



A review of deep learning models to detect malware in Android applications

Elliot Mbunge^{a,b,*}, Benhildah Muchemwa^a, John Batani^c, Nobuhle Mbuyisa^a

^a Department of Computer Science, Faculty of Science and Engineering, University of Eswatini, Private Bag 4 Kwaluseni, Eswatini

^b Department of Information Technology, Faculty of Accounting and Informatics, Durban University of Technology, P O Box 1334, Durban 4000, South Africa

^c Faculty of Engineering and Technology, Botho University, Maseru 100, Lesotho

ARTICLE INFO

Keywords:

Malicious software
Android applications
Detection
Deep learning
Smartphones

ABSTRACT

Android applications are indispensable resources that facilitate communication, health monitoring, planning, data sharing and synchronization, social interaction, business and financial transactions. However, the rapid increase in the smartphone penetration rate has consequently led to an increase in cyberattacks. Smartphone applications use permissions to allow users to utilize different functionalities, making them susceptible to malicious software (malware). Despite the rise in Android applications' usage and cyberattacks, the use of deep learning (DL) models to detect emerging malware in Android applications is still nascent. Therefore, this review sought to explain DL models that are applied to detect malware in Android applications, explore their performance as well as identify emerging research gaps and present recommendations for future work. This study adopted the preferred reporting items for systematic reviews and meta-analyses (PRISMA) guidelines to guide the review. The study revealed that convolutional neural networks, gated recurrent neural networks, deep neural networks, bidirectional long short-term memory, long short-term memory (LSTM) and cubic-LSTM are the most prominent deep learning-based malicious software detection models in Android applications. The findings show that deep learning models are increasingly becoming an effective technique for malicious software detection in Android applications in real-time. However, monitoring and tracking information flow and malware behavior is a daunting task because of the evolving nature of malware and human behavior. Therefore, training mobile application users and sharing updated malware datasets is paramount in developing detection models. There is also a need to detect malicious software before downloading mobile applications to improve the security of Android smartphones.

1. Introduction

The increased mobile phone penetration rate facilitates the development and deployment of mobile applications in various domains [1]. Mobile phones, especially smartphones, facilitate communication [2], health monitoring [3,4], support teaching and learning, planning, data sharing and synchronization, social interaction, business, and financial transactions. As smartphones become ubiquitous and pervasive, they also become vulnerable to cyberattacks and malicious users. For instance, between 2016 and 2020, approximately 218 billion mobile applications ("apps") were downloaded, which is an increase of over 50% from 140.7 billion mobile apps downloaded in 2016 [5]. This consists of Android, BlackBerry, iPhone, and Symbian apps, among others. Among other platforms, Android-based mobile applications recorded the highest number of downloads, installed on over 1.5 billion mobile devices in 2021 [6]. The official Android play store presently has at least 2.6 million apps which can be downloaded and installed for different purposes [7].

However, the rise in mobile app usage makes mobile phones vulnerable to cyberattacks and threats such as malware (Trojan horses, viruses, worms, and spyware, among others). Such threats violate data integrity, system or device availability, and data confidentiality [6]. Due to popularity, third-party code and openness, Android applications tend to be vulnerable to various types of malware. For instance, at least 3.25 million Android applications were infected with malicious software in 2016 [8]. In response to these attacks, several methods, including the dynamic taint analysis mechanism, have been applied to detect malware. Dynamic taint analysis monitors and tracks information flow leakage at runtime [9]. This approach is error-prone because of the greater possibility of missing some important security flaws and emerging malware variants. These approaches detect malware using feature extraction analysis, static, hybrid analysis methods and dynamic analysis. Static analysis gathers features of the Android application by using packer tools like VMprotect and UPX and analyses features to detect malware without executing the Android package kit file [10]. Gener-

* Corresponding author.

E-mail address: mbungeelliot@gmail.com (E. Mbunge).

<https://doi.org/10.1016/j.csa.2023.100014>

Received 14 April 2022; Received in revised form 13 December 2022; Accepted 11 February 2023

Available online 12 February 2023

2772-9184/© 2023 The Authors. Published by Elsevier B.V. on behalf of KeAi Communications Co., Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

ally, the Android package kit file contains information such as operating systems API calls, opcodes, and network addresses, among others. It is difficult to detect malware that uses dynamic code loading using static analysis [11]. However, some scholars applied dynamic analysis to monitor malicious software behavior by tracking information flow during the executing stage [12]. This method encounters some impediments, such as the high computational overhead required to monitor both the process and information flow [10], as well as the maximum number of processes that can be monitored and tracked in real-time while executing user's commands. To improve malware detection accuracy, some scholars combined static and dynamic analysis to develop hybrid malware detection approaches [11]. Despite its outstanding accuracy, hybrid analysis is time-consuming and computationally intensive when extracting malware behaviours, monitoring information flow and collecting static features of malware from the APK file.

To alleviate the challenges associated with classical malware detection approaches, several scholars, including [13–15] applied various machine learning algorithms like support vector machines (SVM), Naïve Bayesian networks, random forest (RF), multilayer perceptron, and decision trees to detect malware in Android applications. Several review studies conducted by [16–18] analyzed machine learning-based malware detection models using training, testing and validation data extracted from Android applications. The extracted features include static features, required permissions, sensitive application programming interfaces (APIs), static data flow, suspicious behaviours, and network flows to detect malware [19]. However, machine learning algorithms contain shallow structured architectures that can address simple and well-constrained classification and clustering problems [20]. Malware apps have been rapidly increasing, making it difficult to detect using machine learning techniques due to the advanced nature and complexity of the attacks and threats. This calls for innovative, proactive and adaptive solutions [21] to identify and detect new variants of malicious software in Android apps.

1.1. Contribution of the study

Cyberattacks have exponentially burgeoned with the rise in human reliance on mobile phones [22]. Accurately detecting emerging malware in Android apps using machine learning models is increasingly becoming difficult due to various factors including (i) limited or outdated datasets [19], (ii) complexities and diversity of malware [23], and (iii) sub-optimal feature extraction [23]. Malware applications have been continuously evolving, making it difficult to detect using classical protection mechanisms due to the sophisticated nature and complexity of attacks and threats. Furthermore, malware applications are heterogeneous, that is, attacks vary depending on the target, the type of service exploited, the spreading source, and the location of the target [24]. Therefore, the accuracy of malware detection models depends on various factors, including proper training of the algorithm, feature extraction, understanding of abstract behavior, and attacking patterns. This requires robust intelligent-based malware detection models such as deep learning with highly structured architectures that can learn and analyze the longer sequence of malware patterns and behaviours from various huge datasets [20]. Deep learning models can extract meaningful insights and analyze huge datasets through a higher level of abstraction and semantic knowledge learning. However, the use of deep learning models to detect emerging malware in Android applications is still nascent. Therefore, this comprehensive review contributes to this nascent research area and aims to:

- (i) Identify and explain DL-based malicious software detection models in Android applications.
- (ii) Analyze the performance of DL models used to detect malware in Android applications.
- (iii) Identify dataset sources used to detect malicious software in Android applications.

- (iv) Identify emerging research gaps in deep learning-based malware detection models in Android applications.

Hereafter, this paper is structured as follows: Section 2 discusses the materials and methods adopted in carrying out the review. Section 3 presents deep learning-based malicious software detection models and their respective performance and the source of the datasets that were used on the models. Section 4 discusses the recommendation for detecting malware in Android applications. Lastly, Section 5 presents the conclusion of the study.

