Malware Detection
Kim et al. [77]	2018	219	54.8	MLP	Malware Detection
IMCFN [151]	2020	107	53.5	CNN	Malware Family Attribution
Droid-Sec [184]	2014	414	51.8	DBN	Malware Detection
MalDozer [73]	2018	207	51.8	CNN	Malware Detection, Malware Family Attribution
Wang et al. [159]	2019	154	51.3	hybrid	Malware Detection
TESSERACT [119]	2019	149	49.7	MLP	Malware Evolution Detection and Defense

technologies used in public data sets are likely to be abandoned. The model trained using these types of data is rather weak to deal with the most recent malicious apps.

The data distribution is highly imbalanced, posing a serial of challenges when performing malware analysis. First, the amount of benign samples is much larger than that of malware samples in the real-world scenario but this fact is usually overlooked by prior researchers [119]. Similarly, the number of samples differs greatly among various malware families or malware categories. For Drebin data set as an example, though its 5560 malware samples can be classified into 179 distinct families, there are only 33 families that include more than 15 examples. Thus, malware families with small scale are easy to be misclassified into the training data set's dominant family categories. Bai et al. [16] demonstrated that although many proposed approaches have been proven good performance with high accuracy, these models suffered from poor performance to predict small families, even with the help of down-sampling methods. As a result, we suggest that considering data imbalance when analyzing malware may be critical for developing a more practical malware defense approach.

Another significant issue is limited reproducibility. Our reviewed results reveal that only a small number of primary studies share their source code, which makes future researchers' validation of assessment results more difficult. Except for the source code, most studies collect evaluation data from a variety of sources but more details for the collected data are missing. Daoudi et al. [27] attempted to reproduce five ML-based Android malware detectors, but only one was successful. Thus, providing a transparent framework to manage performance evaluation is required for future researchers.

RQ2.4 How are DL-based Android malware defenses approaches evaluated?

- Evaluation data is collected from a variety of resources.
- Classic classification evaluation metrics are mostly used to measure the performance.
- Almost half of the sources reported a 98% accuracy.
- Thirty works have made their contributions publicly available.

3.6 Trend Analysis

Advanced deep learning techniques have been widely applied in Android malware defenses since 2014. Fig. 1 reveals there is a growing interest in this emerging research topic, far from reaching its peak. In order to determine general research trends, we conduct a statistic analysis of the sources and present a detailed analysis for some specific popular topics as a response to RQ3.

3.6.1 Statistic analysis. In order to locate current research concerns, we first sorted collected studies according to the number of citations per year, and the counts are based on the citation counts of Google Scholar before Dec 2021. Table 8 mentions the top 10 primary publications. It is apparent from this table that integrating deep learning

Table 9. A summary of recent papers published in top venues

Tool	Year	Venues	DNN models	Objectives
TESSERACT [119]	2019	USENIX Security	MLP	Malware Evolution Detection and Defense
AiDroid [179]	2019	IJCAI	CNN	Malware Detection
DeepIntent [170]	2019	CCS	CNN, RNN, attention, AE	Malicious Behavior Analysis
API-GRATH [187]	2020	CCS	MLP	Malware Evolution Detection and Defense, Malware Detection
Karunanayake et al. [74]	2020	TMC	CNN	Repackaged/Fake App Detection
DENAS [24]	2020	FSE	MLP	Malware Detection
Bai et al. [16]	2020	ICSE	Siamese network	Malware Family Attribution
Severi et al. [135]	2021	USENIX Security	MLP	Adversarial Learning Attacks and Protections
CADE [178]	2021	USENIX Security	AE	Malware Evolution Detection and Defense
XMal [167]	2021	TOSEM	Attention	Malware Detection
Ficco et al. [44]	2021	TC	MLP	Malware Detection
Dr.Droid [37]	2021	SIGKDD	Transformers, GNN	Malware Evolution Detection and Defense, Malware Detection
HDNFDroid [199]	2021	TKDE	AE	Malware Detection
RAMDA [92]	2021	WWW	MLP, AE	Adversarial Learning Attacks and Protections
HRAT [190]	2021	CCS	MLP	Adversarial Learning Attacks and Protections

techniques into malware detection is still the main focus (seven out of the ten sources). But specifically, these research studies employ various deep learning models (e.g., MLP, CNN, DBN.) to detect malicious applications. Note that Grosse et al. [54] has the largest number of citations, and one reason for the high citation counts is that this work investigates the viability of adversarial attack techniques against neural networks in this field, which sets the foundations for future work. Another work [119] published in 2019 that examines malware evolution is also a highly cited paper. These facts demonstrate that the robustness of DL-based detection models is attracting increased research attention.

Although sorting these primary studies based on citations can help us identify high-impacted work, it is not applicable to recent work due to the time delay. We, therefore, list all recent publications (from 2019 to 2021) presented in top venues that have the highest quality ranking (A*) in the CORE ranking system, as shown in Table 9, which can help us seek the current research focus of top researchers. It is worth noting that top venues, especially in the security domain (CCS and USENIX Security), publish an increasing number of relevant studies. Although most listed papers in this table still focus on malware detection, they make deeper analysis on more specific issues in Android malware detection, such as evaluation metrics, data imbalance problems, interpretability, model aging, etc. It is also noteworthy that "Adversarial Learning Attacks and Protections" and "Malware Evolution Detection and Defense" are frequently occurring in the top venues recently. These two issues are closely related to the practicability and effectiveness of the proposed architectures, which belong to key areas for future research to improve the state of the art in Android malware defenses.

Table 8 and 9 further summarize the deep learning architectures used in these listed studies. Fig. 6 displays that CNN, RNN, and MLP have been widely used in Android malware defenses in recent years, MLP appears most frequently among listed important works. This is not a surprising outcome. MLP is the simplest but quintessential deep neural network, and thus researchers would tend to conduct experiments based on MLP networks first when they have a new research idea. Compared with the sources listed in Table 8, recent studies adopt more advanced deep learning models such as attention-based networks and GNN. These observations also illustrate that the application of advanced deep learning techniques to the Android malware defenses domain is actually in a preliminary stage. In recent years, deep learning has brought impressive progress in many domains and many modern DL approaches are up-to-come, such as deep active learning, reinforcement learning, transfer learning, controllable generative models, etc. However, the application of deep learning technologies in the field of mobile security is far behind the development of deep learning itself. Research toward DL-based Android malware defenses is active currently, and hence we believe more advancements are expected in the near future.

