Neural Computing and Applications (2022) 34:1007–1032
https://doi.org/10.1007/s00521-021-06597-0

REVIEW

# Applications of deep learning for mobile malware detection: A systematic literature review

Cagatay Catal[1] · Görkem Giray[2] · Bedir Tekinerdogan[3]

Received: 27 November 2020 / Accepted: 4 October 2021 / Published online: 23 October 2021

**Abstract**
For detecting and resolving the various types of malware, novel techniques are proposed, among which deep learning algorithms play a crucial role. Although there has been a lot of research on the development of DL-based mobile malware detection approaches, they were not reviewed in detail yet. This paper aims to identify, assess, and synthesize the reported articles related to the application of DL techniques for mobile malware detection. A Systematic Literature Review is performed in which we selected 40 journal articles for in-depth analysis. This SLR presents and categorizes these articles based on machine learning categories, data sources, DL algorithms, evaluation parameters & approaches, feature selection techniques, datasets, and DL implementation platforms. The study also highlights the challenges, proposed solutions, and future research directions on the use of DL in mobile malware detection. This study showed that Convolutional Neural Networks and Deep Neural Networks algorithms are the most used DL algorithms. API calls, Permissions, and System Calls are the most dominant features utilized. Keras and Tensorflow are the most popular platforms. Drebin and VirusShare are the most widely used datasets. Supervised learning and static features are the most preferred machine learning and data source categories.

**Keywords** Deep learning · Machine learning · Mobile applications · Malware detection · Systematic literature review (SLR) · Cybersecurity

## 1 Introduction

Malicious software (a.k.a., malware), such as worms, viruses, trojans, rootkits, spyware, backdoors, and botnets, deliberately cause harmful or unexpected effects on the computer systems. Cyber-criminals often use different kinds of malware to launch security attacks that can affect thousands of organizations and millions of users [72]. As such, many anti-malware software vendors such as Symantec, AVAST, ESET, Bitdefender, TrendMicro, McAfee, and cybersecurity researchers developed different malware detection techniques.

Malware threats are expanding not only vertically in terms of the number of incidents but also horizontally in terms of functionality and types of malware [26]. Recently, ransomware [38] and cryptojacking [13] became the most dominant malware categories, among others. In addition, malware addressing mobile [50, 60] and Internet of Things (IoT) systems [19] are on the rise. The key factor for the rise of mobile malware is associated with financial motivation. Since IoT systems are widely used in different application domains, they are also vulnerable to malware attacks. One of the most recent examples is the Mirai family of malware that infected IoT devices with a massive Distributed Denial of Service (DDoS) attacks in late 2016 [3, 39]. The latest mobile malware is related to mobile banking trojans, cryptocurrency mining, and ransomware [40]. Since a large number of mobile applications are developed and released every day, manual inspection of the behavior of these applications is not an option anymore,

✉ Cagatay Catal
ccatal@qu.edu.qa

Bedir Tekinerdogan
bedir.tekinerdogan@wur.nl

[1] Department of Computer Science and Engineering, Qatar University, Doha, Qatar

[2] Izmir, Turkey

[3] Information Technology Group, Wageningen University and Research, Wageningen, The Netherlands

and therefore, automated malware detection approaches emerged. Currently, there are nearly 3 million applications on Google Play Store [4], and the first half of 2020, around 300,000 new Android applications, were added to this store. For such a rapidly growing ecosystem, the need for automated malware detection approaches is just inevitable.

Ucci et al. [70] reviewed papers on the use of machine learning in malware analysis and presented a taxonomy of machine learning techniques for malware analysis based on the objective, feature, and machine learning algorithm dimensions. They divided the objective dimension into the following three categories:

i. *Malware detection* This objective investigates whether a given software is malicious or not. Most of the reviewed papers belong to this category.
ii. *Malware similarity analysis* This objective detects similarities among malware and can be divided into the following sub-categories:
   - o *Variants detection* Many new malicious software applications are the variants of the existing malware because reuse of the code and resources are beneficial for the attackers. The detection of variants reduces the workload of analysts.
   - o *Families detection* The objective is to detect the family of the given malware.
   - o *Similarities detection* The objective is to detect the similarities and differences between the given binary file and already analyzed ones. Researchers can focus on the new part and discard the similar ones.
   - o *Differences detection* This objective determines the differentiating novel aspects of the malware.
iii. *Malware category detection:* The objective is to recognize the category of malware. However, cyber-security firms do not have a standardized malware category taxonomy yet Ucci et al. [70].

For malware detection, three types of malware analysis are mostly performed [26]. The first one is the static analysis that does not execute the application for the analysis. On the other hand, with dynamic analysis, the behavior of the application is analyzed based on its execution. Although static analysis is faster, it usually fails to detect malware if, for example, code obfuscation approaches are applied by malware developers. There are several obfuscation techniques, namely encryption, oligomorphic, polymorphic, metamorphic, stealth, and packaging [5]. Encryption is used to hide malicious code in the code, oligomorphic applies a different key for encrypting and decrypting the payload of the malware, and polymorphic creates a myriad number of distinct decryptors. Metamorphic does not apply the encryption, and the opcode is changed when the malicious code is executed. Stealth (a.k.a., code protection) makes some changes in the system that it locates not to be detected by the detection systems and packaging compresses the malware or hide the actual code using the encryption [5]. Polymorphic malware and code obfuscation cannot help much to malware developers because dynamic analysis checks the dynamic behavior of the application [26]. The third category for malware detection can be considered as the integration of both static analysis and dynamic analysis, which is called hybrid analysis. In addition to these types of analysis approaches, some researchers also proposed malware visualization based on image processing techniques [53]. Researchers transformed binary samples into grayscale images, and the features of the images were used for the classification. After the visualization is performed, the malware detection problem can be evaluated as an image recognition problem [18].

In the early days of malware detection, signature-based approaches [28] were widely used. Hereby, signatures indicate short strings that are used to uniquely represent a certain program, and signature-based approaches used pattern-matching techniques to detect these strings. However, when a slight change such as encryption and obfuscation is introduced in the malicious code, signature-based approaches cannot detect them [74]. To overcome this limitation, heuristic-based malware detection approaches [7] using machine learning algorithms were proposed and evaluated in different benchmarking datasets. Different approaches that analyze the dynamic behavior of the applications [62], permissions [47], and the n-grams in the byte code [33] were developed, and however, they are dependent on the expert knowledge while designing the features of the machine learning models [50].

Recently, deep learning has provided promising results in many different domains, such as computer vision and natural language processing (i.e., image classification, face detection, text classification) [23, 48, 77]. Mahdavifar and Ghorbani [49] reviewed the deep learning approaches applied in cybersecurity and categorized the studies into the following main categories: malware detection and analysis (Android-based and PC-based studies), network intrusion detection, phishing detection, website defacement detection, and spam detection. They specified the following common limitations of deep learning approaches in cybersecurity: Parameters are not fine-tuned precisely, a sufficient number of evaluation parameters are not used for evaluation, datasets are small-scale and out-of-date, no time complexity analysis is performed, the selection of a particular deep learning model is not formally verified, and models cannot explain the logic behind the decision.

In parallel with the developments in deep learning, cybersecurity researchers started to develop deep learning-based techniques for malware detection for coping with the limitations of traditional approaches. For instance, McLaughlin et al. [50] used the disassembled bytecode of the application as textual data and applied Convolutional Neural Networks (CNN) for malware detection. Apart from the high-performance benefit of deep learning algorithms, another motivation for the researchers is that the deep learning approaches can discover the features required to develop models automatically, and likewise, there is no need for a domain expert to define these features manually.

The use of systematic reviews in Software Engineering and the term "Evidence-based Software Engineering" (EBSE) were proposed by Kitchenham et al. [35] in 2004. As part of EBSE, two kinds of systematic reviews, namely Systematic Literature Review (SLR) and Systematic Mapping Studies (SMS), are now used as well-established tools in software engineering [12]. These studies are called secondary studies because they review primary studies that analyze several research questions. Also, they are different than traditional review articles (a.k.a., survey article) because they systematically search electronic databases for relevant articles and follow a well-defined protocol. While SMS studies categorize studies in a particular research area and map them based on several facets [57], SLR studies respond to particular research questions (RQs) by identifying relevant articles, extracting the required data, and synthesizing them [36, 69].

The objective of this article is to better understand the state-of-the-art on the use of deep learning in mobile malware detection by identifying and synthesizing the relevant information systematically and respond to the RQs defined at the beginning of this research. The SLR protocol specified by Kitchenham et al. [37] were followed in this research. Forty articles were selected by applying the study selection criteria. In addition, quality assessment was performed on the selected articles. As such, this paper includes recently published articles in this domain, presents the state-of-the-art, and paves the way for further research on the application of deep learning in mobile malware detection.

To the best of our knowledge, this is the first Systematic Literature Review (SLR) on the review of deep learning-based studies for mobile malware detection. As explained in Sect. 2—Related Work, several survey papers have been published for malware detection so far; however, neither of these focused on deep learning approaches, nor did these analyze mobile malware detection approaches in a systematic way.

A systematic overview of the state-of-the-art on the use of deep learning in mobile malware detection is lacking and this knowledge gap can only be addressed by carrying out a systematic review study (i.e., SLR). Therefore, the objective of this study is to systematically investigate where and how deep learning algorithms have been applied for mobile malware detection by researchers. While there are some review papers published in the literature, there is no systematic review paper that focuses on mobile malware detection. There are some review papers that focused on Android platform, as discussed in Sect. 2.2.; however, there is no such a constraint in our SLR study.

The contributions of this article are threefold:

- High-quality journal articles that focused on the use of deep learning approaches for mobile malware detection were identified, required data were extracted, and synthesized.
- Widely used benchmarking datasets, deep learning algorithms, deep learning development platforms, evaluation parameters, and features were determined and reported.
- Challenges and research gaps in deep learning-based mobile malware detection were briefly presented.

The following research questions have been defined in this SLR study:

- *RQ1* Which machine learning categories (supervised/ unsupervised/semi-supervised/reinforcement learning) have been preferred in deep learning-based mobile malware detection?
- *RQ2* What data sources/features (e.g., API calls and system calls) have been used for the development of the malware detection models?
- *RQ3* What kind of deep learning algorithms (e.g., CNN and LSTM) have been applied?
- *RQ4* What kind of evaluation parameters (e.g., accuracy) and evaluation approaches (e.g., cross-validation) has been used?
- *RQ5* Which deep learning algorithm works best for mobile malware detection?
- RQ6:What kind of feature selection techniques have been used?
- *RQ7* Which public datasets have been analyzed during the development of the models?
- *RQ8* Which deep learning platforms have been used for the implementation of the models?
- *RQ9* What are the challenges and research gaps in mobile malware detection?

The following sections of this paper are organized as follows: Sect. 2 explains Deep Learning algorithms and presents similar review articles. Section 3 describes the adopted research methodology and explains how this SLR study was performed. Section 4 presents the results of the SLR and discusses the responses to research questions. Section 5 presents the general discussion regarding the research questions and also, explains the potential threats to validity. Finally, Sect. 6 discusses the conclusions and future work.

## 2 Background and related work

We explain the deep learning research field and deep learning algorithms in Sect. 2.1. In Sect. 2.2., we present the related review studies and compare our study with these review studies.

### 2.1 Deep learning (DL) and DL algorithms

Deep Learning (DL) is a sub-branch of the machine learning research field that is based on Artificial Neural Networks (ANN) concept. ANN is a biologically inspired computing paradigm that uses artificial neurons in its architecture. Connections (a.k.a., edges) in the architecture transmit a signal from one neuron to another like in the biological brain and edges have weights to adjust the learning process. Neurons are triggered based on the threshold levels. The propagation function is responsible for computing the input to a neuron and the final result is calculated as a weighted sum. A typical ANN model consists of three types of layers: input later, hidden layer, and output layer. Signals reach to the architecture from the input layer, pass through several hidden layers depending on the structure of the network, and finally, reach to the output layer. This typical architecture is also known as Multi-Layer Perceptron (MLP) because it has several layers, and each layer can consist of different number of neurons. The number of neurons in the input layer is determined based on the number of independent variables (i.e., features) and the number of neurons in the output layer is related to the outcome of the prediction task. For instance, for a binary classification task, one neuron is sufficient in the output layer because it can represent two classes when binary representation is used. In DL, the number of hidden layers is much larger than the number of hidden layers in a typical ANN architecture. Due to increasing computing power and different types of computing hardware (e.g., GPU, TPU, etc.), large-scale prediction models can be built easily though it takes a considerable amount of time to find the optimal structure of the network.

Recently, DL algorithms achieved promising results for many problems in different domains such as agriculture [34], medical image analysis [45], mobile & wireless networking [76], and cybersecurity [8, 49].