2. Materials and methods

The researchers adopted the PRISMA approach [25] to search and select relevant papers. Various search keywords were used to search for relevant publications from different online public repositories. The researchers searched papers from different reputable and prominent databases and applied inclusion and exclusion criteria to select relevant papers as illustrated in Fig. 1.

2.1. Search strategy

The study applied search keywords such as “deep learning” OR “deep learning techniques” OR “Deep neural networks” AND “malware detection” OR “detecting malware” OR “malicious software” AND “Android malware detection”, OR “Android applications”, OR “malware detection models in Android applications”. Relevant papers were retrieved from different online repositories such as Web of Science Google Scholar, Scopus, IEEE Xplore, Science Direct and Springer Link.

2.2. Inclusion and exclusion criteria

All papers/articles written in English or had English translation were included in this study. Commentaries, letters to the editor and preprint were not considered in this study. The articles' publication period was restricted to the period between January 2016 and March 2022, inclusive. Articles were excluded if the malware detection model used was not a deep learning model. All duplicate papers were not considered.

2.3. Screening process

The titles, abstracts and content of articles were screened as shown in Fig. 1. Data extracted from the selected articles include references, deep learning applied to detect malware, performance metrics, dataset source, and limitations or future work (see Table 1).

Fig. 1 shows the PRISMA steps following to search literature in various prominent electronic databases and select relevant papers based on the inclusion and criteria explained above. A total of twenty-five (25) papers were selected, as depicted in Table 1.

3. Results

Table 1 presents a summary of the study findings, showing the identified deep learning models, performance, dataset sources, and limitations /future work for each reviewed article. The identified deep learning models are deep belief networks, GRU, CNN, Bi-LSTM, LSTM, deep neural networks, and cubicLSTM.

a) Deep learning models applied to detect malicious software in Android applications

Deep learning (DL) techniques have been successfully used in various domains including fraud detection [21], security [47], object identification and detection [48], and malicious software detection, among others. DL is a subset of machine learning, mostly used in image processing, text classification and speech processing [49]. A deep learning model has numerous hierarchical layers and deeply structured architecture with several hidden layers of several connected artificial neurons

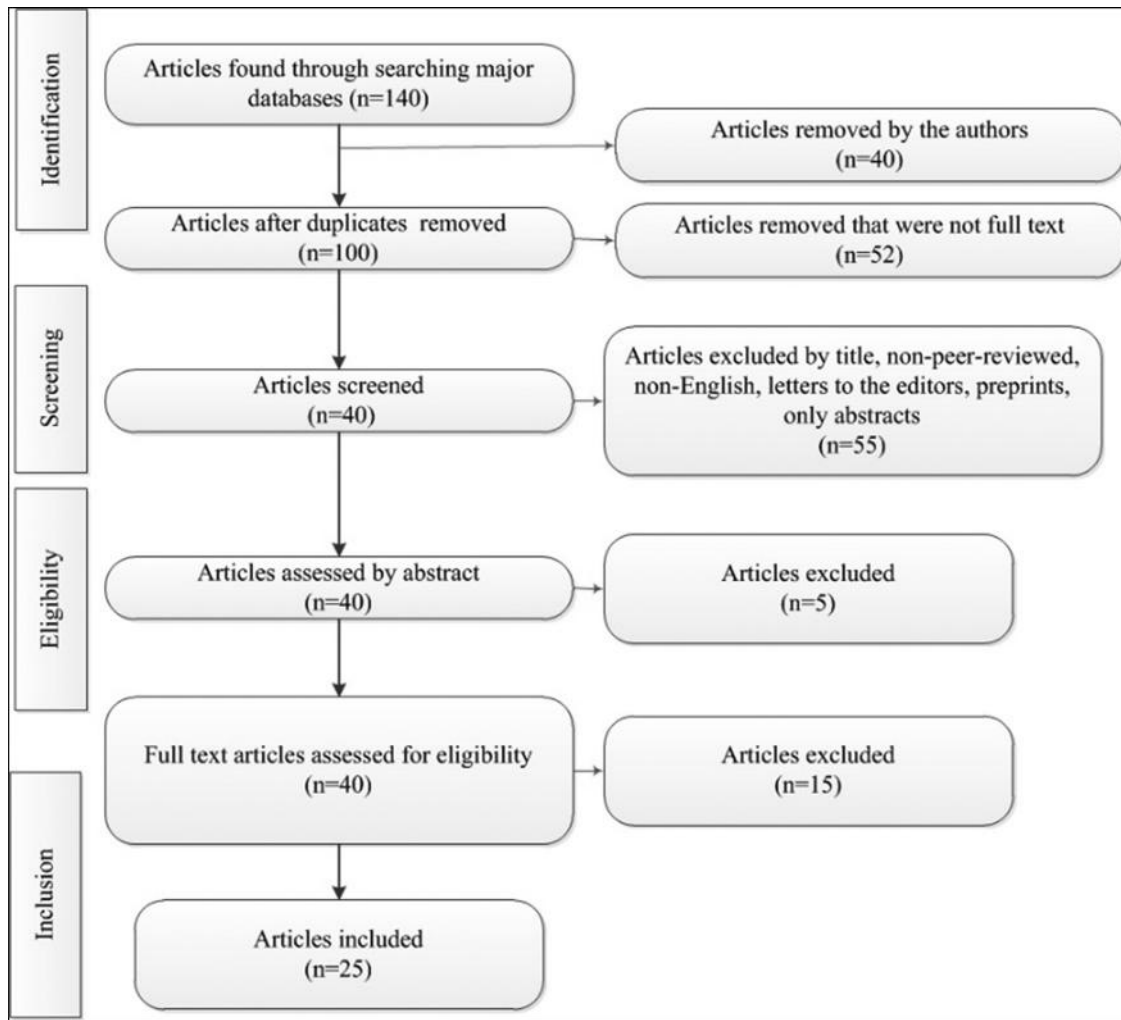


Fig. 1. PRISMA flowchart.

to process data [50]. Each layer of a deep neural network comprises many artificial neurons, each having its weights and potentially its activation functions that could be different from the ones on the other layers [51,52]. Though the weights can differ, the initial weights may be set randomly, or the same weight can be set across all the weights during initialization [53], after which the model will adjust them accordingly based on the error value. The outstanding feature of deep learning is its ability to automatically extract and abstract features, thus, eradicating the need for manual and tedious feature extraction hence, automatically identifying sophisticated and more useful high-order features [41,54]. Many deep learning models can be and have been applied to detect malware in Android applications. Table 1 shows the identified DL models, their performances in detecting malware in Android applications, and the sources of the datasets used.

Fig. 2 shows various DL models applied to detect malware. These models include deep belief networks, Gated Recurrent Units (GRU), CNN, DNN, LSTM, Bidirectional LSTM, Cubic LSTM, and hybrid models.

i Deep Belief Networks (DBNs)

According to [6], the proponent of the deep belief networks was Geoffrey Hinton in 2006. A deep belief network is a probability-based generative model composed of many layers of stochastic [55], latent variables with non-directed and symmetric links between the top two layers [56]. The lower layers of the model have direct links from the above layers with the arrows pointing towards the nearest layer to the data. Deep belief networks are trained in two stages: the pre-training phase

and fine-tuning (optimization) phase. During the pre-training stage, the RBM is trained layer by layer sequentially, starting from the bottom layer [57]. The RBM is a non-directed probability graph model with a layer each for observable and hidden variables [58]. Fine-tuning can be achieved using backpropagation, where the pre-trained Deep Belief Network is optimized in a supervised manner with labelled samples [59]. DBN offers a layer-by-layer learning approach for network initialization and automatic feature extraction, unlike traditional neural networks. However, it consumes system resources during the training stage and consumes time. A study conducted by [6] applied deep belief networks to detect malware using datasets from Android PRAGuard Dataset and VirusShare and achieved 95.79% precision, 97.62% recall and 96.82% accuracy. Also, [33] applied deep belief networks to detect malware using Contagio Community, Android Malware Genome Project and achieved a precision of 95.77%, recall of 97.84% and accuracy of 96.76%.

ii Gated Recurrent Unit (GRU)

The GRU is an enhanced version of recurrent neural networks (RNNs) which was introduced by [6], that is widely used to solve classification problems. The model solves the “vanishing gradient” problem that is inherently associated with standard RNNs [60]. The Gated Recurrent Unit uses two gates (update and reset gates) to handle and deal with the vanishing gradient problem associated with standard RNNs [6,61]. The gates are trainable in information retention from a long back, remove irrelevant information and pass relevant information down a chain of

Table 1
Deep learning models for detecting malicious software in Android applications.