Table 10. A summary of adversarial attacks and defenses

Category	Percentage	Papers
Target phase		
Evasion attacks	87.5%	[25, 29, 54, 75, 87–89, 91, 92, 123, 129, 143, 144, 190]
Poisoning attacks	12.5%	[86, 135]
Attacks scenarios		
White-box	31.3%	[29, 54, 75, 143, 190]
White-box & Black-box	37.5%	[87, 88, 92, 123, 129, 135]
Black-box	31.3%	[25, 86, 89, 91, 144]
Adversarial defenses strategies		
Specified	75.0%	[25, 54, 86–89, 92, 123, 129, 143, 144, 190]
Not Specified	25.0%	[29, 75, 91, 135]

3.6.2 Adversarial learning attacks and protections. Deep learning models are not resistant to adversarial attacks, which can cause a model to output an entirely incorrect prediction. Adversarial examples can be generated by just applying minor but intentional perturbations to original samples. A detailed taxonomy of adversarial sample crafting techniques and defensive techniques can be found in these survey work [2, 115]. As discussed before, we discovered 16 primary studies (12%) related to adversarial learning attacks and protections. Table 10 presents a detailed summary for these 16 primary studies.

Focusing on the target phase of adversarial attacks, it is apparent that evasion attacks receive more research attention, accounting for 87.5%. Evasion attacks modify the data point at inference time, resulting in misclassification. For example, Grosse et al. [54] perform adversarial evasion attacks on DNN-based malware detection models. This work exploited Jacobian based white-box attacks [115] to generate adversarial examples, and the evaluation results indicated that the evasion algorithm can misclassify 63% of all malware samples on Drebin. Conversely, only two primary studies [86, 135] investigated poisoning attacks, where the adversary's objective is to victimize the model training process. It is interesting to observe that both of these two sources are devoted to backdoor poisoning attacks.

Table 10 shows that five sources focus on white-box attacks. White-box attacks assume that the adversary has knowledge about the trained model such as model architectures and hyperparameters. For example, Grosse et al. [54] require the gradient information of DNN networks to craft adversarial examples. Five sources perform black-box attacks on DL-based malware detection models, where the adversary requires no knowledge about the target classifier. Six primary studies also assess the robustness of DNN models in both black-box and white-box scenarios. These studies demonstrated that the attackers are slightly more vulnerable when having a limited knowledge of the model architecture.

With regard to adversarial defense strategies, Table 10 indicates that 75% of sources adopt defense mechanisms for adversarial attacks. It is worth mentioning that most studies apply adversarial training and ensemble learning to defend against adversarial attacks [54, 87, 88, 123, 172, 190].

3.6.3 Malware evolution detection and defense. In the security domain, malware evolution has several similar concepts such as concept drift [178], time decay [119] and model aging [187]. Fig. 2 shows that seven recent sources for detecting and defending malware evolution have been found. In order to better understand the current research state, Table 11 provides a summary of these seven primary studies. It is worth highlighting that four primary studies [37, 85, 174, 187] attempt to capture features' semantic similarity to slow down model aging. Pendlebury et al. [119] propose a time-aware performance metric for measuring classifiers' resilience to malware evolution. Indeed, these approaches merely slow down model performance degradation caused by

Table 11. A summary of DL-based malware evolution detection and protection approaches

Tool	Year	DNN models	Approach	Model updating
TESSERACT [119]	2019	MLP	Proposing a new metric for time decay	Retraining without drift understanding, active learning, classification with rejection
EveDroid [85]	2019	MLP	API semantics	-
API-GRATH [187]	2020	MLP	API semantics	Active learning
Dr.Droid [37]	2021	Transformers, GNN	Semantic relations; Heterogeneous temporal graph	-
Li et al. [89]	2021	MLP, CNN, RNN	Uncertainty	-
SDAC [174]	2020	DNN	API semantics	-
CADE [178]	2021	AE	Concept drift detection	Retraining with drift understanding

malware evolution. Thus, model updating approaches like model retraining or active learning are also frequently investigated to reverse and improve obsolete models [119, 187]. However, such model updating approaches remain evolution-insensitive, requiring periodical retraining. In addition, this process often needs significant amounts of effort in labeling new samples. To this end, Yang et al. employ contrastive learning to identify and understand drift malware samples before updating the aging models.

RQ3 What are the emerging and potential research trends?

- Although there exists much research on DL-based on Android malware defenses, this topic still requires more in-depth analysis.
- Malware evolution and adversarial attacks are two recent hot topics.
- How to improve the reliability, robustness and practicability of DL-based Android malware defense frameworks is a future challenge.

4 OPEN ISSUES AND FUTURE TRENDS

In this work, we summarized the relevant sources of DL-based Android malware defenses and discussed research trends and challenges from various aspects in Section 3. Here we draw on these findings of this systematic review to provide a set of discussion points around the research and practice of Android security for future researchers.

Android malware defenses remain a hot topic for further investigation. As our systematic review revealed, much research has been devoted to DL-based Android malware defenses in recent years, and the amount of relevant research is constantly increasing. It suggests that mobile security is a major concern nowadays. Mobile phones have become an integral part of people's daily life, and mobile users also pay much more attention to private mobile security, particularly the prevention of malicious applications.

However, the majority of existing research focuses on malware detection as a binary classification problem, which is far from enough to address current issues and improve mobile security. Malware remains among the most effective threats in the cyber space, and malware writers continue to update malware techniques to bypass security detection. As a result, this research requires more in-depth analysis rather than simply seeking a binary label. Other research aspects, like malware attribution/behaviors, malware variants, malware triage and treatments of infection, still receive scant attention.

Data imbalance. As shown in Section 3.5.1, the sources prefer to construct a relatively balanced dataset for performance evaluation purposes. However, Android malware defenses suffer from serious sample sparsity and imbalance issues. In the Android landscape, the number of goodware is significantly greater than the number of malware [119]. Additionally, malware families are also highly imbalanced (thousands of samples in some families but only a few in others). Many previous studies have demonstrated that imbalanced data distributions hinder the performance [16, 119]. Thus, how to develop effective solutions against data imbalance in DL-based

Android malware defenses has gained immense research interest. From our reviewed results, we identified two related studies to handle the data imbalance issue, including a Siamese network-based approach for imbalanced family classification [16] and a BERT-based approach for imbalanced malware detection [112]. However, these two studies are limited to relatively simple scenarios, and thus more efforts are still required to overcome the negative influences of data imbalance on Android malware defenses.