DL can be considered as a representation learning or feature learning because in contrast to traditional machine learning (a.k.a., shallow learning) in which features are manually extracted, in DL, the features are automatically discovered. Hand engineered features fail to scale in a big data context, which requires the preference for DL algorithms that help to learn the features directly from the data itself. Due to the massive amount of generated data (i.e., big data), fast implementation DL frameworks (e.g., TensorFlow and Keras), parallelizable Graphics Processing Units (GPU), and recent optimized hardware (i.e., Tensor Processing Unit-TPU), DL algorithms are widely used to address challenging problems in different domains.

Compared to the traditional Artificial Neural Network (ANN) algorithms, DL algorithms have many hidden layers (e.g., 1202 layers in ResNet [31]) and different layer types such as convolutional layers and pooling layers. For instance, the Multi-Layer Perceptron (MLP) feedforward ANN algorithm mostly uses fully-connected layers; however, in Convolutional Neural Networks (CNN), not only fully-connected layers but also several convolutional and pooling layers are applied.

In a recent review article, Pouyanfar et al. [59] presented a review of deep learning algorithms and divided the DL algorithms into the following four categories:

1. *Recursive neural network (RvNN)* In RvNN models, the input is processed hierarchically in a tree manner, and unlike recurrent neural networks (RNN), the time dimension is not considered. This type of model provided remarkable results in Natural Language Processing (NLP).
2. *Recurrent neural network (RNN)* RNN algorithms are different than traditional ANN algorithms because they use sequential information in the network [59]. These models are used for sequential inputs, and time is an important element in these models. Long-Short Term Memory (LSTM) models are the most popular RNN algorithms used. There are different variations of LSTM models, namely Vanilla LSTM, Stacked LSTM, CNN-LSTM, Encoder-Decoder LSTM, Bidirectional LSTM, and Generative LSTM [10].
3. *Convolutional neural network (CNN)* Three types of layers are applied in CNN models, namely convolutional layers, pooling layers, and fully-connected layers. Convolutional layers include filters and feature maps. Filters can be considered as the neurons of the layer and create some output value based on the weighted inputs [9]. The feature map is the output of the filter. Pooling layers are used to down-sample the feature map and can reduce the overfitting [11]. In a typical CNN model, convolutional layers are followed by a pooling layer, and this structure is repeated a few times, and finally, a fully-connected layer is applied.
4. *Deep generative networks (DGN)* Generative models are used in many problems such as speech recognition, visual recognition, and robotics [55]. These models aim to capture the probabilistic distribution of the data to generate similar data. Recently, these models were parameterized with the help of deep neural networks,

and stochastic optimization methods and different deep generative networks emerged. These DGN models can be divided into two main categories, namely cost function-based models and energy-based models [55]. In the review article of Pouyanfar et al. [59], the authors divided Deep Generative Networks into the following four algorithm categories. While Deep Belief Networks and Deep Boltzmann Machines are energy-based models, Autoencoder and Generative Adversarial Network are cost-function-based DGN models.

a. *Deep belief network (DBN)* DBNs are probabilistic models and consist of multiple layers of stochastic latent variables. They can be considered as a stack of Restricted Boltzmann Machines (RBM). Each RBM interacts with the previous and next layers. The top two layers create an associative memory and have undirected connections. The lower layers have directed connections.
b. *Deep boltzmann machine (DBM)* DBMs are the stacked layers of Restricted Boltzmann Machines and consist of several hidden layers [61]. However, a Restricted Boltzmann Machine has a single hidden layer. DBM is also a type of Markov random field including several hidden layers of hidden random variables.
c. *Generative adversarial network (GAN)* GAN models use two neural networks (i.e., generator and discriminator) competing one against the other to create new data [27]. While the generator learns to create the data, the discriminator learns to differentiate the fake data from the real one.
d. *Variational autoencoder (VAE)* VAE models have an encoder, decoder, and a loss function. They are used to learn latent representations for unsupervised learning.

During our Systematic Literature Review (SLR) study on mobile malware detection, we identified the following deep learning categories that were not discussed in the review paper of Pouyanfar et al. [59] and applied in malware detection:

- *Deep neural networks (DNN)* These models are similar to the traditional ANN-based models; however, in DNN models, we have many hidden layers instead of a single hidden layer. Also, fully-connected layers are preferred in DNN models, but in other deep learning models such as CNN, different layers such as convolutional layers and pooling layers are used. If there is more than one hidden layer in the topology, we represent the model as a DNN model in this study.
- *Hybrid deep learning model:* Different deep learning algorithms are combined in a way that the combined model is superior to the individual models. For instance, CNN models are used together with autoencoder models for better performance, or LSTM models are integrated with CNN models.
- *Multi-modal deep learning:* Multi-modal learning uses different feature types from multiple sources [6]. For instance, audio, speech, visual, and text information can be used to create a machine learning model. Recently, several deep learning-based approaches have been developed for multi-modal learning [54, 64].
- *Multi-view deep learning* Multi-view learning algorithms were developed with the help of multi-view data [43, 44]. Recently, deep learning-based techniques have been implemented for multi-view learning [22].

In the survey article of Mahdavifar and Ghorbani [49], they classified deep learning architectures into the following three categories and referred to the article of Deng [20] who divided deep learning architectures into the three classes (i.e., generative, discriminative, and hybrid) based on the intention behind using these algorithms.

1. *Discriminative* Discriminative architectures are not related to the data generation and focus on the prediction task. They are less expensive compared to generative models because generative models need additional modeling effort.
    a. Discriminative restricted boltzmann machine (DRBM)
    b. Convolutional neural network (CNN)
2. *Generative* Generative architectures are beneficial for creating new instances that are similar to the existing ones.
    a. Autoencoder
        i. Stacked autoencoder (SAE)
        ii. Denoising autoencoder (DAE)
    b. Recurrent neural network
        i. Long-short term memory (LSTM)
        ii. Echo state network (ESN)
    c. Boltzmann machine
        i. Deep boltzmann machine (DBM)
        ii. Restricted boltzmann machine (RBM)
            1. Deep belief network (DBN)
    d. Generative adversarial network (GAN)
3. *Hybrid* The goal is discrimination but supported with the generative architectures.
    a. Deep neural network

i. Recursive neural network

### 2.2 Related work

Several review articles have been published on the malware detection problem. According to Gibert et al. [26], the other survey papers published so far [[1, 7, 63, 65, 70, 73]] did not review deep learning-based papers. Gibert et al. [26] compared their study with these survey studies based on feature taxonomy, static methods, dynamic methods, hybrid methods, multimodal learning methods, deep learning methods, and issues & challenges dimensions. They reviewed 67 research papers that focused on the application of machine learning for malware detection, and 16 of these papers applied deep learning algorithms. Their main concern was the detection and classification of malware on the Windows platform instead of mobile applications. They specified the following research directions and challenges: availability of the open and public datasets, concept drift concept, incremental learning, adversarial learning, and the problem of class imbalance. Our study is different than the study of Gibert et al. [26] because we mainly focused on deep learning-based techniques (i.e., 40 articles), covered more deep learning papers (i.e., 40 vs. 16 papers) in this systematic literature review, and investigated the approaches developed for mobile applications instead of Windows platform.

Qamar et al. [60] presented a review of mobile malware attacks, malware detection techniques, and solutions using papers published between 2013 and 2019. They discussed five mobile malware detection studies that applied deep learning algorithms. Four of these papers used the Convolutional Neural Network (CNN) algorithm, and only one of them applied the Artificial Neural Networks (ANN) algorithm. They reported that while deep learning-based techniques provide a better performance, they are computationally more expensive compared to the traditional machine learning algorithms. Aslan and Samet [5] reviewed malware detection papers and categorized them into the following classes: signature-based, behavior-based, heuristic-based, model checking-based, deep learning-based, cloud-based, mobile-based, and IoT-based techniques. In their review article, they explained four deep learning-based papers that used Deep Belief Network (DBN), ANN, and Convolutional layers and stated that API calls, systems calls, and hybrid features were utilized. Their primary concern was not the analysis of deep learning-based approaches for malware detection.

Pan et al. [56] performed an SLR in Android malware detection and reviewed papers that apply static analysis. They divided the static analysis into the following four categories: Android characteristic-based, opcode-based, program graph-based, and symbolic execution-based method. They represented deep learning-based models under the Neural Network (NN) category while classifying the machine learning models in the literature. They showed the following deep learning models in the classification taxonomy: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long-Short Term Memory (LSTM), Deep Belief Network (DBN), and Artificial Neural Network (ANN). They concluded that static analysis is effective in detecting Android malware based on several experiments and empirical results. Their primary focus was the use of static analysis in the Android platform. However, in this article, we do not have such a restriction, and we review deep learning-based approaches for mobile malware detection.

Feizollah et al. [24] reviewed 100 papers based on their features and feature selection techniques and reported that only 8 out of 100 papers applied feature selection algorithms such as information gain. They divided features into the following categories: static features (permissions, Java code, intent filters, network address, strings, and hardware components), dynamic (system calls, network traffic, system components, user interactions), hybrid (the combination of static and dynamic), and applications metadata (application description, creator ID, and application category). They reported that static features were used 45%, dynamic features 42%, the hybrid features 10% and applications metadata 3% in the identified papers. Among static features, Android permission was the most popular one (36%) because permissions are an important barrier to attackers. Among dynamic features, the system calls were the most dominant category, and the number of papers that used the network traffic data was nearly half of the papers that used the system calls. Liu et al. [46] performed a survey on Android malware detection approaches using machine learning. They divided papers based on the following machine learning categories: Decision Trees, Naive Bayes, Linear Model, Support Vector Machines, K-Nearest Neighbors, K-means, Neural Network & Deep Learning (NN & DL), Ensemble Learning, and Online Learning. They showed 21 papers under the NN & DL category and reported that while algorithms in this category provide high performance, they require a lot of data for training. They provided several future research directions from the machine learning perspective. For instance, they referred to the problem of concept drift, which means the predictive performance of classifiers can decline over time. They state that incremental learning can be used to add sample data dynamically, active learning can be utilized for the data scarcity problem, and transfer learning can be applied to transfer the knowledge from one domain to another. They also explain that the reliability estimation based on the evaluation results has not been studied in detail yet for Android malware detection.

According to our literature search and the summary presented in Table 1, no paper has systematically analyzed and synthesized the articles that used deep learning algorithms for mobile malware detection problem yet. Besides only one paper [5] focused on either the Android or Windows platforms. We did not restrict our search only to the Android platform and accessed 40 high-quality articles from electronic databases using a systematic literature review. We focus explicitly on deep learning for malware detection. Hence, our research is different than the other review studies and provides complementary and significant insight into this important problem.

## 3 Research methodology

Figure 1 shows our SLR process, which has been formed based on well-known review protocol and guidelines [37] and our experience in SMS and SLR studies [14, 15, 67, 69]. Our SLR process included three main phases, i.e., primary study selection, data extraction, and data synthesis and reporting. The subsections 3.1, 3.2 and 3.3 present each of these phases.

### 3.1 Primary study selection

We used five widely used online databases, i.e., ACM, IEEE Xplore, ScienceDirect, Springer, and Wiley. We used the following search string to query the five online databases: *“deep learning” AND (“malware detection” OR “malware classification” OR “malware analysis”).*

To identify the relevant set of papers to answers our research questions, we specified our exclusion criteria, as listed in Table 2.

Authors held a meeting and explained how they applied the exclusion criteria. This think-aloud application of selection criteria [2] helped to clarify ambiguities and unintended interpretations. After applying the exclusion criteria, we obtained a set of papers consisting of 29 papers.

Before extracting data from the primary studies, we conducted a quality assessment, as proposed in the literature [29]

Table 3 lists the criteria we used for quality assessment. These criteria have been derived from [37] and used in earlier SLRs, such as [69]. For each criterion, we scored the papers using a 3-point Likert scale (yes = 1, somewhat = 0.5, no = 0). For instance, we scored for Q1 as 1 if the aim of the study was stated clearly in the introduction (expected place); as 0.5 if the aim was vaguely stated, or not at the expected place, and as 0 if the aim was not stated in the paper.