Reference	Deep learning model	Performance	Dataset	Limitations/Future work
[26]	Convolutional neural networks (CNN)	Accuracy- 99.82% F1-score- 99.86% Recall- 99.91% Precision - 99.91%	VirusShare	The model requires more training time.
[22]	Deep Neural Networks (DNN)	Accuracy –93.4% F1-score- 93.2% Recall- 93.4% Precision- 93.5%	CICInvesAndMal2019 and CICAndMal2017	The model could not detect malware benign or malicious applications before downloading it.
[27]	CNN and LSTM	Accuracy- 95.83% Precision- 95.24% Recall- 96.15% F1-score- 95.69%	VirusTotal and Drebin	The model could not detect android malicious software samples based on hybridized image-based features.
[28]	CNN	Accuracy- 99.56%	VirusShare and Drebin	The dynamic analysis coverage and extraction of dynamic features need to be improved.
[29]	CNN	Accuracy- 91.27%	VirusShare	The model excluded obfuscated applications that could not extract application programming interface call graphs from Flowdroid.
[30]	DeepVisDroid (CNN)	Accuracy - 98.96%	Benign	The model could not identify some code camouflage.
[31]	Deep neural networks	Accuracy-98.86% F1-measure- 98.65% Recall - 98.47% Precision - 98.84%	Benign dataset, AMD Dataset, AndroZoo Dataset, Drebin Malware Collection Android PRAGuard Dataset and VirusShare	Dataset features were easily detectable.
[6]	Deep Belief Network-Gated Recurrent Unit	Precision – 95.79% Recall – 97.62% Accuracy– 96.82%		The dataset used was small. Dataset features were easily detectable. The calculation time for the hybrid model was more than the one for the separate models
[32]	CNN	Precision –96.3434% Recall – 96.335% F1-Score–96.333%	VirusShare, Malgenome, Drebin, Contagio Minidump	The model takes more time to calculate better results
[33]	Deep Belief Networks	Precision –95.77% Recall – 97.84% Accuracy –96.76%	Contagio Community, Android Malware Genome Project	The use of a small dataset will not produce better results and does not make better use of the deep learning models to obtain higher accuracy in real-world Android malicious software detection.
[34]	CNN	Precision – 99% Recall – 95% F1-Score – 97% Accuracy –98%	Android Malware Genome Project, McAfee Labs	Vulnerable to impersonation attacks
[35]	CNN	Accuracy - 95.46%	ContagioDump, Marvin, Drebin, and VirusShare	To consider the use of other program graphs like program dependence graphs, for model training
[36]	Cubic-LSTM	Accuracy - 99%	CICAndMal2017	The authors aim to use different datasets as well as different algorithms of deep learning for malicious software detection in future work.
[37]	Deep Neural Networks	Precision - 98.09% Recall - 99.56% Accuracy – 98.5%	McAfee Labs	Time-consuming
[38]	Convolutional Neural Networks	Accuracy – 95.4%	• Drebin • Android Malware Dataset	The approach requires further analysis to prove its efficiency.
[39]	Long Short-Term Memory	Accuracy –97.74%	• VirusShare • MassVet	Sometimes it fails to detect malicious behaviours whose loading and execution occur at runtime. Needs a frequent update with new labelled features to avoid ambiguous predictions
[40]	Deep Neural Network	Precision – 97.15% Recall – 94.18% F1-Score – 95.64%	Drebin	Vulnerable to impersonation attacks
[10]	Long Short-Term Memory	Precision-93.7% Recall-98.8% F1-Score-96.1% Accuracy-93.9%	Malgenome	Applications were run in the emulator for a short time and refrained from showing any malicious activities
[41]	Bidirectional Long Short-Term Memory	Accuracy-97.22% F1-Score=98.21%	Android Malware Dataset	Time-consuming
[42]	Deep Belief Network	Precision-98.68% Recall-98.12% F1-Score-98.40%	• Dreblin • VirusTotal • Contagio	Inherent limitations of static analysis

(continued on next page)

Table 1 (continued)

Reference	Deep learning model	Performance	Dataset	Limitations/Future work
[20]	Deep Belief Network	Precision-98.3% Recall-96.6% F1-Score-97.4% Accuracy-97.4%	- Android Malware Genome Project - Dreblin - Contagio	Dataset too small with too many features
[43]	Deep Belief Network	Precision-98.5% Recall-99.3% F1-Score-98.72% Accuracy-98.71%	- Android Malware Genome Project - VirusShare	Inherent limitations of static analysis
[44]	Gated Recurrent Unit	Precision-96.9% Recall-99.2% F1-Score-98.0% Accuracy-98.2%	CICAndMal2017	Not available
[45]	Deep Neural Network	Precision-95.35% Recall-95.31% F1-Score-95.31% Accuracy-95.31%	- Contagio Mobile malicious software minidump - DroidBench - GitHub- Android malware master - VirusShare - VirusSign	Aim to improve the malware detection accuracy by experimenting with other deep learning methods.
[46]	LSTM	Precision- 99.3% Recall-99.2% F1-Score-99.3% Accuracy-99.3%	Android Malware and Goodware	The model cannot incorporate new behavioural-driven heuristics to make it adapt to new, unseen malicious software threats.

events to make better predictions. The update gate regulates the quantity of information from preceding time steps that has to be forwarded to the future (succeeding steps) [62]. On the other hand, the reset gate decides the amount of the previous information (history) to be forgotten by the network [63]. To improve the RNN's memory capacity and ease of model training, one can use a GRU. The Gated Recurrent Unit was applied by [44] to detect malware using CICAndMal2017 and recorded a precision of 96.9%, recall of 99.2%, F1-measure of 98% and accuracy of 98.2%. However, GRU has slow convergence and low learning efficiency [64].

iii Convolutional Neural Networks (CNNs)

CNNs consist of three distinguishable layers, namely convolutional, pooling and fully connected, as depicted in Fig. 3. The first layer, convolutional, is generally used to compute different feature maps. The outputs of the convolutional layer are then passed to the pooling layer [65]. The pooling layer is connected between two layers of convolution [66]. It is used to reduce the size of the convolved features while maintaining important features. This also decreases the computational power re-

quired to process the data [67]. CNNs can use ReLu, Maxout, tanh and sigmoid activation functions to introduce the nonlinearity required to solve nonlinearly separable problems/ features [65].