Improving practicality and reliability is a priority. Our review also revealed that improving the practicality and reliability of DL-based Android malware defense approaches gathered increasing research interests. Although advanced deep learning techniques have been demonstrated to be effective at defending against malware attacks in a series of research experiments, how to effectively apply these approaches in practice remains unsolved. Future research should not only find a solution to overcome the limitation of mobile computing resources to deploy DL-based malware defense architectures, but also propose a comprehensive framework to tackle many realistic challenges such as internet privacy protection and information updating, which may require knowledge in other specific areas of cyber security and computer internet. Except for that, the black-box nature of deep neural networks poses a serious barrier to implementing these proposed approaches in practice. How to improve the transparency of the malware defense process will be a noteworthy research topic in the future.

Deep learning in Android security is still in an early stage. Compared with other research domains, research towards deep learning in the Android security community seems to be relative singleness. First, supervised DNN networks are the most investigated, and these studies usually consider DNN networks with three to four layers in their proposed architectures. Secondly, more advanced deep learning approaches like reinforcement learning and online learning are covered by a minority of papers. Thirdly, most previous works adopt deep learning techniques on some simple tasks like binary Android malware classification. In fact, deep learning has made considerable achievements in computer vision and natural language processing. These advanced techniques have been demonstrated a powerful capability in solving many complex tasks [83]. For example, Wu et al. [167] introduce an attention mechanism to improve the interpretability of Android malware detection models. Thus, it is promising to apply advanced deep learning techniques to assist us in solving more complex and specific issues in Android security. At the same time, while supervised learning is dominant in the Android security domain, labeling data is time-consuming and requires substantial expertise. As described by deep learning textbook by Lecun et al. [83], unsupervised learning belongs to the future of deep learning. We encourage our fellow researchers to make more efforts on unsupervised deep learning in Android security to make more advancements in the future.

APK embedding is an important but untouched topic. Different from image or word information, Android apps are composed of multiple complex data. Android APK is a zip archive consisting of multiple files. Through reverse engineering, various types of features like permissions and opcode can be extracted for further analysis. In fact, deep learning still struggles to model these complex data modalities [125]. Therefore, existing research either transforms APK into a single type of feature or designs a multimodel deep learning architecture to handle them. To obtain a formalized representation compatible with DL models, embedding techniques from CV and NLP are introduced to encode features, but these techniques may be relatively shallow for APK files with complicated structures. As for Android security, there is still a long way to explore "APK embedding" for DNN models.

5 THREATS TO VALIDITY

Even though this systematic review was conducted by following a well-established methodology [79], we can't guarantee that our results covered all relevant studies, caused by some limitations during the review process. Thus, this section describes possible threats to the validity of our empirical study.

Search items and strategies. One main potential threat is relevant publication collection bias. In order to locate relevant studies, we described a list of search strings and search databases in Section 2.2. Search strings were formulated by different items from both software engineering and artificial intelligence domains. Although we added alternative spellings and synonyms for search items, we may still miss some search items. For instance, deep learning is a rapidly developing research field and AI scientists would continue to propose new DL techniques in a short time; thus, identifying all relevant DL items is a challenge. To minimize this issue, we retained the DL items described in [51, 83, 134] surveyed by AI experts. After search strings were identified, five well-known electronic databases were used to collect relevant studies. We also conducted further searching processes on two popular research citation engines and a backward snowballing to cover the relevant publications in the broadest sense.

Data selection bias. The selection of publications was conducted by one researcher only, which may cause missing studies. Despite that, all authors formulated an appropriate study selection scheme together, and the other three authors gave effective and detailed feedback and monitored review execution closely throughout the review process. On the other hand, in order to reduce the impact of subjective factors in the quality evaluation process, H5-index of the venues was introduced as a quality appraisal criterion. Although the H5-value may change over time, the influential papers from the top venues were guaranteed to be taken into account.

6 CONCLUSIONS

This article provides an in-depth examination of the use of deep learning in Android malware defenses. Additionally, the study discusses research objectives, characteristics, approaches, and challenges associated with Android malware defenses using deep learning. We collected 132 relevant studies. Our reviewed results indicate that deep learning techniques are becoming a powerful and promising tool to defend against Android malicious applications. We discovered that (1) most studies are conducted to detect malware, but other types of more detailed analysis on malicious apps are receiving increasing attention; (2) static program analysis is widely used to collect features, and semantic features are frequently occurring; (3) various DNN architectures are employed to analyze malware, among which MLPs and CNNs are the most widely used; (4) most approaches are performed as a supervised classification task; (5) distributed analysis and on-device analysis is gradually valued; (6) adversarial learning and malware evolution are two recent hot topics.

REFERENCES

[1] Abada Abderrahmane, Guettaf Adnane, Challal Yacine, and Garri Khireddine. 2019. Android Malware Detection Based on System Calls Analysis and CNN Classification. In *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, 1–6.

[2] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[3] Ebtesam J Alqahtani, Rachid Zagrouba, and Abdullah Almuhaideb. 2019. A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms.. In *2019 Sixth International Conference on Software Defined Systems (SDS)*. IEEE, 110–117.

[4] Hani Alshahrani, Harrison Mansourt, Seaver Thorn, Ali Alshehri, Abdulrahman Alzahrani, and Huirong Fu. 2018. DDefender: Android application threat detection using static and dynamic analysis. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 1–6.

[5] Mohammed K Alzaylaee, Suleiman Y Yerima, and Sakir Sezer. 2017. Emulator vs real phone: Android malware detection using machine learning. In *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics*. 65–72.

[6] Mohammed K Alzaylaee, Suleiman Y Yerima, and Sakir Sezer. 2020. DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security* 89 (2020), 101663.

[7] Muhammad Amin, Babar Shah, Aizaz Sharif, Tamleek Ali, Ki-lL Kim, and Sajid Anwar. 2019. Android malware detection through generative adversarial networks. *Transactions on Emerging Telecommunications Technologies* (2019), e3675.