**Table 1** Summary of related work

| Year | Source | Review type | Mobile platforms | Focus | Description |
|---|---|---|---|---|---|
| 2015 | Feizollah et al. [24] | Non-systematic | Android | Only traditional ML algorithms | A non-systematic literature review of feature selection for Android malware detection |
| 2019 | Qamar et al. [60] | Non-systematic | Android | Five DL-based primary studies for mobile malware detection | A non-systematic literature review of Android mobile malware attacks and detection methods |
| 2020 | Aslan and Samet [5] | Non-systematic | Android and windows | Four DL-based primary studies for mobile malware detection | A non-systematic literature review of malware detection methods on Android and Windows platforms |
| 2020 | Gibert et al. [26] | Systematic | Windows | 16 DL-based primary studies out of 67 DL machine learning papers | A systematic literature review of malware detection and classification approaches using machine learning on Windows platform |
| 2020 | Liu et al. [46] | Systematic | Android | 21 DL-based primary studies | A systematic literature review of Android malware detection methods using ML |
| 2020 | Pan et al. [56] | Systematic | Android | No special focus on traditional ML or DL; Only detection methods using static analysis included (dynamic and hybrid analysis excluded) | A systematic literature review of Android malware detection methods using static analysis |

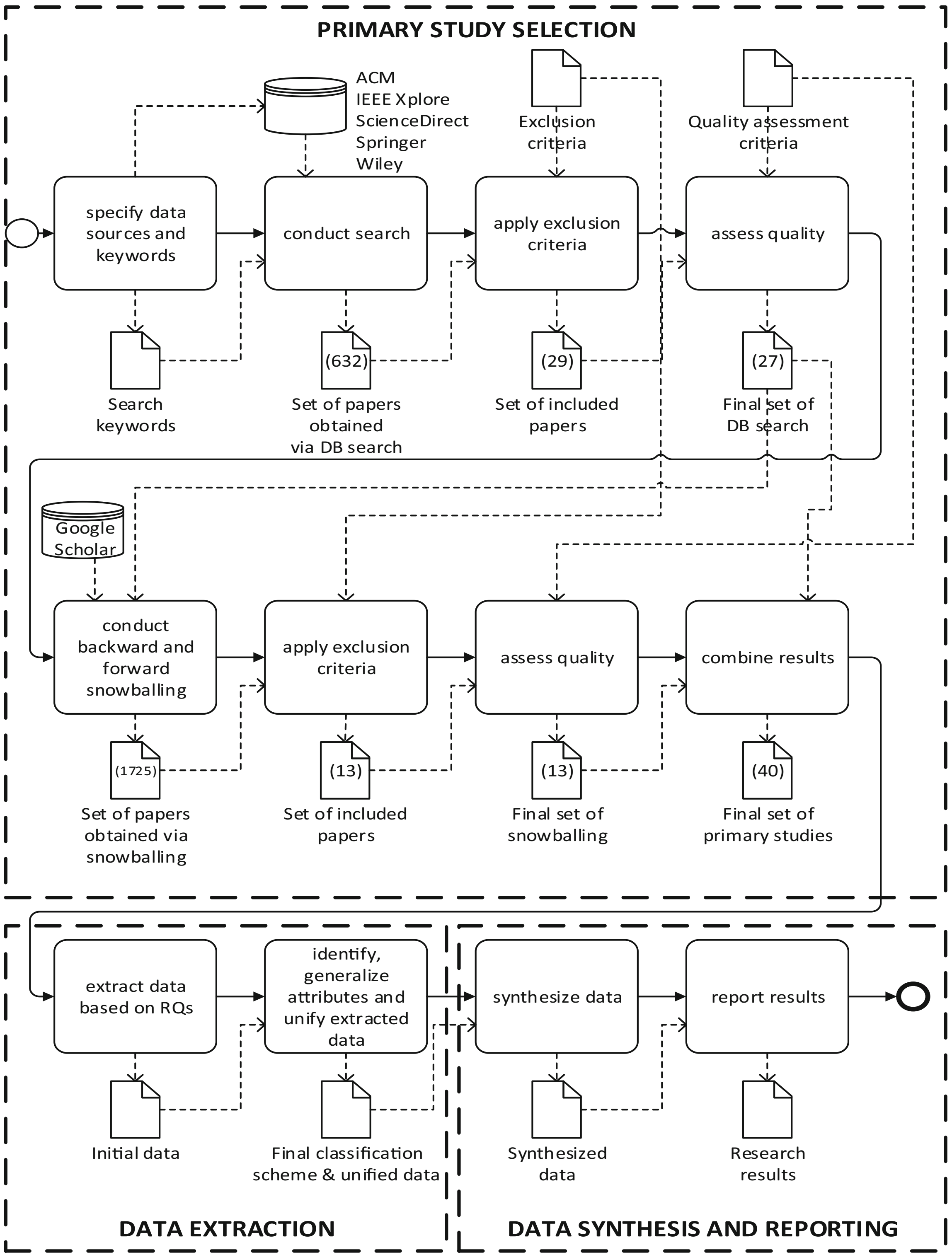

**Fig. 1** SLR process used in this study

To maintain a high-quality input of primary studies for this SLR, we decided to exclude the papers with a score lower than four points out of eight. We excluded two studies [32, 66] with a score under our threshold. As a result of the database search, we had a total number of 27 papers.

To identify the primary studies that we could have overlooked with our automatic queries, we further employed the snowballing strategy [71]. We used Google Scholar to check the papers that cited the 27 primary studies we identified via database search. After applying

**Table 2** Exclusion criteria

| # | Criterion |
|---|---|
| EC1 | Duplicate papers from multiple sources |
| EC2 | Papers without full text available |
| EC3 | Papers not written in English |
| EC4 | Papers not published in a journal |
| EC5 | Short papers, editorials, issue introductions |
| EC6 | Secondary studies, such as literature review, SMS, SLR |
| EC7 | Papers which do not use deep learning to detect malware in mobile applications |
| EC8 | Papers which only use traditional ML algorithms |
| EC9 | Papers which do not include empirical results |

**Table 3** Quality assessment criteria

| # | Question |
|---|---|
| *Q*1 | Are the aims of the study clearly stated? |
| *Q*2 | Are the scope and context and experimental design of the study clearly defined? |
| *Q*3 | Are the variables in the study likely to be valid and reliable? |
| *Q*4 | Is the research process documented adequately? |
| *Q*5 | Are all the study questions answered? |
| *Q*6 | Are the negative findings presented? |
| *Q*7 | Are the main findings stated clearly (regarding creditability, validity, and reliability)? |
| *Q*8 | Do the conclusions relate to the aim of the purpose of the study, and are they reliable? |

the snowballing strategy and our exclusion criteria listed in Table 2, we obtained an additional set of 13 primary studies.

Furthermore, we assessed the quality of these papers using the criteria listed in Table *3*. None of the papers obtained via snowballing was disqualified after quality assessment. Figure 2 shows the quality scores of the selected primary studies. While x-axis shows the quality score of papers, the y-axis represents how many papers had this score. For instance, according to Fig. 2, it is shown that 8 papers had 7.5 scores and only 2 papers had 3.5 scores. Since papers that had less than 4 points were removed, these two papers were not used in this SLR study. As shown in Fig. 2, most of the papers (i.e., 26 papers) had more than 6 points and the rest had a score between 4 and 6.

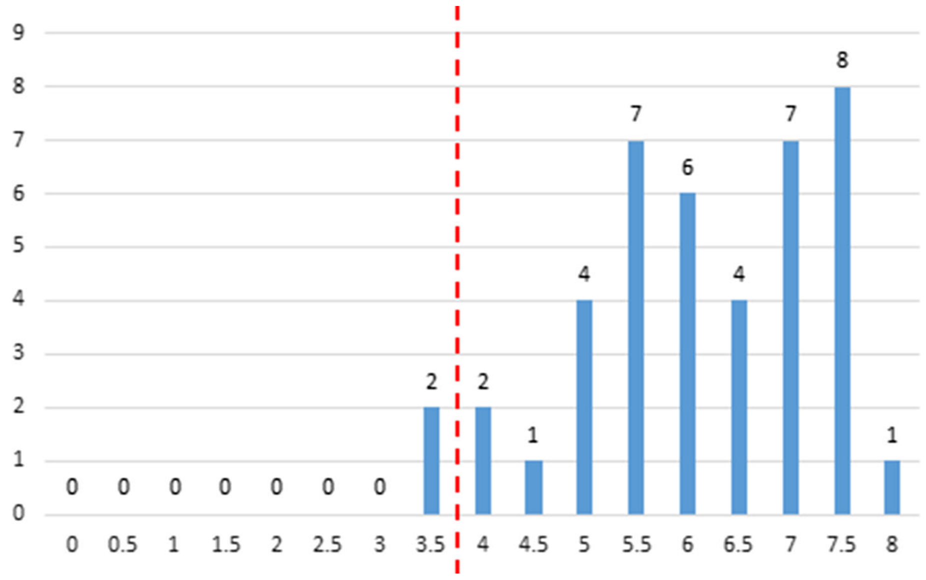

**Fig. 2** Quality score distribution of the selected papers

Finally, we combined the data about 40 primary studies (listed in Sect. 0) in a Google sheet.

## 3.2 Data extraction

After selecting the primary studies, we started with the data extraction phase. Data extraction steps were highly iterative and required close collaboration among the authors.

As a result, the authors formed the final classification scheme, as listed in the second column of Table 4.

## 3.3 Data synthesis and reporting

Since we managed to categorize the extracted data for most of the RQs, the data extraction phase yielded a set of quantitative data to be synthesized. We reported the frequencies and percentages of each identified category to answer the RQs.

The only RQ that required qualitative analysis is RQ9, that is, the challenges and proposed solutions. We conducted open coding [51] in cycles to analyze the challenges. A code symbolically assigns a summative or evocative attribute for a portion of qualitative data [51].

## 4 Results

In this section, we present the responses to research questions defined at the beginning of this research. Before presenting the responses, we provide additional information about the identified articles, such as the yearly distribution and distribution of articles per journal. As shown in Fig. 3, the number of articles is increasing each year. Although we completed our search process in the middle of 2020, the number of papers is more than the number of papers in 2019. This indicates that the application of deep learning algorithms for mobile malware detection is the new trend among cybersecurity researchers. We expect more research in this direction in the upcoming years. In this SLR study, we focused on high-quality papers and, therefore, selected articles published in journals.

In Table 5, we show the distribution of articles per journal. According to this table, the most preferred journal is Computers & Security, and the second one is IET Information Security. In addition to these journals, we identified two papers from the following journals: Cluster Computing, Journal of Ambient Intelligence and Humanized Computing, Journal of Intelligent & Fuzzy Systems, KSII Transactions on Internet and Information Systems (TIIS), Multimedia Tools and Applications, and Soft Computing. The other journals included only one article on

**Table 4** Data extraction form

| Field | Categories |
|---|---|
| Journal title | Free text |
| Publication year | Number |
| Paper title | Free text |
| Abstract | Free text |
| ML category | Supervised, unsupervised, semi-supervised |
| Type of analysis | Static, dynamic, hybrid |
| Data sources | API call, application component, binary info, bytecode, certificate info, environmental feature, hardware component, intent, network address, network traffic, opcode, payload info, permissions, system call, source code, URL |
| DL approaches | Autoencoder, CNN, DBN, DNN, GAN, Hybrid DL model, multi-modal DL, multi-View DL, RNN/LSTM |
| Evaluation parameters | Accuracy, AUC, *F*-measure, FNR, FPR, PPV, precision, recall, TPR |
| Validation method | Cross-validation, Hold-out, Not mentioned |
| Best algorithm | CNN, DBN, DNN, GAN, Hybrid DL model, LSTM, Multimodal DL, Multiview DL, Traditional ML |
| Feature selection method | Autoencoder-based, boruta, infogain, random forest-based, relief, SAILS, Not mentioned |
| Evaluation dataset | AMD dataset, android adware and general malware dataset (AAGM), android malware genome project, android PRAGuard dataset, AndroZoo, Contagio mobile website (iOS), drebin, ISCX android botnet dataset, maldozer, virusshare, virustotal, custom dataset, other, not mentioned |
| The implementation platform | Caffe, H2O, keras, matlab, tensorflow, torch, not mentioned |
| Challenges and proposed solutions | Free text |

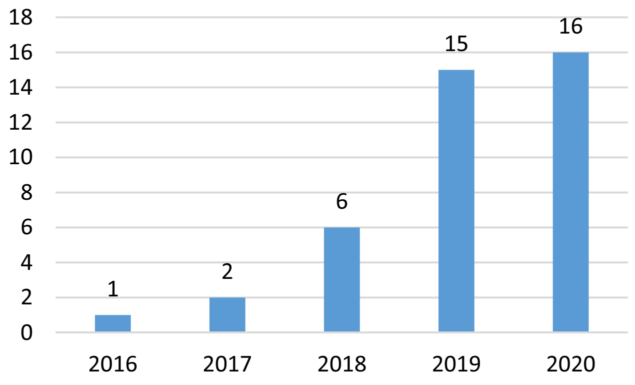

**Fig. 3** Number of papers until March 2020

the use of deep learning for mobile malware detection. Researchers who would like to publish their recent research results can consider these journals because of the relevant scope of these journals.