The fully connected layer determines the association between features and the target class [68]. Several studies, including [21] [17, 22-25, 34], and [38], successfully applied CNNs to detect malware in Android applications. A study conducted by [26] applied Convolutional neural networks to detect malware using the dataset from VirusShare and achieved the best performance results of 99.82% accuracy, F1-score of 99.86%, recall of 99.91% and 99.91% precision. However, the model can effectively detect malware when working with huge and updated malware datasets.

iv Deep Neural Networks (DNNs)

DNNs entail numerous layers used to solve complex nonlinear problems [69]. It consists of many hidden layers connected to the input and output layers [70]. However, deep neural networks generally have more layers than artificial neural networks [71]. The input layer accepts an input vector, while the hidden layers perform some computa-

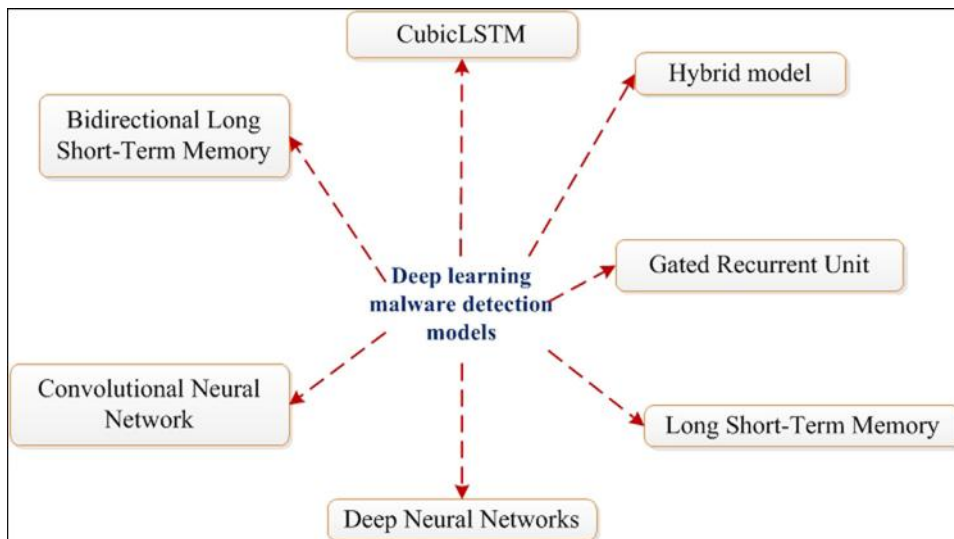


Fig. 2. Deep learning-based malware detection models.

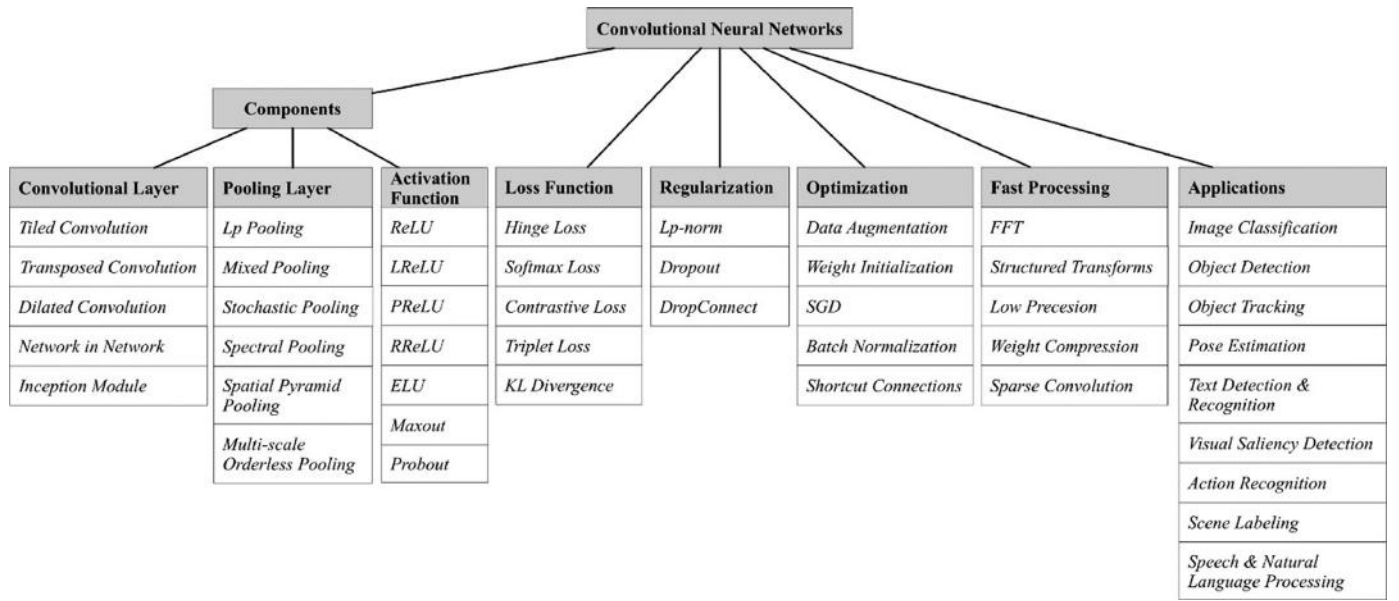


Fig. 3. Taxonomy of CNN and application areas [65].

tions and send the output to the output layer. To minimize the error or cost, the weights and biases are iteratively fine-tuned using algorithms such as stochastic gradient descent). The model performs well in error minimization between the network output and the true class because, after performing every forward pass through the network, backpropagation executes a reverse (backward) pass to automatically adjust the model's parameters. A study by [45] applied deep neural networks to detect malware using datasets from Contagio Mobile malicious software minidump, DroidBench, VirusShare and VirusSign and achieved a precision of 95.35%, recall of 95.31%, F1-score-95.31% and accuracy of 95.31%. Also, [35] detected malware in the Drebin dataset using deep neural networks and achieved a precision of 97.15%, recall of 94.18%, and F1-score of 95.64%.

v Long Short-Term Memory (LSTM)

LSTM is a subset of RRN that trains and learns long-range temporal dynamics in sequences of arbitrary length [10]. Generally, the LSTM model consists of an input gate, output gate and forget gate [72]. The model remembers the long sequences for a long period. The LSTM model has a cell state which ensures that there are no alterations to the information flowing through the units. Each unit has an input gate, an output gate, and a forget gate used by the cells to control the information to be retained or discarded before relaying the long-term and short-term information to the succeeding cell. The gates act as filters that remove irrelevant and unwanted selected information [10]. The filtered-out information is only that from variables deemed useless. The forget gate uses the sigmoid activation function to decide the information from the previous cell state to keep or forget. It is achieved by computing the product of the inbound long-term memory and a forget vector produced by the present input and inbound short memory. Lastly, there is the output gate which takes the present input, the preceding short-term memory, and a newly calculated long-term memory to generate new short-term memory and decide which information to forward to the following hidden state. Long Short-Term Memory can only retain previous information since it takes received inputs from the preceding neurons, known as backward direction and this results in a poor prediction rate as it is deprived of information about the future [73]. A study conducted by [74] applied LSTM to detect malware in Android and achieved a precision of 91.3%, recall of 96.6% as well as accuracy of 93.7% and a low false positive rate of 9.3%. Also, [37] applied LSTM to detect malware

in MassVet and VirusShare malware datasets and achieved an accuracy of 97.74%.

vi Bidirectional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM comprises two LSTMs used to accept input, nonetheless in opposite directions (forward and backward directions). The input flows in two directions (front and back) to preserve the future and past information unlike in LSTM where the input flows in one direction, either backwards or forward. The network can create a condition for each character in the input text depending on its past and future [73]. By doing this, detection accuracy can be easily achieved when using Bi-LSTM because it preserves past and future information. For instance, a study by [40] applied Bi-LSTM to detect malware using Android Malware Dataset and achieved an accuracy of 97.22% and an F1-score of 98.21%.