[8] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*. 173–182.

[9] A Ananya, A Aswathy, TR Amal, PG Swathy, P Vinod, and Shojafar Mohammad. 2020. SysDroid: a dynamic ML-based android malware analyzer using system call traces. *Cluster Computing* (2020), 1–20.

[10] AndroZoo 2020. *AndroZoo*. Retrieved Oct 11, 2020 from https://androzoo.uni.lu/

[11] Apktool 2010. *APKTOOL*. Retrieved Oct 25, 2021 from https://ibotpeaches.github.io/Apktool/
[12] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2020. Dos and Don'ts of Machine Learning in Computer Security. *arXiv preprint arXiv:2010.09470* (2020).
[13] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket.. In *Ndss*, Vol. 14. 23–26.
[14] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.
[15] Alessandro Bacci, Alberto Bartoli, Fabio Martinelli, Eric Medvet, and Francesco Mercaldo. 2018. Detection of obfuscation techniques in Android applications. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*. 1–9.
[16] Yude Bai, Zhenchang Xing, Xiaohong Li, Zhiyong Feng, and Duoyuan Ma. 2020. Unsuccessful story about few shot malware family classification and siamese network to the rescue. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1560–1571.
[17] Yude Bai, Zhenchang Xing, Duoyuan Ma, Xiaohong Li, and Zhiyong Feng. 2021. Comparative analysis of feature representations and machine learning methods in Android family classification. *Computer Networks* 184 (2021), 107639.
[18] Nazanin Bakhshinejad and Ali Hamzeh. 2019. Parallel-CNN network for malware detection. *IET Information Security* 14, 2 (2019), 210–219.
[19] Khaled Bakour and Halil Murat Ünver. 2020. VisDroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques. *Neural Computing and Applications* (2020), 1–21.
[20] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[21] BlackHat 2011. *Androguard*. Retrieved Oct 25, 2021 from https://code.google.com/archive/p/androguard
[22] Marcus Botacin, Fabricio Ceschin, Ruimin Sun, Daniela Oliveira, and André Grégio. 2021. Challenges and pitfalls in malware research. *Computers & Security* 106 (2021), 102287.
[23] Dewan Chaulagain, Prabesh Poudel, Prabesh Pathak, Sankardas Roy, Doina Caragea, Guojun Liu, and Xinming Ou. 2020. Hybrid Analysis of Android Apps for Security Vetting using Deep Learning. In *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.
[24] Simin Chen, Soroush Bateni, Sampath Grandhi, Xiaodi Li, Cong Liu, and Wei Yang. 2020. DENAS: automated rule generation by knowledge extraction from neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 813–825.
[25] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. 2019. Android HIV: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security* 15 (2019), 987–1001.
[26] Gianni D'Angelo, Massimo Ficco, and Francesco Palmieri. 2020. Malware detection in mobile environments based on Autoencoders and API-images. *J. Parallel and Distrib. Comput.* 137 (2020), 26–33.
[27] Nadia Daoudi, Kevin Allix, Tegawendé F Bissyandé, and Jacques Klein. 2021. Lessons Learnt on Reproducibility in Machine Learning Based Android Malware Detection. *Empirical Software Engineering* 26, 4 (2021), 1–53.
[28] Nadia Daoudi, Jordan Samhi, Abdoul Kader Kabore, Kevin Allix, Tegawendé F Bissyandé, and Jacques Klein. 2021. DexRay: A Simple, yet Effective Deep Learning Approach to Android Malware Detection Based on Image Representation of Bytecode. In *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 81–106.
[29] Asim Darwaish, Farid Naït-Abdesselam, Chafiq Titouna, and Sumera Sattar. 2021. Robustness of image-based android malware detection under adversarial attacks. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.
[30] Andrea De Lorenzo, Fabio Martinelli, Eric Medvet, Francesco Mercaldo, and Antonella Santone. 2020. Visualizing the outcome of dynamic analysis of Android malware with VizMal. *Journal of Information Security and Applications* 50 (2020), 102423.
[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[32] Yuxin Ding, Xiao Zhang, Jieke Hu, and Wenting Xu. 2020. Android malware detection method based on bytecode image. *Journal of Ambient Intelligence and Humanized Computing* (2020), 1–10.
[33] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé, Tianming Liu, Guoai Xu, and Jacques Klein. 2018. FraudDroid: Automated Ad Fraud Detection for Android Apps. In *The 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*.
[34] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.
[35] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 1–29.

[36] Ming Fan, Wenying Wei, Xiaofei Xie, Yang Liu, Xiaohong Guan, and Ting Liu. 2020. Can We Trust Your Explanations? Sanity Checks for Interpreters in Android Malware Analysis. *IEEE Transactions on Information Forensics and Security* (2020).

[37] Yujie Fan, Mingxuan Ju, Shifu Hou, Yanfang Ye, Wenqiang Wan, Kui Wang, Yinming Mei, and Qi Xiong. 2021. Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2831–2839.

[38] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur, Mauro Conti, and Muttukrishnan Rajarajan. 2014. Android security: a survey of issues, malware penetration, and defenses. *IEEE communications surveys & tutorials* 17, 2 (2014), 998–1022.

[39] Johannes Feichtner and Stefan Gruber. 2020. Understanding Privacy Awareness in Android App Descriptions Using Deep Learning. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. 203–214.

[40] Ruitao Feng, Sen Chen, Xiaofei Xie, Lei Ma, Guozhu Meng, Yang Liu, and Shang-Wei Lin. 2019. Mobidroid: A performance-sensitive malware detection system on mobile platform. In *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 61–70.

[41] Ruitao Feng, Sen Chen, Xiaofei Xie, Guozhu Meng, Shang-Wei Lin, and Yang Liu. 2020. A performance-sensitive malware detection system using deep learning on mobile devices. *IEEE Transactions on Information Forensics and Security* 16 (2020), 1563–1578.

[42] Yinglan Feng, Liang Chen, Angyu Zheng, Cuiyun Gao, and Zibin Zheng. 2019. AC-Net: Assessing the Consistency of Description and Permission in Android Apps. *IEEE Access* 7 (2019), 57829–57842.