### 4.1 RQ-1: Machine learning categories

The first research question is on the machine learning categories. The labeling process of the data points for malware detection is time-consuming, labor-intensive, and expensive. For supervised learning approaches, labeled data points are needed because unlabeled data points cannot be used in that case; however, for unsupervised learning, these labels are not needed; instead, unlabeled data points can be used directly. In semi-supervised learning, unlabeled data points are used together with the labeled data points to improve the overall performance, and first, pre-labels are assigned to the unlabeled data points, and then, the labels are adjusted depending on the error parameter iteratively. In reinforcement learning, an agent observes the environment, performs some actions based on the available data and rules, and gets rewards (positive or negative) depending on the preferred action. The aim of reinforcement learning is to maximize rewards while

**Table 5** Distribution of papers per journal

| Journal | # of Papers | Reference(s) |
|---|---|---|
| Computers & security | 4 | [P2,P20, P21, P36] |
| Iet information security | 3 | [P6, P29, P33] |
| Cluster computing | 2 | [P5, P19] |
| Journal of ambient intelligence and humanized computing | 2 | [P10, P32] |
| Journal of intelligent & fuzzy systems | 2 | [P22, P30] |
| KSII transactions on internet and information systems (TIIS) | 2 | [P7, P27] |
| Multimedia tools and applications | 2 | [P34, P40] |
| Soft computing | 2 | [P24, P28] |
| Alexandria engineering journal | 1 | [P25] |
| Applied soft computing | 1 | [P26] |
| Concurrency and computation: practice and experience | 1 | [P14] |
| Digital investigation | 1 | [P12] |
| Engineering applications of artificial intelligence | 1 | [P16] |
| Expert systems with applications | 1 | [P39] |
| Future generation computer systems | 1 | [P4] |
| IEEE access | 1 | [P1] |
| IEEE transactions on information forensics and security | 1 | [P13] |
| IEEE transactions on systems, man, and cybernetics: systems | 1 | [P37] |
| Information Sciences | 1 | [P31] |
| International journal of engineering & technology | 1 | [P11] |
| Journal of computer virology and hacking techniques | 1 | [P18] |
| Journal of information security and applications | 1 | [P9] |
| Journal of parallel and distributed computing | 1 | [P8] |
| Microelectronics reliability | 1 | [P35] |
| Neural computing and applications | 1 | [P15] |
| Neurocomputing | 1 | [P23] |
| Procedia computer science | 1 | [P17] |
| Transactions on emerging telecommunications technologies | 1 | [P3] |
| Tsinghua science and technology | 1 | [P38] |

continuously interacting with the environment. According to our analysis shown in Table 6, all the articles applied supervised learning approaches; however, one article also included a semi-supervised learning algorithm, and one article applied unsupervised learning algorithm as well. Since we explain deep learning approaches in Sect. 4.3 in detail, in this subsection, we do not explain these supervised learning studies.

There is only one article that applied semi-supervised learning [P26] in addition to the deep learning-based supervised learning model. Sharmeen et al. [P26] stated that their semi-supervised approach extracts the hidden patterns with the help of a clustering algorithm and combines these patterns to the supervised classifier with the Euclidean metric. They concluded that the deep learning-based supervised learning approach provides better performance compared to the clustering-based semi-supervised learning approach. However, their semi-supervised learning approach did not apply deep learning-based semi-supervised learning approaches, and therefore, we consider that deep learning-based semi-supervised learning approaches can be a future research direction for mobile malware detection.

Only one study focused on the use of unsupervised learning. Kim et al. [P13] applied Multi-Modal Neural Network (MNN) shaped autoencoder and Deep Neural Network (DNN) shaped autoencoder for malware detection and reported that MNN shaped one provides 6% better performance than the DNN shaped autoencoder approach. We did not encounter any article that applied deep learning-based reinforcement learning.

This is an important observation because it shows that most of the DL-based mobile malware prediction efforts go into the supervised learning strategies and models. However, more impact and benefits will come from unsupervised and semi-supervised learning sides. While deep learning algorithms provide great performance for supervised learning, there is a need to develop new algorithms for other learning categories as well. This will also trigger developing new malware prediction models based on these new learning algorithms.

**Table 6** Distribution of articles per machine learning category

| Machine learning category | Number of papers |
|---|---|
| Supervised learning | 40 |
| Unsupervised learning | 1 |
| Semi-supervised learning | 1 |
| Reinforcement learning | 0 |

## 4.2 RQ-2: Data sources/features

Features of the models are one of the most critical elements while designing machine learning-based malware detection models. As explained in the Introduction section, three kinds of analysis, namely static, dynamic, and hybrid, can be performed for the detection of malware. While static features can be collected without the execution of the application, dynamic features require the execution of the application. Hybrid features can be considered as a combination of static and dynamic features. Most of the time, binary files are used to collect static features. For instance, opcode, which is a single instruction executed by the CPU, is an example of static features. Network traffic-related features are examples of dynamic features because the application must be installed in a smartphone, and the network traffic must be observed and collected. As shown in Fig. 4, most of the articles (i.e., 55%) applied static features, and only 12,5% of the articles used hybrid features. The collection of dynamic data requires more effort because the application must be run, and the required data must be collected. As such, most of the researchers might have preferred to develop models based on only static features. However, the addition of dynamic features to the models also provides extra benefits, and we expect to see more research on the use of hybrid features in the future.

In Fig. 5, we show the distribution of data sources that were used for the implementation of the prediction models. API calls and permissions are the most used features for malware detection. Since most of the articles focused on the Android platform and Android apps request permission for sensitive data, researchers used these *permissions* more often than the other features. The API calls are also used as much as permissions because, in this context, API calls provide valuable information to infer certain behavior. The other data sources are presented in Fig. 5. The other data sources are system calls, intent, application component, hardware component, opcode, binary information, bytecode, network traffic, URL, certificate info, environmental features, network address, and payload info.

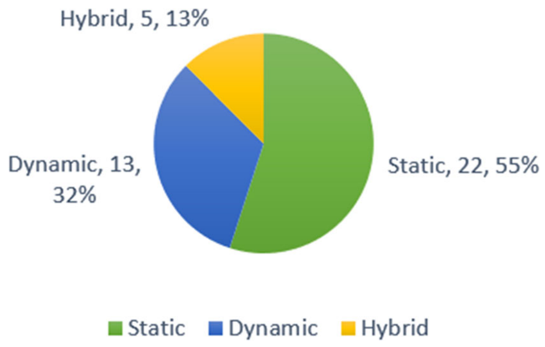

**Fig. 4** Distribution of feature types

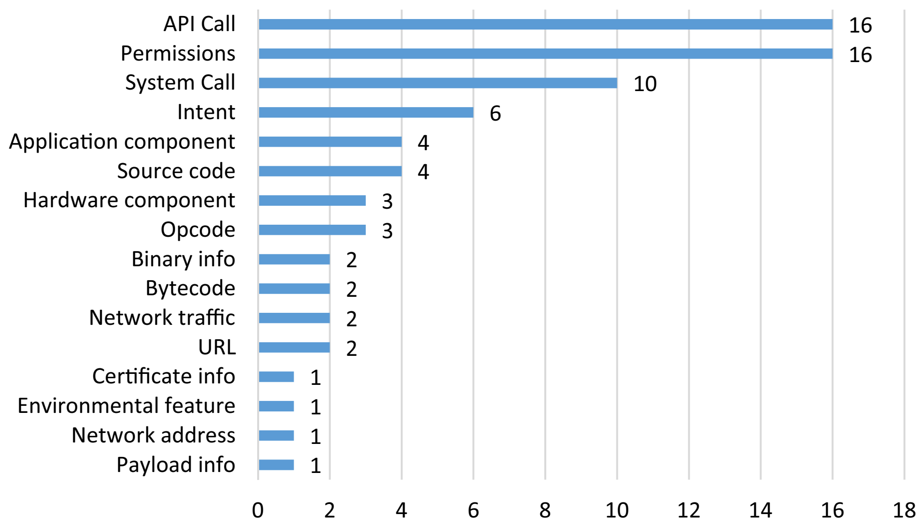

**Fig. 5** Distribution of data sources

Figures 4 and 5 provide important observations. The first one indicates that researchers did not focus on combining the static and dynamic features much, however, the power of this combination (i.e., hybrid features) will result in improved performance in malware prediction models because the models will utilize from different kinds of features instead of depending on one type of feature set. The second one shows that while there are different data sources types, most of the researchers focused on one type of data source (e.g., API calls and permissions), however, the integration of different kinds of data sources can dramatically improve the performance of the prediction models. This big picture shows the possibilities for researchers and practitioners that have not been evaluated in detail before.

## 4.3 RQ-3: deep learning approaches

In Sect. 2.1, we explained the deep learning and related algorithms applied for mobile malware detection. In Fig. 6, we show the distribution of deep learning algorithms. The most used deep learning algorithms are Convolutional Neural Networks (CNN) and Deep Neural Network (DNN) algorithms. The other widely used algorithms are Recurrent Neural Networks (RNN)/Long-Short Term Memory (LSTM) networks and Deep Belief Networks (DBN). Since DNN algorithms resemble the traditional ANN algorithms, researchers easily apply these algorithms to this problem. CNN and LSTM are also widely known and used algorithms in many different problems, and therefore, they might be selected for the application. However, some of the algorithms are not used much, such as Autoencoder, Hybrid deep learning algorithms, Generative Adversarial Networks (GAN), Multi-Modal Deep Learning, and Multi-View Deep Learning.

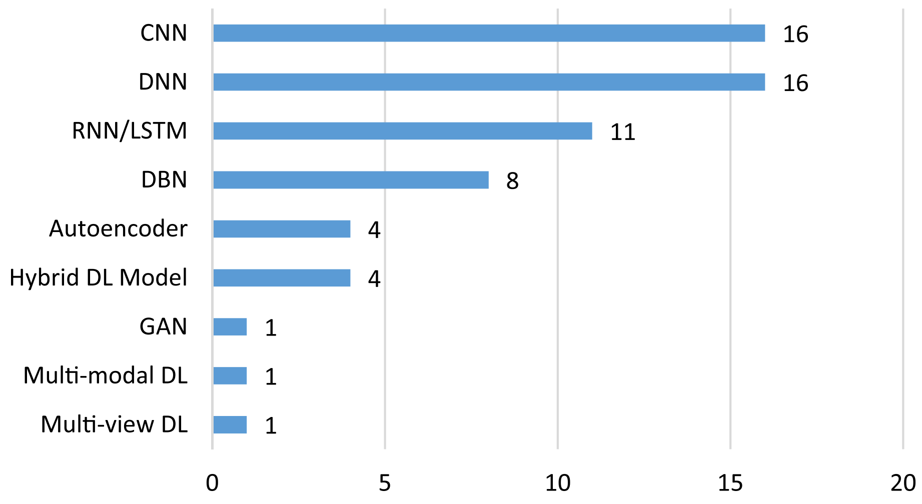

**Fig. 6** Distribution of DL approaches

Advantages and disadvantages of these algorithms are listed as follows:

- CNN
  - o *Advantage* Neurons are not necessarily fully connected to each other. More complex problems can be handled with these algorithms compared to the shallow learning algorithms.
  - o *Disadvantage* A large dataset is required to build the model and finding the optimal architecture takes time.
- DNN
  - o *Advantage* It is effectively used for classification and regression problems. Building the model is relatively easier because there are not many layer types to select.
  - o *Disadvantage* The number of hidden layers affect the network complexity and therefore, complex network requires more computational time and power compared to the shallow learning algorithms.
- RNN/LSTM
  - o *Advantage* It can model sequence data and handle varying length inputs.
  - o *Disadvantage* While RNNs cannot store past information for a long time, LSTMs are vulnerable to overfitting.
- DBN
  - o *Advantage* Training time is shorter and provides fast inference.
  - o *Disadvantage* Increasing run-time complexity.
- Autoencoder
  - o *Advantage* There is no need for labeled data.
  - o *Disadvantage* There is a need for a pre-training stage.
- Hybrid Model
  - o *Advantage* It combines the powerful features of different deep learning algorithms.
  - o *Disadvantage* More computational time and power are required.
- GAN
  - o *Advantage* It does not introduce any deterministic bias like autoencoders. It does not require any Monte Carlo approximations for training.
  - o *Disadvantage* Different types of data must be provided.
- Multi-modal DL
  - o *Advantage* It combines different types of information to improve performance.
  - o *Disadvantage* Combining different levels of noise is challenging and difficult to manage.
- Multi-view DL
  - o *Advantage* Heterogeneous sources (i.e., views) are used to address the given problem.
  - o *Disadvantage* It requires more training data.

We did not encounter any article that used multi-task learning for mobile malware detection. Also, the number of papers that used multi-modal and multi-view learning is very limited. We expect to see more research on the use of multi-modal deep learning and multi-view deep learning in the future because these algorithms combine several types of features and provide better performance in some cases. Also, multi-task learning can be applied for mobile malware detection in the future.

This is an important observation because most of the researchers and practitioners develop malware prediction models using well-known DL algorithms such as CNN and LSTM, however, they do not consider the other alternatives shown in Fig. 6. This figure also shows that there are still future research opportunities because some algorithms such as multi-modal DL and multi-view DL have not been investigated in detail yet in the literature.

## 4.4 RQ-4: evaluation parameters and validation approaches

For the evaluation of the prediction models, researchers used different evaluation parameters and different validation approaches. In Fig. 7, we depict the frequency of each evaluation parameter that was used in the identified articles. Most of the papers used the accuracy evaluation parameter, and the second most used parameter is the F-measure. Precision and recall parameters are the other widely used parameters. The other evaluation parameters are False Positive Rate (FPR), Area under ROC Curve (AUC), True Positive Rate (TPR), False Negative Rate (FNR), and Positive Predictive Value (PPV).