vii Hybrid Model (CubicLSTM and Bi-LSTM)

The hybrid model combines the Cubic-LSTM and Bi-LSTM to detect malicious software in Android applications. The CubicLSTM is a network with two states namely the temporal and spatial states that are created by two independent convolutions so that it allows diverse types of information to be processed and carried by different operations and states. The information in the temporal state and spatial state is processed separately to reduce the burden of prediction. CubicLSTM comprises three branches which are the temporal, spatial, and output branches. The branches are built along the three axes in the Cartesian coordinate system [75]. The temporal branch flows along the x-axis and the convolution aims to acquire and process motions. Since the temporal branch contains the motion information its responsibility is to create or produce the temporal state. The spatial branch flows along the z-axis and the convolution's responsibility is to capture and analyze moving objects. This is the branch that generates the spatial state as it ferries the spatial layout information regarding moving objects. The output branch produces the final forecasting frames along the y-axis in accordance with the forecasted motions given by the temporal branch and the moving object information supplied by the spatial branch. A two-dimensional network can be formed by piling several Cubic LSTM units along the spatial and output branches. The two-dimensional network if evolved along the x-axis can further construct a three-dimensional network[36]. Using the hybrid model helps in detecting higher accuracy rates and reduced false rates. A hybrid model is also time efficient as compared to a single

Table 2
Malware dataset sources.

Ref	Dataset	Collection Time	Malware Samples	Source
[77]	CICAndMal2017	2017	365	https://www.unb.ca/cic/datasets/andmal2017.html
[26]	VirusShare	2018, 2019, 2020	4038	https://virusshare.com/
[78]	Android Malware Genome Project	August2010-October 2011	1260	http://www.malgenomeproject.org/
[79]	MassVet	2015	127,429	https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-chen-kai.pdf
[13]	Intel Security	Not available	11,505	https://steppa.ca/portfolio-view/malware-threat-intel-datasets/
[80]	Android PRAGuard Dataset	2015	2260	https://pralab.diee.unica.it/en/AndroidPRAGuardDataset
[14]	Contagio	December 2011-March 2013	1150	http://contagiodump.blogspot.com/
[76]	Dreblin	August 2010-October 2012	5560	https://www.sec.cs.tu-bs.de/~danarp/drebin/
[81]	Android Malware Dataset	2010–2016	24,650	http://amd.arguslab.org/
[6]	VirusTotal	2012–2018	Not available	http://www.virustotal.com
[45]	VirusSign	2011	146	www.virusign.com

model. For instance, [34] applied a hybrid model by combining Bi-LSTM and Cubic LSTM to detect malware in the CICAndMal2017 dataset and achieved 99% accuracy. However, the hybrid model is highly computationally intensive and requires more memory space.

viii Hybrid Model (Deep Belief Network and Gated Recurrent Unit)

A hybrid model that combines DBN and GRU was used because of the various features static and dynamic features of Android apps. Deep Belief Network has some advantages having a better performance and faster learning rate of static features of Android applications. Comparing it with the traditional Recurrent Neural Network model, the Gate Recurrent Unit performs better when handling fewer parameters with prolonged time operation sequences, quicker training rate and fewer data needed to obtain a good generalization. This makes it ideal for processing the dynamic characteristics of Android apps. For training the models, the dynamic feature vectors were used for the Deep Belief Network, and the static feature vectors were used for the Gated Recurrent Unit, and the output vectors were inputted to the fully connected layer. To fine-tune the parameters of the hybrid model (comprising the deep belief network and Gated Recurrent Unit), they used the SoftMax activation function. This activation function squeezes the output of numerous neurons, squashing it in the (0,1) range, while the classification output is probabilistic [6]. The hybrid model improves the model's malware detection accuracy relative to its independent constituent models. Moreover, the DBN-GRU malware detection model's results are not distorted because of repackaging the software. This is so because the extracted features (both dynamic and static) are not affected by the repackaging of the software as well as the model's training process. A study conducted by [4] developed and applied a hybrid model that combines both Deep Belief Network and Gated Recurrent Unit to detect malware using Android PRAGuard Dataset and VirusShare and achieved a precision of 95.79%, recall of 97.62% and accuracy of 96.82%.

b) Identified malware dataset sources

The study revealed that deep learning-based malware detection models use various datasets from various online malware databases, as shown in Table 2. Deep learning thrives on huge datasets; hence, it requires a big dataset that is kept up to date with current and recent malicious malware to leverage the full potential of deep learning. Table 2 shows that the most used dataset is the DREBIN dataset [76], with 5 560 malware samples collected over nearly two years from August 2010 to October 2012.

The Android Malware Genome Project dataset [78] has 1260 malicious samples gathered between August 2010 and October 2011. The Contagio dataset consists of 1150 malware samples collected in 2011, and the VirusShare dataset which is publicly available through the website (<https://virusshare.com/>) has 4712 samples that were collected in 2018, 2019 and 2020. Android Malware Dataset [81] has 24,650 malicious software samples which were elicited between 2010 and 2016.

CICAndMal2017 [77] was collected in 2017 and has 365 malicious software samples. There is also the McAfee Labs dataset which can be accessed through the website with 11 505 malicious samples. The Android PRAGuard dataset [80] consists of 10 479 malicious samples and was collected in 2015. Marvin dataset [82] with 10 559 malicious samples collected between June 2012 and May 2014. There is also the MassVet dataset [79] which consists of 127 429 malicious samples, VirusSign collected in 2011 with 146 malware samples, the VirusTotal dataset which can be accessed through the website and lastly DroidBench with 30 samples and GitHub- Android malware master with 80 samples, both datasets can be accessed on GitHub. The deep learning models in Table 1 achieved more than 90% detection accuracy using the sources of the datasets shown in Table 2 although some of them have few malicious samples.

c) Identified research gaps emanating from deep learning-based malware detection models

The study identified some challenges and weaknesses associated with datasets used to detect malware by previous researchers. For instance, some of the datasets used are small, however, deep learning models require huge datasets to perform better. Some deep learning-based malware detection models used old datasets such as DREBLIN to detect malware, thus, making detection models susceptible to emerging malware. Also, some datasets are not accessible publicly making it difficult to validate the detection models. Some deep learning models have challenges of vulnerability to attacks during the training phase and testing phases. The models can suffer “data poisoning” attacks during training. The data poisoning attacks are perpetrated through manipulation of the model training to instill data that make the model make errors. During testing, the malware detection models are exposed to adversary attacks, impersonation attacks, and many others. An adversary attack can mislead DNNs and consequently cause misclassification [83].

4. Recommendations for detecting malware in android applications

There is a need to increase malware dataset sizes as well as improve the accessibility of malware datasets to the public in order to train, test and validate DL-based malware detection models using various datasets. As Android applications continue to increase in the market, more dynamic and hybrid DL-based malware detection models are required to detect emerging malicious software. Some researchers such as [6] suggested a hardening of deep learning models against adversarial attacks. To achieve this, they proposed retraining and distillation as a way of confronting the adversarial attacks on deep learning models. However, these solutions were initially created to fight “adversarial attacks” in computer vision, instead of malicious software, so further studies can be done to investigate the effectiveness of these techniques in detecting malware.