[43] Hossein Fereidooni, Mauro Conti, Danfeng Yao, and Alessandro Sperduti. 2016. ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications. In *2016 8th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE, 1–5.

[44] Massimo Ficco. 2021. Malware Analysis By Combining Multiple Detectors and Observation Windows. *IEEE Trans. Comput.* (2021).

[45] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. 2018. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560* (2018).

[46] Rafa Gálvez, Veelasha Moonsamy, and Claudia Diaz. 2021. Less is More: A privacy-respecting Android malware classifier using Federated Learning. *Proceedings on Privacy Enhancing Technologies* 1 (2021), 20.

[47] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. 2014. Malware analysis and classification: A survey. *Journal of Information Security* 2014 (2014).

[48] Han Gao, Shaoyin Cheng, and Weiming Zhang. 2021. GDroid: Android malware detection and classification with graph convolutional network. *Computers & Security* 106 (2021), 102264.

[49] Amirhossein Gharib and Ali Ghorbani. 2017. Dna-droid: A real-time android ransomware detection framework. In *International Conference on Network and System Security*. Springer, 184–198.

[50] Liangyi Gong, Zhenhua Li, Feng Qian, Zifan Zhang, Qi Alfred Chen, Zhiyun Qian, Hao Lin, and Yunhao Liu. 2020. Experiences of landing machine learning onto market-scale mobile malware detection. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–14.

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[52] Google Play Protect 2020. *Google Play Protect*. Retrieved Sep 09, 2020 from https://www.android.com/play-protect/

[53] Petr Gronát, Javier Alejandro Aldana-Iuit, and Martin Bálek. 2019. MaxNet: Neural Network Architecture for Continuous Detection of Malicious Activity. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 28–35.

[54] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*. Springer, 62–79.

[55] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[56] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820* (2018).

[57] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42.

[58] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 364–379.

[59] Ke He and Dong-Seong Kim. 2019. Malware detection with malware images using deep learning techniques. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 95–102.

[60] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29, 6 (2012), 82–97.

[61] Geoffrey E Hinton. 2009. Deep belief networks. *Scholarpedia* 4, 5 (2009), 5947.

[62] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[63] Shifu Hou, Aaron Saas, Lifei Chen, and Yanfang Ye. 2016. Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. IEEE, 104–111.

[64] Shifu Hou, Aaron Saas, Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Deep neural networks for automatic android malware detection. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 803–810.

[65] Shifu Hou, Aaron Saas, Yanfang Ye, and Lifei Chen. 2016. Droiddelver: An android malware detection system using deep belief network based on api call blocks. In *International Conference on Web-Age Information Management*. Springer, 54–66.

[66] TonTon Hsien-De Huang and Hung-Yu Kao. 2018. R2-D2: color-inspired convolutional neural network (CNN)-based android malware detections. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2633–2642.

[67] Na Huang, Ming Xu, Ning Zheng, Tong Qiao, and Kim-Kwang Raymond Choo. 2019. Deep Android Malware Classification with API-Based Feature Graph. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 296–303.

[68] Shinelle Hutchinson, Bing Zhou, and Umit Karabiyik. 2019. Are We Really Protected? An Investigation into the Play Protect Service. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 4997–5004.

[69] Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2021. Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security* 105 (2021), 102198.

[70] Amir Namavar Jahromi, Sattar Hashemi, Ali Dehghantanha, Reza M Parizi, and Kim-Kwang Raymond Choo. 2020. An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020).

[71] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Hoa Khanh Dam, and John Grundy. 2020. An Empirical Study of Model-Agnostics Techniques for Defect Prediction Models. *IEEE Transactions on Software Engineering (TSE)* (2020).

[72] ElMouatez Billah Karbab and Mourad Debbabi. 2021. PetaDroid: Adaptive Android Malware Detection Using Deep Learning. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 319–340.

[73] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. 2018. MalDozer: Automatic framework for android malware detection using deep learning. *Digital Investigation* 24 (2018), S48–S59.

[74] Naveen Karunanayake, Jathushan Rajasegaran, Ashanie Gunathillake, Suranga Seneviratne, and Guillaume Jourjon. 2020. A Multi-modal Neural Embeddings Approach for Detecting Mobile Counterfeit Apps: A Case Study on Google Play Store. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[75] Mahbub Khoda, Tasadduq Imam, Joarder Kamruzzaman, Iqbal Gondal, and Ashfaqur Rahman. 2019. Selective adversarial learning for mobile malware. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 272–279.

[76] M. E. Khoda, J. Kamruzzaman, I. Gondal, T. Imam, and A. Rahman. 2019. Mobile Malware Detection: An Analysis of Deep Learning Model. In *2019 IEEE International Conference on Industrial Technology (ICIT)*. 1161–1166.

[77] TaeGuen Kim, BooJoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14, 3 (2018), 773–788.

[78] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[79] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.

[80] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[81] Rahul Krishna and Tim Menzies. 2020. Learning Actionable Analytics from Multiple Software Projects. *Empirical Software Engineering (EMSE)* 25, 5 (2020), 3468–3500.

[82] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[83] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[84] William Younghoo Lee, Joshua Saxe, and Richard Harang. 2019. SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks. In *Deep Learning Applications for Cyber Security*. Springer, 197–210.

[85] Tao Lei, Zhan Qin, Zhibo Wang, Qi Li, and Dengpan Ye. 2019. EveDroid: Event-aware Android malware detection against model degrading for IoT devices. *IEEE Internet of Things Journal* 6, 4 (2019), 6668–6680.

[86] Chaoran Li, Xiao Chen, Derui Wang, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. 2021. Backdoor Attack on Machine Learning Based Android Malware Detectors. *IEEE Transactions on Dependable and Secure Computing* (2021).

[87] Deqiang Li and Qianmu Li. 2020. Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3886–3900.

[88] Deqiang Li, Qianmu Li, Yanfang Ye, and Shouhuai Xu. 2021. A framework for enhancing deep neural networks against adversarial malware. *IEEE Transactions on Network Science and Engineering* 8, 1 (2021), 736–750.