In Fig. 8, we represent the distribution of validation approaches. According to this figure, most of the papers preferred the use of a cross-validation approach instead of a hold-out strategy. There were also a few papers that did not mention the validation approach. We suggest researchers explain this kind of information because the repeatability in experimental studies is crucial.

Figures 7 and 8 provide important observations. The first one indicates that some evaluation metrics (i.e., accuracy and F-measure) are widely preferred by researchers and the second one shows that cross-validation

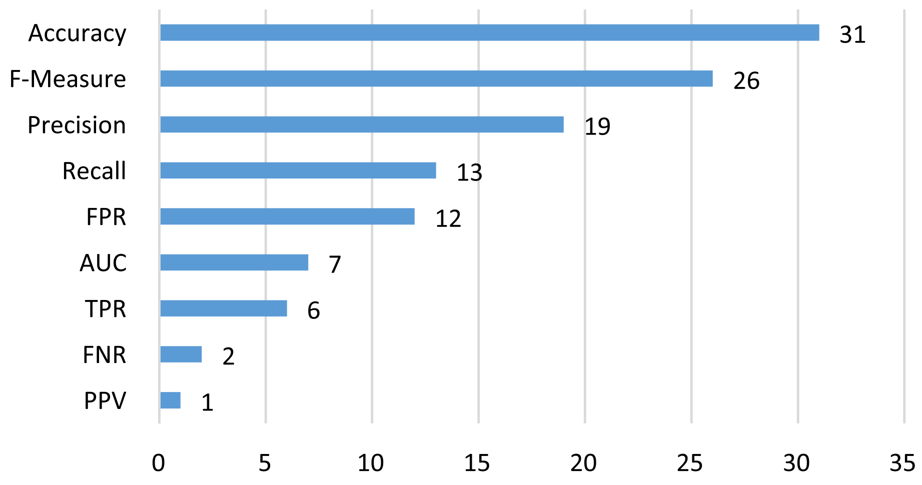

**Fig. 7** Distribution of evaluation parameters

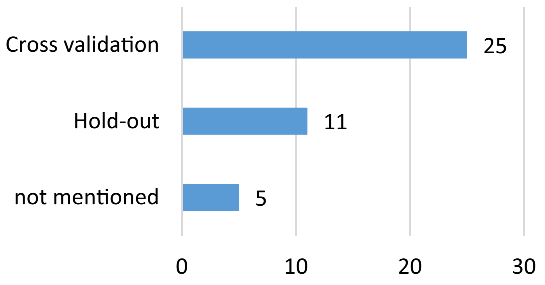

**Fig. 8** Distribution of validation approaches

is mostly used in malware prediction studies. For new researchers in this field, these observations are very valuable because it's hard to select the right metrics and validation approaches for research.

### 4.5 RQ-5: best performing algorithms

In Sect. 4.3, we showed how many times each deep learning algorithm has been used in articles. We observed that CNN and DNN were applied 16 times out of 40 articles. However, the best performance might have been reached with another deep learning algorithm in these studies. To this end, we investigated articles based on the best performing algorithm.

According to Fig. 9, Deep Neural Networks (DNN) was reported the best algorithm in 11 articles, and Convolutional Neural Network (CNN) algorithm was reported as the best algorithm in 8 articles. The other best performing algorithms are Long-Short Term Memory (LSTM) (i.e., 5) and Deep Belief Network (DBN) (i.e., 5) algorithms. Traditional machine learning algorithms were reported as the best performing algorithm in 4 articles. Three articles reported that the best model is achieved with the hybrid deep learning models, and two articles reported the best model based on the multi-modal deep learning approach. Generative Adversarial Networks (GAN) and Multi-View Deep Learning were reported as the best performing algorithm only in one article.

This is an important observation because most of the researchers and practitioners develop malware prediction models using well-known DL algorithms such as CNN and LSTM. However, as shown here, DNN algorithm also provides good performance for this problem. It is also shown that some algorithms such as hybrid DL, multi-modal DL, multi-view DL, and GAN have not been investigated in detail yet in literature. These algorithms can improve the performance of prediction models and therefore, they provide insights for researchers.

### 4.6 RQ-6: feature selection techniques

Since deep learning algorithms can learn using many features, the selection of features is not mostly preferred. Also, this type of algorithms can discover the features from the data itself instead of hand-engineered features. To this end, we observed that many of the articles (i.e., 32) did not specify any feature selection technique in the article. This means that most of the papers did not need any feature selection technique in the study. However, eight articles stated that researchers used some feature selection techniques, namely Random Forest-based feature selection, InfoGain, SAILS, Relief, Boruta, and Autoencoder-based feature selection. In Fig. 10, the distribution of these techniques is presented. If we investigated the traditional machine learning-based mobile malware detection articles, we would probably encounter more feature selection techniques. This observation indicates that the feature selection technique is not required while developing deep learning-based models. This is an important observation because new researchers in DL might consider what type of feature selection techniques they must prefer while developing their models. However, as shown in this figure, a

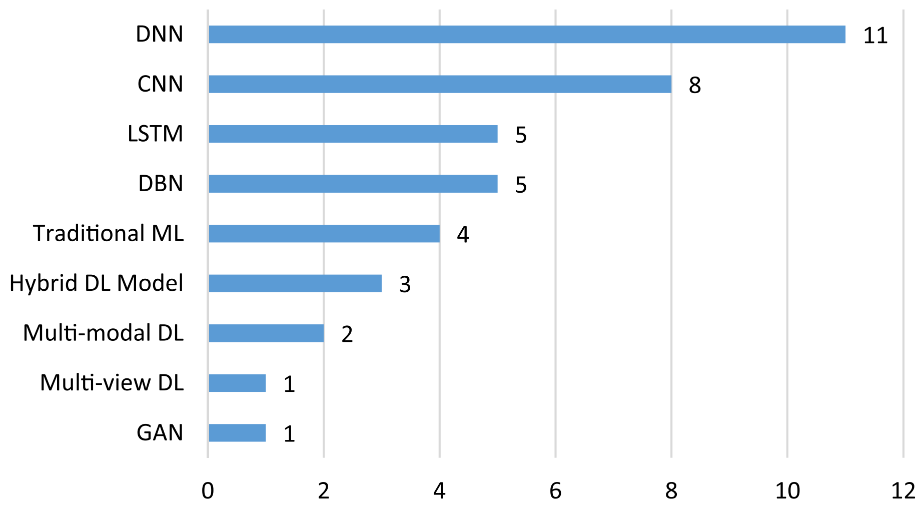

**Fig. 9** Distribution of best performing algorithms

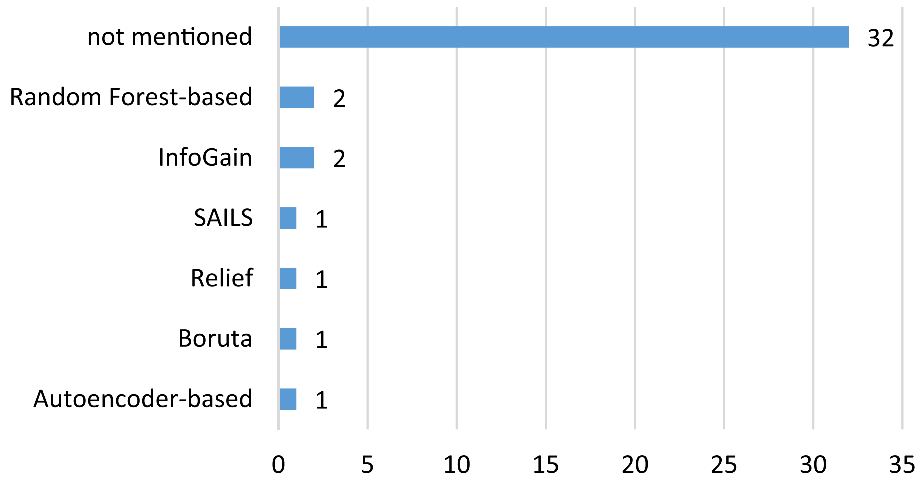

**Fig. 10** Feature selection

feature selection component is not required for DL-based malware prediction models.

### 4.7 RQ-7: datasets

We investigated the datasets that were used by the researchers in the articles. The following datasets were used in these articles: Drebin, VirusShare, Android Malware Genome Project, AMD dataset, Contagio Mobile website, AndroZoo, VirusTotal, Android PRAguard dataset, MalDozer, ISCX Android botnet dataset, and Android Adware and General Malware (AAGM) dataset.

As shown in Fig. 11, the most used dataset is the Drebin dataset, and the second most used one is the VirusShare dataset. Some articles combined existing datasets and created their own datasets (i.e., custom dataset category) or did not specify the applied dataset (i.e., not mentioned category). The webpages of these datasets are shown in Table 7. There are also other datasets that are not used in these articles. Canadian Institute for Cybersecurity also shares several datasets in the following link: https://www.unb.ca/cic/datasets/index.html. Also, the Kharton dataset can be accessed from the following link: http://kharon.gforge.inria.fr/dataset/

Figures 10 and 11 provide important information for researchers because studies in this field either use a few datasets for research or do not discuss the possible other datasets. Here, we presented these datasets with their webpages.

### 4.8 RQ-8: implementation platforms

We investigated the deep learning implementation platforms used during the experiments. The following deep learning development platforms were identified during this SLR study: Keras (https://keras.io/), TensorFlow (https://www.tensorflow.org/), Torch (https://pytorch.org/), H2O (https://www.h2o.ai/), Caffe (https://caffe.berkeleyvision.org/), and MATLAB (https://www.mathworks.com/products/matlab.html).

**Fig. 11** Distribution of datasets

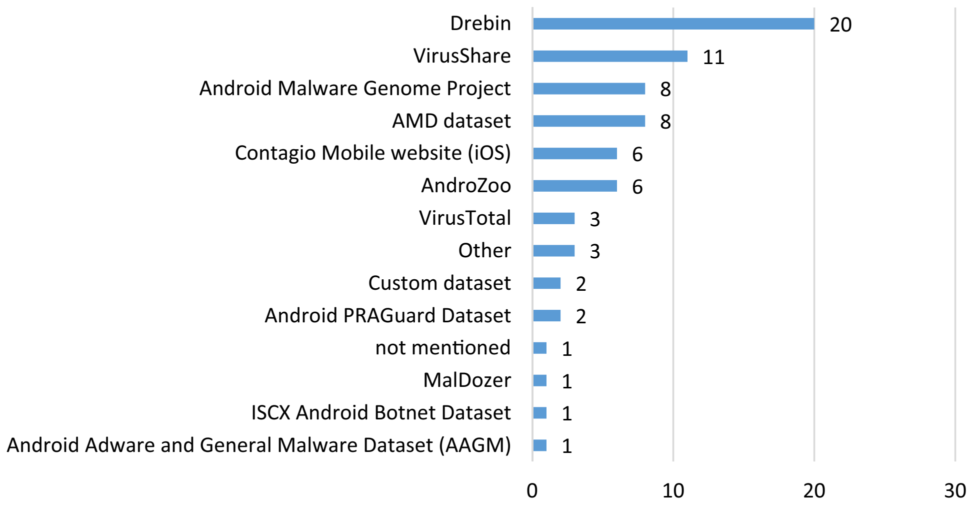

**Table 7** Malware datasets and their webpages

| Dataset | Webpage |
|---|---|
| Drebin | https://www.sec.cs.tu-bs.de/~danarp/drebin/ |
| VirusShare | https://virusshare.com/ |
| AMD dataset | http://amd.arguslab.org/ |
| Android malware genome project | http://www.malgenomeproject.org/ |
| Contagio | http://contagiomobile.deependresearch.org/index.html |
| AndroZoo | https://androzoo.uni.lu/ |
| VirusTotal | https://www.virustotal.com/intelligence/help/ |
| AndroidPRAGuard dataset | http://pralab.diee.unica.it/en/AndroidPRAGuardDataset |
| AAGM | https://www.unb.ca/cic/datasets/android-adware.html |
| ISCX android botnet | https://www.unb.ca/cic/datasets/android-botnet.html) |
| MalDozer [P12] | Available upon contacting authors |

Keras is a high-level Application Programming Interface (API) that runs on top of several frameworks such as TensorFlow and Theano. TensorFlow is an open-source deep learning software library and was developed by the Google Brain team. For quick implementations, Keras is preferred instead of TensorFlow because of its easy to use functions. Keras was developed by a Google developer as part of the ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) project. The Torch platform has been developed by the Facebook AI group, and it is an open-source deep learning library. H2O is an open-source deep learning library and was developed by a company now called H20.ai. Caffee is a deep learning framework and was developed by Berkeley AI Research. MATLAB platform is a computing environment developed by MathWorks. Although it is not specifically designed for deep learning research, it can be used to develop deep learning-based models.

In Fig. 12, we show the distribution of DL implementation platforms. According to this figure, the most used implementation platform is Keras. On the other hand, 14 studies did not specifically mention the platform. In addition to Keras, TensorFlow is also widely preferred (i.e., ten studies) by researchers. However, the use of Torch, H2O, Caffe, and MATLAB is very limited. Except for the MATLAB platform, these deep learning development platforms were used only in 1 article.