Conclusion

The unmatched increase of malware in Android applications requires some efficient solutions to prevent them. The study revealed that classical techniques based on static, dynamic and hybrid analyses continue to be susceptible to emerging malicious software. Therefore, as malware features continue increasing, deep learning models can detect malware with high accuracy. The study further revealed that deep learning models such as DBNs, GRU, CNNs, DBNs, LSTM, Bi-LSTM, CubicLSTM, and hybrid models can effectively detect malicious software in Android applications. The study also revealed that Convolutional Neural Networks have been prominently used and achieved generally high accuracy as compared to other malware detection models. However, there is a need for frequently update malware datasets so that deep learning-based malware detection models can train, learn and detect emerging malware and new cyberattacking trends and techniques [21]. The findings of this study show that deep learning can be an effective technique for detecting malicious software in Android applications. Future work can focus on applying deep learning models to detect malicious software before downloading Android applications to improve the security of Android smartphone devices.

Declaration of Competing Interests

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

References

- [1] GSMA, 2016 mobile industry impact report: Sustainable Development Goals, GSMA; (2016).
- [2] J. Batani, S. Musungwini, T.G. Rebanowako, An Assessment of the use of mobile phones as sources of agricultural information by tobacco Smallholder farmers in Zimbabwe, *J. Syst. Integr.* 2019 (2019) 1–22, doi:10.20470/jsi.v10i3.375.
- [3] J. Batani, M.S. Maharaj, Towards data-driven models for diverging emerging technologies for maternal, neonatal and child health services in Sub-Saharan Africa: a systematic review, *Glob. Heal. J.* (2022), doi:10.1016/j.glohe.2022.11.003.
- [4] J. Batani, M.S. Maharaj, Towards data-driven pediatrics in Zimbabwe, 2022 Int. Conf. Artif. Intell. Big Data, Comput. Data Commun. Syst. (2022) 1–7 IEEE, doi:10.1109/icABCD54961.2022.9855907.
- [5] L. Ceci, Annual number of mobile app downloads worldwide 2020 | Statista, Statista (2021).
- [6] T. Lu, Y. Du, L. Ouyang, Q. Chen, X. Wang, Android malware detection based on a hybrid deep learning model, *Secur. Commun. Netw.* 2020 (2020), doi:10.1155/2020/8863617.
- [7] A. Mahindru, A.L. Sangal, FSDroid: a feature selection technique to detect malware from android using machine learning techniques: fsdroid, *Multim. Tools Appl.* 80 (2021) 13271–13323, doi:10.1007/S11042-020-10367-W/TABLES/21.
- [8] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, H. Liu, A review of android malware detection approaches based on machine learning, *IEEE Access* 8 (2020) 124579–124607, doi:10.1109/ACCESS.2020.3006143.
- [9] V.G. Shankar, G. Soman, M.S. Gaur, V. Laxmi, M. Conti, AndroTaint: an efficient android malware detection framework using dynamic taint analysis, *ISEA Asia Secur. Priv. Conf.* (2017) ISEASP 2017 2017, doi:10.1109/ISEASP.2017.7976989.
- [10] R. Vinayakumar, K.P. Soman, P. Poornachandran, S. Sachin Kumar, Detecting android malware using long short-term memory (LSTM), *J. Intell. Fuzzy Syst.* 34 (2018) 1277–1288, doi:10.3233/JIFS-169424.
- [11] A. Alotaibi, Identifying malicious software using deep residual long-short term memory, *IEEE Access* 7 (2019) 163128–163137, doi:10.1109/ACCESS.2019.2951751.
- [12] A. Qamar, A. Karim, V. Chang, Mobile malware attacks: review, taxonomy & future directions, *Fut. Gener. Comput. Syst.* 97 (2019) 887–909, doi:10.1016/J.FUTURE.2019.03.007.
- [13] Zadeh Nojoo Kamar M.E., Esmailzadeh A., Kim Y., Taghva K. A survey on mobile malware detection methods using machine learning 2022:0215–21. doi:10.1109/CCWC54503.2022.9720753.
- [14] Q. Wu, X. Zhu, B. Liu, A survey of android malware static detection technology based on machine learning, *Mob. Inf. Syst.* 2021 (2021), doi:10.1155/2021/8896013.
- [15] I. Idika N., Mathur A.P. A survey of malware detection techniques 2007.
- [16] J. Senanayake, H. Kalutarage, Al-Kadri MO, Android mobile malware detection using machine learning: a systematic review, *Electron* 10 (2021) 1606 Page 1606 2021;10, doi:10.3390/ELECTRONICS10131606.
- [17] Kouliaridis V., Kambourakis G. A comprehensive survey on machine learning techniques for android malware detection. *Inf* 2021, Vol 12, Page 185 2021;12:185. doi:10.3390/INFO12050185.
- [18] M. Taleby, Q. Li, M. Rabbani, A. Raza, A survey on smartphones security: software vulnerabilities, malware, and attacks, *Int. J. Adv. Comput. Sci. Appl.* 8 (2017), doi:10.14569/IJACSA.2017.081005.
- [19] Gaurav A., Gupta B.B., Panigrahi P.K. A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system. <https://doi.org/10.1080/17517575.2021.2023764> 2022. doi:10.1080/17517575.2021.2023764.
- [20] X. Su, W. Shi, X. Qu, Y. Zheng, X. Liu, DroidDeep: using Deep Belief Network to characterize and detect android malware, *Soft Comput.* 24 (2020) 6017–6030 2020 248, doi:10.1007/S00500-019-04589-W.