[89] Deqiang Li, Tian Qiu, Shuo Chen, Qianmu Li, and Shouhuai Xu. 2021. Can We Leverage Predictive Uncertainty to Detect Dataset Shift and Adversarial Examples in Android Malware Detection?. In *Annual Computer Security Applications Conference*.
[90] Dan Li, Lichao Zhao, Qingfeng Cheng, Ning Lu, and Wenbo Shi. 2020. Opcode sequence analysis of Android malware by a convolutional neural network. *Concurrency and Computation: Practice and Experience* 32, 18 (2020), e5308.
[91] Heng Li, ShiYao Zhou, Wei Yuan, Jiahuan Li, and Henry Leung. 2019. Adversarial-example attacks toward android malware detection system. *IEEE Systems Journal* 14, 1 (2019), 653–656.
[92] Heng Li, Shiyao Zhou, Wei Yuan, Xiapu Luo, Cuiying Gao, and Shuiyan Chen. 2021. Robust Android Malware Detection against Adversarial Example Attacks. In *Proceedings of the Web Conference 2021*. 3603–3612.
[93] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick Mcdaniel. 2015. IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. In *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*.
[94] Li Li, Tegawendé F Bissyandé, and Jacques Klein. 2019. Rebooting research on detecting repackaged android apps: Literature review and benchmark. *IEEE Transactions on Software Engineering* 47, 4 (2019), 676–693.
[95] Li Li, Tegawendé F Bissyandé, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, and Le Traon. 2017. Static analysis of android apps: A systematic literature review. *Information and Software Technology* 88 (2017), 67–95.
[96] Li Li, Daoyuan Li, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, David Lo, and Lorenzo Cavallaro. 2017. Understanding Android App Piggybacking: A Systematic Study of Malicious Code Grafting. *IEEE Transactions on Information Forensics & Security (TIFS)* (2017).
[97] Li Li, Timothée Riom, Tegawendé F Bissyandé, Haoyu Wang, Jacques Klein, and Yves Le Traon. 2019. Revisiting the Impact of Common Libraries for Android-related Investigations. *Journal of Systems and Software (JSS)* (2019).
[98] Yang Li and Tao Yang. 2018. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*. Springer, 83–104.
[99] Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu. 2020. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* 8 (2020), 124579–124607.
[100] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé F Bissyandé, and Jacques Klein. 2020. MadDroid: Characterising and Detecting Devious Ad Content for Android Apps. In *The Web Conference 2020 (WWW 2020)*.
[101] Tianliang Lu, Yanhui Du, Li Ouyang, Qiuyu Chen, and Xirui Wang. 2020. Android Malware Detection Based on a Hybrid Deep Learning Model. *Security and Communication Networks* 2020 (2020).
[102] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*. 4765–4774.
[103] Haoyu Ma, Jianwen Tian, Kefan Qiu, David Lo, Debin Gao, Daoyuan Wu, Chunfu Jia, and Thar Baker. 2020. Deep-learning–based app sensitive behavior surveillance for Android powered cyber–physical systems. *IEEE Transactions on Industrial Informatics* 17, 8 (2020), 5840–5850.
[104] E Mariconti, L Onwuzurike, P Andriotis, E De Cristofaro, G Ross, and G Stringhini. 2017. MamaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. 2017 (2017).
[105] Alejandro Martín, Félix Fuentes-Hurtado, Valery Naranjo, and David Camacho. 2017. Evolving deep neural networks architectures for android malware classification. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1659–1666.
[106] Niall McLaughlin, Jesus Martinez del Rincon, BooJoong Kang, Suleiman Yerima, Paul Miller, Sakir Sezer, Yeganeh Safaei, Erik Trickel, Ziming Zhao, Adam Doupé, et al. 2017. Deep android malware detection. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 301–308.
[107] Francesco Mercaldo and Antonella Santone. 2020. Deep learning for image-based mobile malware detection. *Journal of Computer Virology and Hacking Techniques* (2020), 1–15.
[108] Stuart Millar, Niall McLaughlin, Jesus Martinez del Rincon, Paul Miller, and Ziming Zhao. 2020. DANdroid: A Multi-View Discriminative Adversarial Network for Obfuscated Android Malware Detection. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. 353–364.
[109] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu. com.
[110] Abdelmonim Naway and Yuancheng Li. 2018. A review on the use of deep learning in android malware detection. *arXiv preprint arXiv:1812.10360* (2018).
[111] Robin Nix and Jian Zhang. 2017. Classification of android apps and malware using deep neural networks. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1871–1878.
[112] Rajvardhan Oak, Min Du, David Yan, Harshvardhan Takawale, and Idan Amit. 2019. Malware Detection on Highly Imbalanced Data through Sequence Modeling. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 37–48.
[113] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–48.

[114] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1105–1114.

[115] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[116] Xinjun Pei, Long Yu, Shengwei Tian, Huanhuan Wang, and Yongfang Peng. 2020. Combining multi-features with a neural joint model for Android malware detection 1. *Journal of Intelligent & Fuzzy Systems* Preprint (2020), 1–11.

[117] Abdurrahman Pektaş and Tankut Acarman. 2020. Deep learning for effective Android malware detection using API call graph embeddings. *Soft Computing* 24, 2 (2020), 1027–1043.

[118] Abdurrahman Pektaş and Tankut Acarman. 2020. Learning to detect Android malware via opcode sequences. *Neurocomputing* 396 (2020), 599–608.

[119] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 729–746.

[120] Kewen Peng and Tim Menzies. 2020. Defect Reduction Planning (using TimeLIME). *arXiv preprint arXiv:2006.07416* (2020).

[121] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[122] Fabio Pierazzi, Ghita Mezzour, Qian Han, Michele Colajanni, and VS Subrahmanian. 2020. A Data-driven Characterization of Modern Android Spyware. *ACM Transactions on Management Information Systems (TMIS)* 11, 1 (2020), 1–38.

[123] Robert Podschwadt and Hassan Takabi. 2019. On Effectiveness of Adversarial Examples and Defenses for Malware Classification. In *International Conference on Security and Privacy in Communication Systems*. Springer, 380–393.

[124] Chanathip Pornprasit and Chakkrit Tantithamthavorn. 2021. JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction. In *Proceedings of the International Conference on Mining Software Repositories (MSR)*. To Appear.

[125] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and SS Iyengar. 2018. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)* 51, 5 (2018), 1–36.

[126] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, Yu Wang, and Yang Xiang. 2019. A3CM: automatic capability annotation for android malware. *IEEE Access* 7 (2019), 147156–147168.