Figure 12 presents important information for practitioners and researchers because it shows that Keras platform and Tensorflow are widely used by researchers. In other articles, researchers can see other types of platforms, however, in this paper we evaluated 40 studies and presented the preferred implementation platforms. It is also interesting to note that there are many papers (i.e., 14 papers) that did not mention about the platform, however, we believe that the specification of the implementation platform is also important for repeatable experiments.

## 4.9 RQ-9: challenges

Challenges and possible solutions were extracted from the identified articles; however, not all of these primary studies

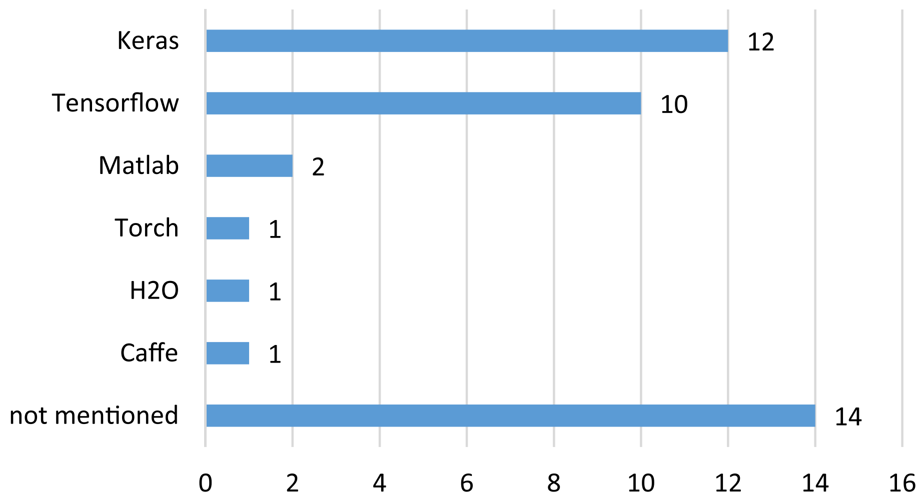

**Fig. 12** Distribution of DL implementation platforms

**Table 8** The challenges and proposed solutions reported in 19 primary studies

| Category | Challenges ($C1$–$C12$) | Proposed solutions ($S1$–$S19$) | Reference |
|---|---|---|---|
| Dataset | $C1$. Obtaining training dataset | $S1$. Researchers' contribution to public datasets | [P38] |
| | $C2$. Imbalanced training dataset | $S2$. Using imbalanced learning algorithms | [P16] |
| | $C3$. Including only permission sets in the dataset | $S3$. Adding extra inputs such as system calls and network traffic to the training dataset | [P10] |
| | $C4$. Wrong labels in the training dataset | $S4$. Checking samples against well-known data sources, such as VirusTotal service | [P18] |
| Model building | $C5$. Long training time | $S5$. Using parallel strategies and algorithms | [P39] |
| | | $S6$. Using CNN algorithm instead of LSTM algorithm | [P4] |
| | | $S7$. Using deep autoencoder algorithm as a pre-training step of the CNN algorithm | [P32] |
| | | *No specific solution proposed* | [P8] |
| | | $S8$. Using broad learning algorithm instead of the backpropagation algorithm conducted over multiple hidden layers on CNN and MLP | [P37] |
| | | $S9$. Using early stopping, i.e., monitoring the performance of a model and terminating optimization when there is no further improvement in an epoch | [P24] |
| | $C6$. Hyper-parameter optimization | $S10$. Using Grid Search algorithm | [P23] |
| | | $S11$. Using Tree-structured Parzen Estimator instead of Grid Search and the Random Search | [P24] |
| | $C7$. Choosing an optimal feature set for the model | $S12$. Using the DNN algorithm | [P20] |
| | | $S13$. Using k-max pooling instead of 1-max pooling | [P14] |
| | $C8$. Adversarial attacks | $S14$. Building DL models trained with adversarial patterns | [P5] |
| | $C9$. Limitations of static and dynamic analysis techniques | $S15$. Using byte-level representation | [P36] |
| | $C10$. Misdetection of malware with obfuscated code | $S16$. Using the API method calls | [P12] |
| | | $S17$. Using various kinds of features | [P13] |
| | $C11$. Over-fitting problem | $S18$. Training a neural network with fewer parameters unless more parameters/layers are required | [P15] |
| Network traffic | $C12$. Encrypted traffic | $S19$. Using VPN API on mobile devices | [P31] |

presented or discussed challenges. We could extract these challenges and solutions from 19 identified articles.

Table 8 lists the challenges and the proposed solutions reported in these 19 primary studies for mobile malware detection. These observations are very important for researchers and practitioners because no paper has provided these challenges like this before. Each of these challenges can be addressed by researchers to pave the way for further research.

We classified these challenges into the following three main categories:

- *Dataset* A training dataset is required for building a machine learning model.
- *Model building* A machine learning model building process includes several steps, namely data collection, preparation of the data, algorithm selection, training, testing, parameter tuning (a.k.a., hyperparameter optimization), and prediction.
- *Network traffic* Network traffic-related features are represented in several mobile malware detection datasets and used in machine learning models.

Obtaining a sufficient and up-to-date training dataset is a common obstacle [P38]. Therefore, Yuan et al. [P38] suggested that researchers should contribute to public datasets [P38]. For multi-class classification problems such as the identification of malware families, the availability of training datasets becomes a more challenging problem [P10]. It is prevalent to face with an insufficient number of samples for some malware families [P10]. To cope with such a challenge, researchers can use imbalanced learning algorithms [30]. Even if a sufficient number of samples are available, wrong labels may become a problem when building a model [P18]. Therefore, Mercaldo and Santone [P18] proposed to check samples against well-known data sources, such as VirusTotal service [P18].

Training time is a common problem when building a DL model for malware detection [P4, P24, P32, P37, P39]. Zhong and Gu [P39] used parallel strategies and algorithms to cope with long computation time for DL models [P39]. Amin et al. [P4] preferred to use CNN instead of LSTM to decrease the computation time of the DL model [P4]. Wang et al. [P32] used deep autoencoder as a pre-training method of CNN to reduce the training time [P32]. On the other hand, D'Angelo et al. [P8] reported the use of autoencoder as a bottleneck for processing time and did not propose a solution for this problem [P8]. Yuan et al. [P37] observed that the backpropagation algorithm conducted over multiple hidden layers resulted in a time-consuming training process for CNN and MLP [P37]. To reduce the training time, they used a broad learning algorithm [P37]. Pektaş and Acarman [P24] used an early stopping criterion to shorten the time required for hyper-parameter optimization [P24]. The same researchers used a Grid Search algorithm to find the best combination of hyper-parameters in one of their earlier studies [P23]. They realized the high computational cost of Grid Search and used a Random Search to overcome this obstacle [P24]. Since Grid Search and Random Search explore hyper-parameter space randomly and blindly, they used Tree-Structured Parzen Estimator to optimize hyper-parameters intelligently [P24].

Choosing an optimal set of features for training is another crucial challenge reported in the primary studies [P14, P20]. Nguyen-Vu et al. [P20] relied on DNNs for selecting proper feature sets [P20]. Li et al. [P14] reported a shortcoming of max pooling for choosing features in a CNN [P14]. Max pooling cannot select multiple essential features and hence, may lose important information [P14]. Li et al. [P14] used k-max pooling for feature selection to deal with top-k important features [P14].

For the model building, other challenges should also be considered, such as adversarial attacks [P5], limitations of static and dynamic analysis [P36], obfuscation techniques [P12, P13], and over-fitting problem [P15]. Ananya et al. [P5] developed DL models trained with adversarial patterns to decrease the risk of adversarial attacks [P5]. Yuan et al. [P36] identified the limitations of static and dynamic analysis techniques and proposed a byte-level solution [P36]. Obfuscation techniques are another challenge for malware detectors [P12, P13]. Kim et al. [P13] reported the negative impact of obfuscation techniques on static analysis and proposed using various kinds of features to alleviate this negative impact [P13]. Karbab et al. [P12] specifically used API method calls to cope with obfuscated code [P12]. Mahdavifar and Ghorbani [P15] reported that building a neural network with many hidden layers and neurons in each layer causes the over-fitting problem [P15]. To solve this problem, they trained the neural network with more parameters, and thus, the training error was driven to a small value, while the error on new data was relatively large [P15]. In other words, the network memorized the training data, but it failed to generalize to new situations [P15].

Wang et al. [P31] identified encrypted traffic as a challenge for malware identification [P31]. To deal with encrypted traffic, they designed a system that leverages VPN API on mobile devices to provide full access to the network traffic of these devices and identify malware samples with HTTPs traffic [P31]. A forwarder in gateway transparently bridges packets on the VPN interface and payload data on the regular socket interface and forwards the traffic to the detection server for analysis [P31].

## 5 Discussion

In this section, we present the discussion related to each research question (Sect. 5.1), and also, we present the potential threats to validity (Sect. 5.2).

## 5.1 General discussion

There are several differences between the Systematic Literature Review (SLR) and Systematic Mapping Studies (SMS) studies [15]. While RQs of SMS studies are general, RQs of SLR studies are very specific. RQs drive the search process in SLR studies, and research topics in SMS studies drive the search process. All the relevant papers must be identified in SLR studies, and quality assessment is mandatory; however, quality assessment is optional, and all the relevant papers are not required in SMS studies. Since we have very specific research questions as part of this research, we aimed to perform an SLR study instead of an SMS study. However, researchers can also work on an SMS study on malware detection without limiting the techniques to deep learning.

Discussion per research question is presented as follows:

- *RQ-1* We observed that only 1 article used semi-supervised learning, and 1 article applied unsupervised learning together with supervised learning approaches. There was no article that applied reinforcement learning. For instance, Deep Recurrent Q-Network (DQN) was developed in 2015 by researchers of DeepMind and provided state-of-the-art results in different problems [21, 74]. The DQN algorithm uses deep learning and reinforcement learning and also applies the experience replay technique [51]. However, this algorithm has not been used in mobile malware detection problem yet. To this end, the integration of reinforcement learning with deep learning approaches and the application of these models on malware detection is considered as a future research direction. We did not encounter any article that applied a semi-supervised deep learning approach, and therefore, we can consider this research field as a potential research field because a lot of unlabeled data can be processed with the help of semi-supervised learning approaches and the time-consuming process of labeling can be relatively reduced. There are deep learning-based semi-supervised learning approaches [41, 67] that can be investigated for mobile malware detection. Similarly, unsupervised deep learning approaches can be investigated in detail [43, 44] as well. Similarly, Mahdavifar and Ghorbani [49] stated that the use of deep learning in cybersecurity is mostly about supervised learning, which assumes that there are labeled data points. However, the collection and labeling of the data points are time-consuming and expensive. As such, more research is needed on the application of deep unsupervised, deep semi-supervised, and deep reinforcement learning deep learning approaches in cybersecurity. The outcome of our SLR analysis is aligned with the outcome of Mahdavifar and Ghorbani [49].
- *RQ-2* We identified that while static features are mostly preferred by the researchers (i.e., 55%), the use of hybrid features is very limited (i.e., 13%). Since static features-based models are not resilient to the obfuscation techniques, the integration of dynamic features to the models are important. For instance, network traffic-related features can provide very valuable information; however, we noticed that this type of feature was not widely applied yet. The collection of dynamic features might be costly and time-consuming, but its benefit is beyond its limitations. To this end, we expect to see more research on the use of hybrid features in the future to develop resilient malware detection models.
- *RQ-3* We noticed that CNN and DNN algorithms were the most applied algorithms. LSTM and DBN algorithms were applied as the second and third most applied algorithms, respectively. It is interesting to see that the hybrid models were not preferred much (i.e., four studies) in these articles. We expect to see more research on the combination of deep learning models in the future because each deep learning algorithm has its own pros and cons, and therefore, the power of multiple algorithms can be more than the power of each individual deep learning algorithm in some cases. Also, we observed that some machine learning types such as multi-task learning, multi-modal learning, and multi-view learning were neglected, but they can be applied effectively in mobile malware detection problem.
- *RQ-4* We identified that accuracy and F-measure are widely used evaluation parameters, and the cross-validation is the most preferred validation approach. F-measure is also widely used in other prediction problems that have unbalanced datasets such as software fault prediction [17]. We observed that there is not a consensus on the evaluation parameter yet because most of these articles reported several experimental results with respect to the different evaluation parameters such as FPR and FNR [16]. While the AUC parameter is widely used in many prediction problems such as software fault prediction, it is not widely preferred in malware detection articles. This indicates that more research is required for convergence across studies and for improving the confidence in experimental studies. Lessmann et al. [42] presented such a framework for software defect prediction studies because there was three potential bias at that time, namely the use of a few proprietary datasets, conceptually inaccurate evaluation parameters, and inappropriate statistical testing procedures. A similar framework for benchmarking mobile malware detection

studies can be beneficial to decide on the best performing algorithm.