- [21] J. Batani, An adaptive and real-time fraud detection algorithm in online transactions, *Int. J. Comput. Sci. Bus. Informatics* 17 (2017) 1–12.
- [22] S.I. Imtiaz, Rehman S ur, A.R. Javed, Z. Jalil, X. Liu, W.S. Alnumay, Deep-AMD: detection and identification of Android malware using high-efficient Deep Artificial Neural Network, *Futur. Gener. Comput. Syst.* 115 (2021) 844–856, doi:10.1016/J.FUTURE.2020.10.008.
- [23] Tirkey A., Mohapatra R.K., Kumar L. Sniffing android malware using deep learning 2022:489–505. doi:10.1007/978-981-19-0019-8_37.
- [24] B. Urooj, M.A. Shah, C. Maple, M.K. Abbasi, S. Riasat, Malware detection: a framework for reverse engineered android applications through machine learning algorithms, *IEEE Access* (2022), doi:10.1109/ACCESS.2022.3149053.
- [25] M.J. Page, J.E. McKenzie, P.M. Bossuyt, I. Boutron, T.C. Hoffmann, C.D. Mulrow, et al., The PRISMA 2020 statement: an updated guideline for reporting systematic reviews, *BMJ* 372 (2021), doi:10.1136/BMJ.N71.
- [26] W. Wang, M. Zhao, J. Wang, Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network, *J. Ambient Intell. Humaniz. Comput.* 10 (2018) 3035–3043 2018 108, doi:10.1007/S12652-018-0803-6.
- [27] N. Lu, D. Li, W. Shi, P. Vijayakumar, F. Piccialli, V. Chang, An efficient combined deep neural network based malware detection framework in 5G environment, *Comput. Netw.* 189 (2021) 107932, doi:10.1016/J.COMNET.2021.107932.
- [28] Z. Wang, G. Li, Z. Zhuo, X. Ren, Y. Lin, J. Gu, A deep learning method for android application classification using semantic features, *Secur. Commun. Netw.* 2022 (2022), doi:10.1155/2022/1289175.
- [29] J. Kim, Y. Ban, E. Ko, H. Cho, J.H. Yi, MAPAS: a practical deep learning-based android malware detection system, *Int. J. Inf. Secur.* (2022) 1–14, doi:10.1007/S10207-022-00579-6/TABLES/4.
- [30] K. Bakour, H.M. Ünver, DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques, *Neural Comput. Appl.* 33 (2021) 11499–11516, doi:10.1007/S00521-021-05816-Y/TABLES/7.
- [31] A. Pektaş, T. Acarman, Deep learning for effective Android malware detection using API call graph embeddings, *Soft Comput.* 24 (2020) 1027–1043, doi:10.1007/S00500-019-03940-5/TABLES/6.
- [32] Karbab E.B., Debbabi M., Derhab A., Mouheb D. Android malware detection using deep learning on API method sequences 2017.
- [33] Yuan Z., Lu Y., Xue Y. DroidDetector: Android malware characterization and detection using deep learning. vol. 21. 2016.
- [34] N. McLaughlin, J.M. Del Rincon, B.J. Kang, S. Yerima, P. Miller, S. Sezer, et al., Deep android malware detection, *CODASPY 2017 - Proc. 7th ACM Conf. Data Appl. Secur. Priv.* (2017) 301–308, doi:10.1145/3029806.3029823.
- [35] Z. Xu, K. Ren, S. Qin, Craciun F. CDGDroid, Android malware detection based on deep learning using CFG, *DFG* (2018) 11232.
- [36] M. Ahmad, D. Javeed, M. Shoaib, N. Younas, A. Zaman, An efficient approach of deep learning for android malware detection, *United Int. J. Res. Technol.* 02 (2021) 15–20.
- [37] M.K. Alzaylaee, S.Y. Yerima, Sezer S. DL-Droid, Deep learning based android malware detection using real devices, *Comput. Secur.* 89 (2020) 101663, doi:10.1016/J.COSE.2019.101663.
- [38] C. Hasegawa, H. Iyatomi, One-dimensional convolutional neural networks for Android malware detection, in: *Proc - 2018 IEEE 14th Int Colloq Signal Process Its Appl* CSP 2018, 2018, pp. 99–102, doi:10.1109/CSPA.2018.8368693.
- [39] Xu K., Li Y., Deng R.H., Chen K. DeepRefiner: multi-layer android malware detection system applying deep neural networks; deeprefiner: multi-layer android malware detection system applying deep neural networks 2018. doi:10.1109/EuroSP.2018.00040.
- [40] D. Li, Z. Wang, Y. Xue, Fine-grained android malware detection based on deep learning, 2018 IEEE Conf Commun Netw Secur CNS 2018, 2018, doi:10.1109/CNS.2018.8433204.
- [41] Ma Z., Ge H., Wang Z., Liu Y., Liu X. Droidetec: Android malware detection and malicious code localization through deep learning 2020.
- [42] T. Chen, Q. Mao, M. Lv, H. Cheng, Li Y. Droidvecdeep, Android malware detection based on word2vec and deep belief network, *KSII Trans. Internet Inf. Syst.* 13 (2019) 2180–2197, doi:10.3837/TIIS.2019.04.025.
- [43] X. Qin, F. Zeng, Zhang Y. MSndroid, The android malware detector based on multi-class features and deep belief network, *ACM Int Conf Proceeding Ser*, 2019, doi:10.1145/3321408.3321606.
- [44] O.N. Elayan, A.M. Mustafa, Android malware detection using deep learning, *Procedia Comput. Sci.* 184 (2021) 847–852, doi:10.1016/J.PROCS.2021.03.106.
- [45] A. Naway, Y. Li, A review on the use of deep learning in android malware detection, *Cryptotaphy Secur.* (2018), doi:10.48550/arXiv.1812.10360.
- [46] E. Amer, S. El-Sappagh, Robust deep learning early alarm prediction model based on the behavioural smell for android malware, *Comput. Secur.* 116 (2022) 102670, doi:10.1016/J.COSE.2022.102670.
- [47] T.M. Sanyanga, M.S. Chinzvende, T.D. Kavuu, J. Batani, Searching objects in a video footage, *Int. J. ICT Res. Africa Middle East* 8 (2019) 18–31, doi:10.4018/jic-trame.2019070102.