[127] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, and Yang Xiang. 2020. A survey of Android malware detection with deep neural models. *ACM Computing Surveys (CSUR)* 53, 6 (2020), 1–36.

[128] Dilini Rajapaksha, Chakkrit Tantithamthavorn, Christoph Bergmeir, Wray Buntine, Jirayus Jiarpakdee, and John Grundy. 2021. SQAPlanner: Generating data-informed software quality improvement plans.

[129] Hemant Rathore, Sanjay K Sahay, Piyush Nikam, and Mohit Sewak. 2021. Robust android malware detection system against adversarial attacks using q-learning. *Information Systems Frontiers* 23, 4 (2021), 867–882.

[130] Zhongru Ren, Haomin Wu, Qian Ning, Iftikhar Hussain, and Bingcai Chen. 2020. End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Networks* 101 (2020), 102098.

[131] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[132] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations.. In *AAAI*, Vol. 18. 1527–1535.

[133] Justin Sahs and Latifur Khan. 2012. A machine learning approach to android malware detection. In *2012 European Intelligence and Security Informatics Conference*. IEEE, 141–147.

[134] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85 – 117. https://doi.org/10.1016/j.neunet.2014.09.003

[135] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[136] Lwin Khin Shar, Biniam Fisseha Demissie, Mariano Ceccato, and Wei Minn. 2020. Experimental comparison of features and classifiers for Android malware detection. (2020).

[137] Alireza Souri and Rahil Hosseini. 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-centric Computing and Information Sciences* 8, 1 (2018), 3.

[138] Statista 2020. *Mobile operating systems' market share worldwide from January 2012 to July 2020*. Retrieved Sep 09, 2020 from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[139] Xin Su, Weiqi Shi, Xilong Qu, Yi Zheng, and Xuchong Liu. 2020. DroidDeep: using Deep Belief Network to characterize and detect android malware. *Soft Computing* (2020), 1–14.

[140] Xin Su, Dafang Zhang, Wenjia Li, and Kai Zhao. 2016. A deep learning approach to android malware feature learning and detection. In *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 244–251.

[141] Yuxia Sun, Yanjia Chen, Yuchang Pan, and Lingyu Wu. 2019. Android Malware Family Classification Based on Deep Learning of Code Images. *IAENG International Journal of Computer Science* 46, 4 (2019).
[142] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
[143] Rahim Taheri, Reza Javidan, and Zahra Pooranian. 2020. Adversarial android malware detection for mobile multimedia applications in IoT environments. *Multimedia Tools and Applications* (2020), 1–17.
[144] Rahim Taheri, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti. 2020. On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications* (2020), 1–20.
[145] Kimberly Tam, Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, and Lorenzo Cavallaro. 2017. The evolution of android malware and android analysis techniques. *ACM Computing Surveys (CSUR)* 49, 4 (2017), 1–41.
[146] Kimberly Tam, Salahuddin J Khan, Aristide Fattori, and Lorenzo Cavallaro. 2015. Copperdroid: automatic reconstruction of android malware behaviors.. In *Ndss*.
[147] Xinrui Tan, Hongjia Li, Liming Wang, and Zhen Xu. 2020. End-Edge Coordinated Inference for Real-Time BYOD Malware Detection using Deep Learning. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–6.
[148] Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, and John Grundy. 2020. Explainable AI for Software Engineering. *arXiv preprint arXiv:2012.01614* (2020).
[149] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. 2019. Survey of machine learning techniques for malware analysis. *Computers & Security* 81 (2019), 123–147.
[150] Farhan Ullah, Hamad Naeem, Muhammad Rashid Naeem, Sohail Jabbar, Shehazad Khalid, Fadi Al-Turjman, and Abdelrahman Abuarqoub. 2019. Detection of clone scammers in Android markets using IoT-based edge computing. *Transactions on Emerging Telecommunications Technologies* (2019), e3791.
[151] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. 2020. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks* 171 (2020), 107138.
[152] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[153] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. 2017. Deep android malware detection and classification. In *2017 International conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 1677–1683.
[154] R Vinayakumar, KP Soman, Prabaharan Poornachandran, and S Sachin Kumar. 2018. Detecting Android malware using long short-term memory (LSTM). *Journal of Intelligent & Fuzzy Systems* 34, 3 (2018), 1277–1288.
[155] VirusShare.com 2020. *Because Sharing is Caring.* Retrieved Oct 11, 2020 from https://virusshare.com/
[156] Xiaoyue Wan, Geyi Sheng, Yanda Li, Liang Xiao, and Xiaojiang Du. 2017. Reinforcement learning based mobile offloading for cloud-based malware detection. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 1–6.
[157] Ji Wang, Qi Jing, Jianbo Gao, and Xuanwei Qiu. 2020. SEdroid: A Robust Android Malware Detector using Selective Ensemble Learning. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–5.
[158] Shanshan Wang, Zhenxiang Chen, Qiben Yan, Ke Ji, Lin Wang, Bo Yang, and Mauro Conti. 2018. Deep and broad learning based detection of android malware via network traffic. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–6.
[159] Wei Wang, Mengxue Zhao, and Jigang Wang. 2019. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing* 10, 8 (2019), 3035–3043.
[160] Yuxuan Wang, Yutai Hou, Wanxiang Che, and Ting Liu. 2020. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics* (2020), 1–20.
[161] Zi Wang, Juecong Cai, Sihua Cheng, and Wenjia Li. 2016. DroidDeepLearner: Identifying Android malware using deep learning. In *2016 IEEE 37th Sarnoff Symposium*. IEEE, 160–165.
[162] Zhiqiang Wang, Qian Liu, and Yaping Chi. 2020. Review of Android Malware Detection Based on Deep Learning. *IEEE Access* (2020).
[163] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. 2020. Evaluating explanation methods for deep learning in security. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 158–174.
[164] Supatsara Wattanakriengkrai, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Hideaki Hata, and Kenichi Matsumoto. 2020. Predicting Defective Lines Using a Model-Agnostic Technique. *IEEE Transactions on Software Engineering (TSE)* (2020).
[165] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. 2017. Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 252–276.
[166] Lilian Weng. 2018. Attention? attention. *Lil'Log, June* 24 (2018).
[167] Bozhi Wu, Sen Chen, Cuiyun Gao, Lingling Fan, Yang Liu, Weiping Wen, and Michael R Lyu. 2021. Why an Android App Is Classified as Malware: Toward Malware Classification Interpretation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–29.