- *RQ-5* DNN was reported as the best algorithm in terms of predictive accuracy in 11 articles. The CNN algorithm was reported as the best algorithm in 8 studies. This shows that most of the researchers are still applying algorithms (i.e., DNN) similar to the traditional ANN algorithms because they have prior experience in these traditional algorithms. This might be limiting the number of DL algorithms used in the experiments to reach a conclusion on the best performing algorithm. Based on our paper pool, out of the 11 studies reported DNN as the best performing algorithm, nine studies compared DNN only with traditional ML algorithms. Two studies analyzed DNN's performance against other DL algorithms: Pektaş and Acarman [P23] against a Hybrid DL model, CNN, and RNN/LTSM,Pektaş and Acarman [P24] against CNN. Out of the eight studies that reported CNN as the best performing, none of them compared CNN's performance against another DL algorithm. If a framework can be set up for benchmarking of algorithms, potential bias can be minimized, and all the algorithms can be evaluated in a similar manner. Since all the algorithms were not evaluated in an article in detail, it is difficult to state that the best algorithm for mobile malware detection is DNN. We expect to see more research on the benchmarking of deep learning algorithms in the future.
- *RQ-6* We observed that most of the studies (i.e., 32) did not mention the feature selection technique. When the high-performance of the models is considered, this observation indicates that the feature selection component is not required for deep learning-based malware detection models. The use of feature selection techniques is also related to the number of features, and therefore, we can conclude that the number of features is not too much.
- *RQ-7* We identified several public datasets hosted on different websites. On the other hand, we also observed that some of them are not actively maintained, and some links are not active. For sustainable and repeatable experiments, the maintenance of these datasets for malware detection is important, and researchers who build these datasets should continue to maintain them for further research. For better performance, deep learning algorithms require a huge amount of data, and therefore, we expect to see very large-scale datasets in the future for better mobile malware detection models.

  Another deficiency we observed in experimental design is the number of datasets used in experiments to reach a conclusion on the best performing algorithm. Since the performance of ML/DL algorithms is heavily based on training, testing and validation data, it is important to use various datasets when comparing the performance of ML/DL algorithms. Out of the eight studies that reported CNN as the best performing, six of them used only one dataset in their experiments. Four studies out of 11, reported DNN as the best performing DL algorithm using only one dataset. We propose the use of more datasets in experimental design to reach more reliable results on the performance of ML/DL algorithms in mobile malware detection.
- *RQ-8* We noticed that 14 studies did not specify the deep learning implementation platform. For repeatable, improvable, and refutable experiments [25], it is also crucial to report not only the algorithms and parameters but also the implementation platform because the implementation of the algorithms might be different across platforms and some implementations might be optimized whereas some of them are not.
- *RQ-9* We also identified several research challenges and gaps in mobile malware detection. These challenges and research gaps were reported based on the identified papers and the analysis performed by the researchers in the articles. As such, there might be some challenges and research gaps that were not discussed by the original authors of the articles, and we might have skipped those challenges.

## 5.2 Potential threats to validity

Validity considerations are applicable for SMS and SLR studies similar to empirical studies [57, 58]. The threats to the validity of this SLR are mainly related to the specification of the candidate pool of papers, primary study selection bias, data extraction, and data synthesis.

The candidate pool of papers has been specified by searching online databases using keywords. We used broad terms to form search keywords to decrease the risk of excluding potentially relevant studies. Besides, we did not include a keyword for mobile applications and hence aimed at obtaining all malware detection studies using DL regardless of their platform, i.e., mobile, desktop. With this approach, we decreased precision and increased recall hence obtained more candidate papers to be assessed for specifying the final set of primary studies. Also, we searched for five widely used online databases in SMS and SLR studies in computer science and software engineering. To mitigate the risk of missing some relevant studies, we also conducted both backward and forward snowballing. We believe that an adequate pool of candidate papers has been formed for this study, and if there is any missing journal paper, the rate will be negligible.

Since we searched in the widely used computer science databases and performed forward & backward snowballing during this research, we believe that we covered the deep learning-based articles published so far. However, there might be some articles that were not published in high-quality journals and not indexed in these databases, and therefore, they might not be included in this SLR study. The snowballing step of this research also helped us to double-check whether we covered all the deep learning-based articles. Our main objective was to present the state-of-the-art of the deep learning-based mobile malware detection studies, and as such, we did not include articles that do not apply deep learning or do not present experiments on the datasets related to the mobile systems.

Application of exclusion criteria is subject to researchers' bias and a potential threat to validity. The authors built a comprehensive list of exclusion criteria (Table 2) and used the think-aloud application of exclusion criteria [2] to mitigate the risk of ambiguous interpretations. In addition, the authors selected primary studies using a joint voting mechanism. All of the conflicts have been recorded and resolved via discussions among the authors.

The validity of the data extraction is another essential aspect, which directly affects the results of this study. To ensure the correctness of the extracted data, the authors formed categories iteratively and incrementally. They aimed at decreasing the risk of researcher bias via mapping the relevant data in primary studies to the specified groups. Whenever an author was undecided about the data to be extracted, he recorded that case, and these cases were resolved via discussions among the authors. Since we mainly used descriptive statistics to answer the RQs, we think that threats to internal validity are relatively small.

## 6 Conclusion and future work

With the broad application of mobile computing, malware has become a serious problem that can have a dramatic impact on systems. Together with the advancements in computing, deep learning solutions have become feasible for both effective and efficient detection of malware. Several solutions have been provided in the literature with their specific benefits, but also with the open challenges. To identify these solutions and make explicit the open challenges, in this study, we have presented the results of a systematic review in which we have reviewed 40 articles that applied deep learning algorithms for mobile malware detection. Compared to the existing secondary studies in this domain, this is the first study that adopts a systematic review approach on malware detection using deep learning for different platforms.

In the SLR, we synthesized the results from the identified primary studies based on machine learning categories, data sources/features, deep learning approaches, evaluation parameters & validation techniques, best performing algorithms, feature selection techniques, datasets, and implementation platforms. Further, we derived the challenges and research directions. We identified the following research directions to pave the way for further research and explained them in the Discussion section:

1. The development of unsupervised deep learning approaches for malware detection
2. The development of semi-supervised deep learning models for malware detection
3. The development of reinforcement learning-based deep learning techniques for malware detection
4. More hybrid features-based techniques
5. A framework for better benchmarking of deep learning-based approaches and better experimental design to compare more DL algorithms to reach more general results
6. Better maintenance of malware datasets and sustainable platforms for hosting malware datasets
7. More datasets should be used in experimental design to compare the performance of ML/DL algorithms to reach more reliable results

The results of the study can be beneficial for both researchers and practitioners. Research can identify the key research and solution directions, while practitioners can use the result to focus on best-practices. As future work, we aim to focus on these research directions. In particular, we are planning to develop multi-modal deep learning-based and semi-supervised deep learning techniques for mobile malware detection.

**Funding** No funding was received for conducting this study.

## Declarations

**Conflict of interest** The authors have no relevant financial or non-financial interests to disclose.

## References

### This section is divided into two parts: (1) Regular references cited throughout the paper and (2) Citations to the primary studies reviewed in the SLR

1. Ab Razak MF, Anuar NB, Salleh R, Firdaus A (2016) The rise of "malware": bibliometric analysis of malware study. J Netw Comput Appl 75:58–76
2. Ali NB, Petersen K (2014) Evaluating strategies for study selection in systematic literature studies. In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement pp. 1–4
3. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Durumeric Z, Halderman JA, Invernizzi L, Kallitsis M, Kumar D (2017) Understanding the mirai botnet. In: 26th {USENIX} security symposium ({USENIX} Security 17) pp. 1093–1110
4. AppBrain, "Number of Android apps on Google Play." [Online]. Available: https://www.appbrain.com/stats/number-of-android-apps. [Accessed: 17-July-2020].
5. Aslan ÖA, Samet R (2020) A comprehensive review on malware detection approaches. IEEE Access 8:6249–6271
6. Baltrušaitis T, Ahuja C, Morency LP (2018) Multimodal machine learning: a survey and taxonomy. IEEE Trans Pattern Anal Mach Intell 41(2):423–443
7. Bazrafshan Z, Hashemi H, Fard SMH, Hamzeh A (2013) A survey on heuristic malware detection techniques. In: The 5th conference on information and knowledge technology IEEE, pp. 113–120
8. Berman DS, Buczak AL, Chavis JS, Corbett CL (2019) A survey of deep learning methods for cyber security. Information 10(4):122
9. Brownlee J (2016) Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras. Machine Learning Mastery, Vermont
10. Brownlee J (2017) Long Short-term memory networks with Python: develop sequence prediction models with deep learning. Machine Learning Mastery, Vermont
11. Brownlee J (2019) Deep learning for computer vision: image classification, object detection, and face recognition in Python. Machine Learning Mastery, Vermont
12. Budgen D, Brereton P, Drummond S, Williams N (2018) Reporting systematic reviews: some lessons from a tertiary study. Inf Softw Technol 95:62–74
13. Carlin D, Burgess J, O'Kane P, Sezer S (2019) You could be mine (d): the rise of cryptojacking. IEEE Secur Priv 18(2):16–22
14. Catal C (2012) On the application of genetic algorithms for test case prioritization: a systematic literature review. In: Proceedings of the 2nd international workshop on Evidential assessment of software technologies pp. 9–14.
15. Catal C, Mishra D (2013) Test case prioritization: a systematic mapping study. Software Qual J 21(3):445–478
16. Catal C, Sevim U, Diri B (2010) Metrics-driven software quality prediction without prior fault data. In: Ao SI, Gelman L (eds) Electronic Engineering and Computing Technology. Springer, Dordrecht, pp 189–199
17. Choudhary GR, Kumar S, Kumar K, Mishra A, Catal C (2018) Empirical analysis of change metrics for software fault prediction. Comput Electr Eng 67:15–24
18. Cui Z, Xue F, Cai X, Cao Y, Wang GG, Chen J (2018) Detection of malicious code variants based on deep learning. IEEE Trans Industr Inf 14(7):3187–3196
19. Darabian H, Dehghantanha A, Hashemi S, Homayoun S, Choo KKR (2020) An opcode-based technique for polymorphic Internet of Things malware detection. Concurr Comput Practice Exp 32(6):e5173
20. Deng L (2014) A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Trans Signal Inform Process. https://doi.org/10.1017/atsip.2013.9
21. Du Z, Miao Q, Zong C (2020) Trajectory planning for automated parking systems using deep reinforcement learning. Int J Automot Technol 21(4):881–887
22. Elkahky AM, Song Y, He X (2015) A multi-view deep learning approach for cross domain user modeling in recommendation systems. In Proceedings of the 24th international conference on world wide web pp. 278–288
23. Farfade SS, Saberian MJ, Li LJ (2015) Multi-view face detection using deep convolutional neural networks. In: Proceedings of the 5th ACM on international conference on multimedia retrieval pp. 643–650
24. Feizollah A, Anuar NB, Salleh R, Wahab AWA (2015) A review on feature selection in mobile malware detection. Digit Investig 13:22–37
25. Gay G, Menzies T, Cukic B, Turhan B (2009) How to build repeatable experiments. In: Proceedings of the 5th international conference on predictor models in software engineering pp. 1–9
26. Gibert D, Mateu C, Planes J (2020) The rise of machine learning for detection and classification of malware: research developments, trends and challenges. J Network Comput Appl 153:102526
27. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Bengio Y (2014) Generative adversarial nets. Adv Neural Info Process Syst 27
28. Griffin K, Schneider S, Hu X, Chiueh TC (2009) Automatic generation of string signatures for malware detection. In: International workshop on recent advances in intrusion detection. Springer, Berlin, Heidelberg. pp. 101–120
29. Hassler E, Carver JC, Kraft NA, Hale D (2014) Outcomes of a community workshop to identify and rank barriers to the systematic literature review process. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. pp. 1–10
30. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
31. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition pp. 770–778
32. Hsiao SC, Kao DY, Liu ZY, Tso R (2019) Malware image classification using one-shot learning with Siamese networks. Proced Comput Sci 159:1863–1871
33. Jerome Q, Allix K, State R, Engel T (2014) Using opcode-sequences to detect malicious Android applications. In: 2014 IEEE international conference on communications (ICC) IEEE. pp. 914–919
34. Kamilaris A, Prenafeta-Boldú FX (2018) Deep learning in agriculture: a survey. Comput Electron Agric 147:70–90