- [48] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, S. Nazir, A. Haq, Object detection through modified YOLO neural network, *Sci. Program* (2020) 2020.
- [49] E. Mbunge, J. Batani, R. Mafumbate, C. Gurajena, S. Fashoto, T. Rugube, et al., Predicting student dropout in massive open online courses using deep learning models - a systematic review, *Cybern. Perspect. Syst. CSOC* 2022. *Lect. Notes Netw. Syst., Cham: Springer* (2022) 212–231, doi:[10.1007/978-3-031-09073-8_20](https://doi.org/10.1007/978-3-031-09073-8_20).
- [50] C. Janiesch, P. Zschech, K. Heinrich, Machine learning and deep learning, *Electron Mark* 31 (2021) 685–695, doi:[10.1007/s12525-021-00475-2](https://doi.org/10.1007/s12525-021-00475-2).
- [51] J. Batani, E. Mbunge, B. Muchemwa, G. Gaobotse, C. Gurajena, S. Fashoto, et al., A review of deep learning models for detecting cyberbullying on social media networks, in: *Lecture. Notes Networks System, Cham: Springer*, 2022, pp. 528–550, doi:[10.1007/978-3-031-09073-8_46](https://doi.org/10.1007/978-3-031-09073-8_46).
- [52] E. Mbunge, R.C. Millham, M.N. Sibiyi, S. Takavarasha, Application of machine learning models to predict malaria using malaria cases and environmental risk factors, 2022 Conf Inf Commun Technol Soc ICTAS 2022 - Proc 2022, 2022, doi:[10.1109/IC-TAS53252](https://doi.org/10.1109/IC-TAS53252).
- [53] E. Mbunge, S.G. Fashoto, H. Bimha, Prediction of box-office success: a review of trends and machine learning computational models, *Int. J. Bus. Intell. Data Min.* 20 (2022) 192–207, doi:[10.1504/IJBIDM.2022.120825](https://doi.org/10.1504/IJBIDM.2022.120825).
- [54] A. Vial, D. Stirling, M. Field, M. Ros, C. Ritz, M. Carolan, et al., The role of deep learning and radiomic feature extraction in cancer-specific predictive modelling: a review, *Transl. Cancer Res.* 7 (2018) 803–816, doi:[10.21037/tcr.2018.05.02](https://doi.org/10.21037/tcr.2018.05.02).
- [55] M.A. Keyvanrad, M.M. Homayounpour, A brief survey on deep belief networks and introducing a new object oriented toolbox, (DeeBNet) (2014), doi:[10.48550/arxiv.1408.3264](https://doi.org/10.48550/arxiv.1408.3264).
- [56] G.E. Hinton, Deep belief networks, *Scholarpedia* 4 (2009) 5947, doi:[10.4249/SCHOLARPEDIA.5947](https://doi.org/10.4249/SCHOLARPEDIA.5947).
- [57] A.R. Mohamed, G. Hinton, G. Penn, Understanding how deep belief networks perform acoustic modelling, *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process - Proc.* (2012) 4273–4276, doi:[10.1109/ICASSP.2012.6288863](https://doi.org/10.1109/ICASSP.2012.6288863).
- [58] Y. Hua, J. Guo, H. Zhao, Deep Belief Networks and deep learning, in: *Proceeding of the 2015 International Conference on Intell Computer Internet Things, ICIT 2015*, 2015, pp. 1–4, doi:[10.1109/ICAOT.2015.7111524](https://doi.org/10.1109/ICAOT.2015.7111524).
- [59] K. Zhang, S. Shi, S. Liu, J. Wan, L. Ren, Research on DBN-based evaluation of distribution network reliability, *E3SWC* 242 (2021) 03004, doi:[10.1051/E3SCONF/202124203004](https://doi.org/10.1051/E3SCONF/202124203004).
- [60] H. Zhou, X. Yang, H. Pan, W. Guo, An android malware detection approach based on SIMGRU, *IEEE Access* 8 (2020) 148404–148410, doi:[10.1109/ACCESS.2020.3007571](https://doi.org/10.1109/ACCESS.2020.3007571).
- [61] I. Chingombe, T. Dzinamarira, D. Cuadros, M.P. Mapingure, E. Mbunge, S. Chaput-sira, et al., Predicting HIV status among men who have sex with men in bulawayo & harare, zimbabwe using bio-behavioural data, recurrent neural networks, and machine learning techniques, *Trop. Med. Infect. Dis.* 7 (2022) 231 Page 231 20227, doi:[10.3390/TROPICALMED7090231](https://doi.org/10.3390/TROPICALMED7090231).
- [62] Rehman S ur, Khaliq M, S.I. Imtiaz, A. Rasool, M. Shafiq, A.R. Javed, et al., DIDDOS: an approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU), *Futur. Gener. Comput. Syst.* 118 (2021) 453–466, doi:[10.1016/J.FUTURE.2021.01.022](https://doi.org/10.1016/J.FUTURE.2021.01.022).
- [63] S. Kostadinov, *Understanding GRU networks, Towar Data Sci.* (2017).
- [64] X. Wang, J. Xu, W. Shi, J. Liu, OGRU: an optimized gated recurrent unit neural network, *J. Phys. Conf. Ser.* 1325 (2019), doi:[10.1088/1742-6596/1325/1/012089](https://doi.org/10.1088/1742-6596/1325/1/012089).
- [65] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, et al., Recent advances in convolutional neural networks, *Pattern Recognit.* 77 (2018) 354–377, doi:[10.1016/J.PATCOG.2017.10.013](https://doi.org/10.1016/J.PATCOG.2017.10.013).
- [66] E. Mbunge, S. Fashoto, R. Mafumbate, S. Nxumalo, Diverging hybrid and deep learning models into predicting students' performance in smart learning environments – a review, *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST* 405 (2022) 182–202 LNICST, doi:[10.1007/978-3-030-93314-2_12/COVER](https://doi.org/10.1007/978-3-030-93314-2_12/COVER).
- [67] Z. Wang, Q. Liu, Y. Chi, Review of android malware detection based on deep learning, *IEEE Access* 8 (2020) 181102–181126, doi:[10.1109/ACCESS.2020.3028370](https://doi.org/10.1109/ACCESS.2020.3028370).
- [68] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, *J. Big Data* 8 (2021), doi:[10.1186/S40537-021-00444-8](https://doi.org/10.1186/S40537-021-00444-8).
- [69] G. Montavon, W. Samek, K.R. Müller, Methods for interpreting and understanding deep neural networks, *Digit Signal Process* 73 (2018) 1–15, doi:[10.1016/J.DSP.2017.10.011](https://doi.org/10.1016/J.DSP.2017.10.011).
- [70] W. Samek, G. Montavon, S. Lapuschkin, C.J. Anders, K.R. Müller, Explaining deep neural networks and beyond: a review of methods and applications, *Proc. IEEE* 109 (2021) 247–278, doi:[10.1109/JPROC.2021.3060483](https://doi.org/10.1109/JPROC.2021.3060483).
- [71] E. Mbunge, B. Muchemwa, Deep learning and machine learning techniques for analyzing travelers, in: *online reviews: a review. Https://ServicesIgi-GlobalCom/Resolvedoi/Resolve.aspx?Doi=104018/978-1-7998-8306-7CH002 1AD*, 2022, pp. 20–39, doi:[10.4018/978-1-7998-8306-7.CH002](https://doi.org/10.4018/978-1-7998-8306-7.CH002).
- [72] P. Shanmugam, B. Venkateswarulu, R. Dharmadurai, T. Ranganathan, M. Indiran, M. Nanjappan, Electro search optimization based long short-term memory network for mobile malware detection, *Concurr. Comput. Pract. Exp.* 34 (2022) e7044, doi:[10.1002/CPE.7044](https://doi.org/10.1002/CPE.7044).
- [73] I.U. Haq, T.A. Khan, A. Akhunzada, A dynamic robust DL-based model for android malware detection, *IEEE Access* 9 (2021) 74510–74521, doi:[10.1109/ACCESS.2021.3079370](https://doi.org/10.1109/ACCESS.2021.3079370).
- [74] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, A.K. Sangaiah, Android malware detection based on system call sequences and LSTM, *Multimed. Tools Appl.* 78 (2019) 3979–3999, doi:[10.1007/S11042-017-5104-0/FIGURES/9](https://doi.org/10.1007/S11042-017-5104-0/FIGURES/9).
- [75] H. Fan, L. Zhu, Y. Yang, Cubic LSTMs for video prediction, in: *33rd AAAI Conference on Artificial Intelligence AAAI 2019, 31st Innovation Appl Artificial Intelligence Conference IAAI 2019 9th AAAI Symp Educ Adv Artif Intell EAAI 2019*, 2019, pp. 8263–8270, doi:[10.1609/aaai.v33i01.33018263](https://doi.org/10.1609/aaai.v33i01.33018263).
- [76] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K.R. Ndss, 2014U. Drebin: effective and explainable detection of android malware in your pocket, *ProseclMsecOrg* (2014).
- [77] A.H. Lashkari, A.F.A. Kadir, L. Taheri, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark android malware datasets and classification, *Proc. - Int. Carnahan Conf. Secur. Technol.* (2018) 2018-Octob, doi:[10.1109/CCST.2018.8585560](https://doi.org/10.1109/CCST.2018.8585560).
- [78] Y. Zhou, X. Jiang, Dissecting Android malware: characterization and evolution, *Proc. - IEEE Symp. Secur. Priv.* (2012) 95–109, doi:[10.1109/SP.2012.16](https://doi.org/10.1109/SP.2012.16).
- [79] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, et al., Finding unknown malware in 10 seconds: mass vetting for new threats at the Google-Play scale, *Undefined* (2015).
- [80] D. Maiorca, D. Ariu, I. Corona, M. Aresu, G. Giacinto, Stealth attacks: an extended insight into the obfuscation effects on Android malware, *Comput. Secur.* 51 (2015) 16–31, doi:[10.1016/J.COSE.2015.02.007](https://doi.org/10.1016/J.COSE.2015.02.007).
- [81] F. Wei, Y. Li, S. Roy, X. Ou, W. Zhou, Deep ground truth analysis of current android malware, *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* 10327 (2017) 252–276 LNCS, doi:[10.1007/978-3-319-60876-1_12](https://doi.org/10.1007/978-3-319-60876-1_12).
- [82] M. Lindorfer, M. Neugschwandtner, C. MARVIN Platzer, Efficient and comprehensive mobile app classification through static and dynamic analysis, *Proc. - Int. Comput. Softw. Appl. Conf.* 2 (2015) 422–433, doi:[10.1109/COMPSAC.2015.103](https://doi.org/10.1109/COMPSAC.2015.103).
- [83] Specht F., Otto J. Hardening deep neural networks in condition monitoring systems against adversarial example attacks 2021:103–11. doi:[10.1007/978-3-662-62746-4_11](https://doi.org/10.1007/978-3-662-62746-4_11).