[168] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. 2012. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*. IEEE, 62–69.

[169] Huanyu Wu. 2020. A Systematical Study for Deep Learning Based Android Malware Detection. In *Proceedings of the 2020 9th International Conference on Software and Computer Applications*. 177–182.

[170] Shengqu Xi, Shao Yang, Xusheng Xiao, Yuan Yao, Yayuan Xiong, Fengyuan Xu, Haoyu Wang, Peng Gao, Zhuotao Liu, Feng Xu, et al. 2019. DeepIntent: Deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2421–2436.

[171] Xusheng Xiao. 2019. An image-inspired and CNN-based Android malware detection approach. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1259–1261.

[172] Xi Xiao, Shaofeng Zhang, Francesco Mercaldo, Guangwu Hu, and Arun Kumar Sangaiah. 2019. Android malware detection based on system call sequences and LSTM. *Multimedia Tools and Applications* 78, 4 (2019), 3979–3999.

[173] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.

[174] Jiayun Xu, Yingjiu Li, Robert Deng, and Ke Xu. 2020. SDAC: A Slow-aging solution for Android malware detection using semantic distance based API clustering. *IEEE Transactions on Dependable and Secure Computing* (2020).

[175] Ke Xu, Yingjiu Li, Robert H Deng, and Kai Chen. 2018. Deeprefiner: Multi-layer android malware detection system applying deep neural networks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 473–487.

[176] Lifan Xu, Dongping Zhang, Nuwan Jayasena, and John Cavazos. 2016. Hadm: Hybrid analysis for detection of malware. In *Proceedings of SAI Intelligent Systems Conference*. Springer, 702–724.

[177] Jinpei Yan, Yong Qi, and Qifan Rao. 2018. LSTM-based hierarchical denoising network for Android malware detection. *Security and Communication Networks* 2018 (2018).

[178] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. 2021. {CADE}: Detecting and Explaining Concept Drift Samples for Security Applications. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[179] Yanfang Ye, Shifu Hou, Lingwei Chen, Jingwei Lei, Wenqiang Wan, Jiabin Wang, Qi Xiong, and Fudong Shao. 2019. Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection.. In *IJCAI*. 4150–4156.

[180] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–40.

[181] Yao-Saint Yen and Hung-Min Sun. 2019. An android mutation malware detection based on deep learning using visualization of importance from codes. *Microelectronics Reliability* 93 (2019), 109–114.

[182] Baoguo Yuan, Junfeng Wang, Dong Liu, Wen Guo, Peng Wu, and Xuhua Bao. 2020. Byte-level malware classification based on markov images and deep learning. *Computers & Security* 92 (2020), 101740.

[183] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* 30, 9 (2019), 2805–2824.

[184] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. 2014. Droid-sec: deep learning in android malware detection. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 371–372.

[185] Zhenlong Yuan, Yongqiang Lu, and Yibo Xue. 2016. Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology* 21, 1 (2016), 114–123.

[186] Xian Zhan, Lingling Fan, Tianming Liu, Sen Chen, Li Li, Haoyu Wang, Yifei Xu, Xiapu Luo, and Yang Liu. 2020. Automated Third-party Library Detection for Android Applications: Are We There Yet?. In *The 35th IEEE/ACM International Conference on Automated Software Engineering (ASE 2020)*.

[187] Xiaohan Zhang, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min 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*. 757–770.

[188] Yanxin Zhang, Yulei Sui, Shirui Pan, Zheng Zheng, Baodi Ning, Ivor Tsang, and Wanlei Zhou. 2019. Familial clustering For weakly-labeled Android malware using hybrid representation learning. *IEEE Transactions on Information Forensics and Security* 15 (2019), 3401–3414.

[189] Zicheng Zhang, Wenrui Diao, Chengyu Hu, Shanqing Guo, Chaoshun Zuo, and Li Li. 2020. An Empirical Study of Potentially Malicious Third-Party Libraries in Android Apps. In *The 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 2020)*.

[190] Kaifa Zhao, Hao Zhou, Yulin Zhu, Xian Zhan, Kai Zhou, Jianfeng Li, Le Yu, Wei Yuan, and Xiapu Luo. 2021. Structural Attack against Graph Based Android Malware Detection. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3218–3235.

[191] Yanjie Zhao, Li Li, Haoyu Wang, Haipeng Cai, Tegawende Bissyande, Jacques Klein, and John Grundy. 2021. On the Impact of Sample Duplication in Machine Learning based Android Malware Detection. *ACM Transactions on Software Engineering and Methodology (TOSEM)* (2021).

[192] XU Zhiwu, Kerong Ren, and Fu Song. 2019. Android malware family classification and characterization using CFG and DFG. In *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*. IEEE, 49–56.

[193] Hanxun Zhou, Xinlin Yang, Hong Pan, and Wei Guo. 2020. An Android Malware Detection Approach Based on SIMGRU. *IEEE Access* 8 (2020), 148404–148410.

[194] Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*. IEEE, 95–109.

[195] Dali Zhu, Hao Jin, Ying Yang, Di Wu, and Weiyi Chen. 2017. DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data. In *2017 IEEE symposium on computers and communications (ISCC)*. IEEE, 438–443.

[196] Dali Zhu, Yuchen Ma, Tong Xi, and Yiming Zhang. 2019. FSNet: Android Malware Detection with Only One Feature. In *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–6.

[197] Dali Zhu, Tong Xi, Pengfei Jing, Di Wu, Qing Xia, and Yiming Zhang. 2019. A Transparent and Multimodal Malware Detection Method for Android Apps. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 51–60.

[198] Dali Zhu, Tong Xi, Pengfei Jing, Qing Xia, Di Wu, and Yiming Zhang. 2020. Sadroid: A Deep Classification Model For Android Malware Detection Based On Semantic Analysis. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–7.

[199] Huijuan Zhu, Liangmin Wang, Sheng Zhong, Yang Li, and Victor S Sheng. 2021. A Hybrid Deep Network Framework for Android Malware Detection. *IEEE Transactions on Knowledge and Data Engineering* (2021).