35. Kitchenham BA, Dyba T, Jorgensen M (2004) Evidence-based software engineering. In: Proceedings. 26th international conference on software engineering IEEE. pp. 273–281
36. Kitchenham B, Pretorius R, Budgen D, Brereton OP, Turner M, Niazi M, Linkman S (2010) Systematic literature reviews in software engineering–a tertiary study. Inf Softw Technol 52(8):792–805
37. Kitchenham B, Pearl Brereton O, Budgen D, Turner M, Bailey J, Linkman S (2009) Systematic literature reviews in software engineering—a systematic literature review. Inf Softw Technol 51(1):7–15. https://doi.org/10.1016/j.infsof.2008.09.009
38. Kok SH, Abdullah A, Jhanjhi NZ, Supramaniam M (2019) Ransomware, threat and detection techniques: a review. Int J Comput Sci Network Secur 19(2):136
39. Kolias C, Kambourakis G, Stavrou A, Voas J (2017) DDoS in the IoT: mirai and other botnets. Computer 50(7):80–84
40. Kouliaridis V, Barmpatsalou K, Kambourakis G, Chen S (2020) A survey on mobile malware detection techniques. IEICE Trans Inf Syst 103(2):204–211
41. Kuznietsov Y, Stuckler J, Leibe B (2017) Semi-supervised deep learning for monocular depth map prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition pp. 6647–6655
42. Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng 34(4):485–496
43. Li R, Wang S, Long Z, Gu D (2018) Undeepvo: monocular visual odometry through unsupervised deep learning. In: 2018 IEEE international conference on robotics and automation (ICRA) IEEE, pp. 7286–7291
44. Li Y, Yang M, Zhang Z (2018) A survey of multi-view representation learning. IEEE Trans Knowl Data Eng 31(10):1863–1883
45. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Jeroen Van Der, Laak JA, Van Ginneken B, Sánchez CI (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88
46. Liu K, Xu S, Xu G, Zhang M, Sun D, Liu H (2020) A review of android malware detection approaches based on machine learning. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3006143
47. Liu X, Liu J (2014) A two-layered permission-based android malware detection scheme. In: 2014 2nd IEEE international conference on mobile cloud computing, services, and engineering IEEE, pp. 142–148
48. Maggiori E, Tarabalka Y, Charpiat G, Alliez P (2016) Convolutional neural networks for large-scale remote-sensing image classification. IEEE Trans Geosci Remote Sens 55(2):645–657
49. Mahdavifar S, Ghorbani AA (2019) Application of deep learning to cybersecurity: a survey. Neurocomputing 347:149–176
50. McLaughlin N, Martinez del Rincon J, Kang B, Yerima S, Miller P, Sezer S, Safaei Y, Trickel E, Zhao Z, Doupé A, Joon Ahn G (2017) Deep android malware detection. In: Proceedings of the seventh ACM on conference on data and application security and privacy. pp. 301–308
51. Miles MB, Huberman AM, Saldana J (2014) Qualitative data analysis: a methods sourcebook, 3rd edn. SAGE Publications Inc., Thousand Oaks, CA
52. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
53. Nataraj L, Karthikeyan S, Jacob G, Manjunath BS (2011) Malware images: visualization and automatic classification. In: Proceedings of the 8th international symposium on visualization for cyber security. pp. 1–7
54. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp 689–696
55. Oussidi A, Elhassouny A (2018) Deep generative models: survey. In: 2018 international conference on intelligent systems and computer vision (ISCV). IEEE, pp. 1–8
56. Pan Y, Ge X, Fang C, Fan Y (2020) A systematic literature review of android malware detection using static analysis. IEEE Access 8:116363–116379
57. Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. In: 12th international conference on evaluation and assessment in software engineering (EASE) 12. pp. 1–10
58. Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: an update. Inf Softw Technol 64:1–18
59. Pouyanfar S, Sadiq S, Yan Y, Tian H, Tao Y, Reyes MP, Shyu ML, Chen SC, Iyengar SS (2018) A survey on deep learning: algorithms, techniques, and applications. ACM Comput Surv (CSUR) 51(5):1–36
60. Qamar A, Karim A, Chang V (2019) Mobile malware attacks: review, taxonomy & future directions. Futur Gener Comput Syst 97:887–909
61. Salakhutdinov R, Hinton G (2009) Deep boltzmann machines. In: Artificial intelligence and statistics. pp 448–455. PMLR
62. Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y (2012) “Andromaly”: a behavioral malware detection framework for android devices. J Intell Inform Syst 38(1):161–190
63. Shabtai A, Moskovitch R, Elovici Y, Glezer C (2009) Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey. Inform Secur Tech Rep 14(1):16–29
64. Sohn K, Shang W, Lee H (2014) Improved multimodal deep learning with variation of information. Adv Neural Inform Process Syst 27:2141–2149
65. Souri A, Hosseini R (2018) A state-of-the-art survey of malware detection approaches using data mining techniques. HCIS 8(1):3
66. Suresh S, Di Troia F, Potika K, Stamp M (2019) An analysis of Android adware. J Comput Virol Hacking Tech 15(3):147–160
67. Tarhan A, Giray G (2017) On the use of ontologies in software process assessment: a systematic literature review. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering. pp. 2–11
68. Tarvainen A, Valpola H (2017) Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results. Adv Neural Inf Process Syst 30
69. Tummers J, Kassahun A, Tekinerdogan B (2019) Obstacles and features of farm management information systems: a systematic literature review. Comput Electron Agric 157:189–204. https://doi.org/10.1016/j.compag.2018.12.044
70. Ucci D, Aniello L, Baldoni R (2019) Survey of machine learning techniques for malware analysis. Comput Secur 81:123–147
71. Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering—EASE ’14, 1–10. Doi: https://doi.org/10.1145/2601248.2601268
72. Ye Y, Chen L, Hou S, Hardy W, Li X (2018) DeepAM: a heterogeneous deep learning framework for intelligent malware detection. Knowl Inf Syst 54(2):265–285
73. Ye Y, Li T, Adjeroh D, Iyengar SS (2017) A survey on malware detection using data mining techniques. ACM Comput Surv (CSUR) 50(3):1–40

74. Yuxin D, Siyi Z (2019) Malware detection based on deep learning algorithm. Neural Comput Appl 31(2):461–472
75. Zeng J, Hu J, Zhang Y (2018) Adaptive traffic signal control with deep recurrent Q-learning. In: 2018 IEEE intelligent vehicles symposium (IV), IEEE, pp. 1215–1220
76. Zhang C, Patras P, Haddadi H (2019) Deep learning in mobile and wireless networking: a survey. IEEE Commun Surv Tutor 21(3):2224–2287
77. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. Adv Neural Inf Process Syst 28:649–657

## Primary studies (sources reviewed in the slr)

P1. Alotaibi A (2019) Identifying malicious software using deep residual long-short term memory. IEEE Access 7:163128–163137
P2. Alzaylaee MK, Yerima SY, Sezer S (2020) DL-Droid: deep learning based android malware detection using real devices. Comput Secur 89:101663
P3. Amin M, Shah B, Sharif A, Ali T, Kim KL, Anwar S (2019) Android malware detection through generative adversarial networks. Trans Emerg Telecommun Technol. https://doi.org/10.1002/ett.3675
P4. Amin M, Tanveer TA, Tehseen M, Khan M, Khan FA, Anwar S (2020) Static malware detection and attribution in android bytecode through an end-to-end deep system. Futur Gener Comput Syst 102:112–126
P5. Ananya A, Aswathy A, Amal TR, Swathy PG, Vinod P, Mohammad S (2020) SysDroid: a dynamic ML-based android malware analyzer using system call traces. Cluster Comput 23:2789–2808
P6. Bakhshinejad N, Hamzeh A (2019) Parallel-CNN network for malware detection. IET Inf Secur 14(2):210–219
P7. Chen T, Mao Q, Lv M, Cheng H, Li Y (2019) DroidVecDeep: android malware detection based on Word2Vec and deep belief network. TIIS 13(4):2180–2197
P8. D'Angelo G, Ficco M, Palmieri F (2020) Malware detection in mobile environments based on autoencoders and API-images. J Parallel Distrib Comput 137:26–33
P9. De Lorenzo A, Martinelli F, Medvet E, Mercaldo F, Santone A (2020) Visualizing the outcome of dynamic analysis of Android malware with VizMal. J Inform Secur App 50:102423
P10. Dharmalingam VP, Palanisamy V (2020) A novel permission ranking system for android malware detection—the permission grader. J Ambient Intell Human Comput 12:5071–5081
P11. Jan S, Ali T, Alzahrani A, Musa S (2018) Deep convolutional generative adversarial networks for intent-based dynamic behavior capture. Int J Eng Technol 7(4.29):101–103
P12. Karbab EB, Debbabi M, Derhab A, Mouheb D (2018) MalDozer: automatic framework for android malware detection using deep learning. Digit Investig 24:S48–S59
P13. Kim T, Kang B, Rho M, Sezer S, Im EG (2018) A multimodal deep learning method for android malware detection using various features. IEEE Trans Inf Forensics Secur 14(3):773–788
P14. Li D, Zhao L, Cheng Q, Lu N, Shi W (2019) Opcode sequence analysis of Android malware by a convolutional neural network. Concurr Comput: Practice Exp 32:e5308
P15. Mahdavifar S, Ghorbani AA (2020) DeNNeS: deep embedded neural network expert system for detecting cyber attacks. Neural Comput Appl 32:14753–14780
P16. Martín A, Rodríguez-Fernández V, Camacho D (2018) CANDYMAN: classifying android malware families by modelling dynamic traces with Markov chains. Eng Appl Artif Intell 74:121–133
P17. Martinelli F, Marulli F, Mercaldo F (2017) Evaluating convolutional neural network for effective mobile malware detection. Proced Comput Sci 112:2372–2381
P18. Mercaldo F, Santone A (2020) Deep learning for image-based mobile malware detection. J Comput Virol Hacking Tech 16:157–171
P19. Nauman M, Tanveer TA, Khan S, Syed TA (2018) Deep neural architectures for large scale android malware analysis. Clust Comput 21(1):569–588
P20. Nguyen-Vu L, Ahn J, Jung S (2019) Android fragmentation in malware detection. Comput Secur 87:101573
P21. Pei X, Yu L, Tian S (2020) AMalNet: a deep learning framework based on graph convolutional networks for malware detection. Comput Secur 93:101792
P22. Pei X, Yu L, Tian S, Wang H, Peng Y (2020) Combining multi-features with a neural joint model for Android malware detection. J Intell Fuzzy Syst (Preprint) 38:2151–2163
P23. Pektaş A, Acarman T (2020) Learning to detect Android malware via opcode sequences. Neurocomputing 396:599–608
P24. Pektaş A, Acarman T (2020) Deep learning for effective Android malware detection using API call graph embeddings. Soft Comput 24(2):1027–1043
P25. Saif D, El-Gokhy SM, Sallam E (2018) Deep belief networks-based framework for malware detection in android systems. Alex Eng J 57(4):4049–4057
P26. Sharmeen S, Huda S, Abawajy J, Hassan MM (2020) An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches. Appl Soft Comput 89:106089
P27. Shi-qi L, Bo N, Ping J, Sheng-wei T, Long Y, Rui-jin W (2019) Deep learning in Drebin: android malware image texture median filter analysis and detection. KSII Trans Internet Inform Syst (TIIS) 13(7):3654–3670
P28. Su X, Shi W, Qu X, Zheng Y, Liu X (2020) DroidDeep: using Deep Belief Network to characterize and detect android malware. Soft Comput 24:6017–6030
P29. Tang M, Qian Q (2018) Dynamic API call sequence visualisation for malware classification. IET Inf Secur 13(4):367–377
P30. Vinayakumar R, Soman KP, Poornachandran P, Sachin Kumar S (2018) Detecting Android malware using long short-term memory (LSTM). J Intell Fuzzy Syst 34(3):1277–1288
P31. Wang S, Chen Z, Yan Q, Ji K, Peng L, Yang B, Conti M (2020) Deep and broad URL feature mining for android malware detection. Inf Sci 513:600–613
P32. Wang W, Zhao M, Wang J (2019) Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. J Ambient Intell Humaniz Comput 10(8):3035–3043
P33. Xiao X, Wang Z, Li Q, Xia S, Jiang Y (2016) Back-propagation neural network on Markov chains from system call sequences: a new approach for detecting Android malware with system call sequences. IET Inf Secur 11(1):8–15
P34. Xiao X, Zhang S, Mercaldo F, Hu G, Sangaiah AK (2019) Android malware detection based on system call sequences and LSTM. Multimed Tools Appl 78(4):3979–3999
P35. Yen YS, Sun HM (2019) An android mutation malware detection based on deep learning using visualization of importance from codes. Microelectron Reliab 93:109–114
P36. Yuan B, Wang J, Liu D, Guo W, Wu P, Bao X (2020) Byte-level malware classification based on markov images and deep learning. Comput Secur 92:101740

P37. Yuan W, Jiang Y, Li H, Cai M (2019) A lightweight on-device detection method for android malware. IEEE transactions on systems, man, and cybernetics: systems

P38. Yuan Z, Lu Y, Xue Y (2016) Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci Technol 21(1):114–123

P39. Zhong W, Gu F (2019) A multi-level deep learning system for malware detection. Expert Syst Appl 133:151–162

P40. Zhou Q, Feng F, Shen Z, Zhou R, Hsieh MY, Li KC (2019) A novel approach for mobile malware classification and detection in Android systems. Multimed Tools Appl 78(3):3529–3552