A Survey of Deep Learning Techniques for Cybersecurity in Mobile Networks

Eva Rodríguez, Beatriz Otero, Norma Gutiérrez and Ramon Canal

Abstract—**The widespread use of mobile devices, as well as the increasing popularity of mobile services has raised serious cybersecurity challenges. In the last years, the number of cyberattacks has grown dramatically, as well as their complexity. Traditional cybersecurity systems have failed to detect complex attacks, unknown malware, and they do not guarantee the preservation of user privacy. Consequently, cybersecurity systems have embraced Deep Learning (DL) models as they provide efficient detection of novel attacks and better accuracy. This paper presents a comprehensive survey of recent cybersecurity works that use DL in mobile and wireless networks. It covers all cybersecurity aspects: infrastructure threads and attacks, software attacks and privacy preservation. First, we provide a detailed overview of DL techniques applied, or with potential applications, to cybersecurity. Then, we review cybersecurity works based on DL. For each cybersecurity threat or attack, we discuss the challenges for using DL methods. For each contribution, we review the implementation details and the performance of the solution. In a nutshell, this paper constitutes the first survey that provides a complete review of the DL methods for cybersecurity. Given the analysis performed, we identify the most effective DL methods for the different threats and attacks.**

Index Terms—**Cyberattacks, Deep Learning, Machine Learning, Mobile Networking, Privacy, Security, Wireless Networking**

I. INTRODUCTION

THE number of individuals that continuously use mobile devices connected to the Internet in their daily lives, both for entertainment and work, is constantly increasing [1]. This upsurge of mobile devices, applications and services has raised important cybersecurity challenges due to the exponential increase of attacks and their sophistication [2][3]. On top of that, the growing diversity and complexity of mobile network architectures has increased the number of security breaches. It has cast a shadow on the adoption of smart mobile applications and services, which has been amplified by the large number of different platforms that provide data, storage, computation, and application services to end-users. All this makes security in mobile networks complex and challenging. According the NDIA Cybersecurity Report [4], more than 25% of industry professionals have experienced a cyberattack, being the companies with more than 500 employees the most affected (44%). Moreover, most of these companies are not confident in their ability to recover from a cyberattack within one day. Traditional cybersecurity systems fail to detect complex attacks, as well as unknown malware and they do not guarantee the preservation of users' privacy. In a first attempt, Machine Learning (ML) techniques were adopted to improve their functionalities [5], but they have not succeeded in identifying the different types of threats and intrusions, especially for unforeseen and unpredictable attacks. This has led cybersecurity systems to embrace DL models.

DL is a ML subfield, which enables computational models composed of multiple processing layers to learn different data representations. It is inspired on the brain's ability to learn from experience, and thus, it performs representation learning through multi-layer transformations. In DL, there are multiple levels of features, which are automatically discovered and composed together in various levels to produce the outputs. Its major benefit over ML is the automatic feature extraction that avoids the tedious labor of generating feature representations manually. Moreover, the self-learning capability of DL improves the processing speed and accuracy of applications. DL has gained great recognition in many areas such as image processing, speech recognition, game playing, and bioinformatics. Nowadays, academia and industry are applying DL to a wider range of applications due to its improvement in accuracy in complex tasks, fostered by recent developments in hardware and software. Similarly, DL techniques are starting to be used in the security domain to improve cybersecurity systems.

Cybersecurity comprises the processes and tools used to protect confidentiality, integrity and availability of resources and assets in the cyberspace [6]. At first, traditional cybersecurity systems protected users and devices through Intrusion Detection Systems (IDS), user authentication, data encryption, firewalls, and anti-virus software. IDSs [7] were designed to detect malicious network traffic, abnormal behaviors and intrusion attempts in computer systems. Two different types of IDSs exist depending on where the intrusion detection is deployed. (1) Host-based IDSs monitor each host, and if they detect malicious activity, for example, the modification of system files or configuration changes, they alert the user. (2) Network-based IDSs check anomalies in the network traffic. They are placed at a network node, for instance, in a router or a gateway. Moreover, IDSs can also be classified according to the method used to detect the intrusion. Signature-based detection systems, also denoted as misuse-based, detect

Manuscript received January 31, 2020.
Eva Rodríguez, Beatriz Otero, Norma Gutiérrez and Ramon Canal are with the Department of Computer Architecture, Universitat Politecnica de Catalunya, Barcelona, Spain (e-mail:evar,botero,norma,rcanal}@ac.upc.edu.

known attacks based on predefined patterns for malicious network activities. They provide high accuracy in detection, but they cannot detect novel (zero-day) attacks. On the other side, anomaly-based detection systems aim to identify unknown attacks. The detection is based on the definition of normal and anomalous behavior patterns. However, they lack high accuracy. Initially, both approaches made extensive use of classical ML techniques [8]. But, they lack automatic feature engineering, they have a low detection rate [9], and they are not efficient in detecting small variants of existing attacks. Consequently, along with the increasing complexity of hacking incidents, ML techniques have been incapable of detecting complex attacks, unknown malware or preserving users' privacy. Consequently, DL techniques are now the focus of cybersecurity research.

In recent years, Information Technology (IT) organizations are conducting cybersecurity reports lead by the increase of cyberattacks affecting organizations worldwide. According to the 2019 SIM IT Trends Report [2] and the Cisco Cybersecurity Report [3], cybersecurity represents the most critical IT management issue. 84% of the organizations spend more than 20% of their IT budget on cybersecurity (2x increase only in the last three years) [10]. Cybersecurity reports identify most common cyberattacks and they classify them in three different areas: Web, Cloud and Internet of Things (IoT). The 2018 Internet Security Report (ISTR) [11] states that web attacks on endpoints increased by 56%. One in ten URLs analyzed was identified as malicious. Most alerts were form-jacking incidents (i.e. malicious JavaScript code to steal credit card data and other payment information). Form-jacking attacks compromised 4,818 websites on average every month. Another large set of alerts result from Living off the Land (LotL) attacks (i.e. break into the organizations' systems via trusted programs and then malicious code injection). The most common LotL are malicious emails, which increased 48% with respect to 2017.

In the Cloud, a wide range of security challenges are observed due to two factors: misconfiguration in hardware equipment, and poorly secured Cloud databases. These caused the theft or leakage of 70 million records in 2018. The number of attacks in IoT stabilized in 2018, after a massive increase of 600% in 2017. The leading cyberattacks in IoT are worms and bots, while infection vectors emerge. Routers and connected cameras are the most infected devices receiving 75% and 15% of the attacks respectively. Moreover, Distributed Denial of Service (DDoS) attacks [12] represent the third most common IoT threat in 2018, with as many as 16 different types of DDoS attacks. Finally, while the overall number of malware infections falls during 2018, enterprise ransomware increases by 12%, and mobile ransomware increases by 33%.

The 2019 MacAfee Security Report [13] points out the rapid evolution of malware and, specially, malware applications in mobile phones. 2019 also represents the year of "everywhere malware", since IoT devices, as medical devices, IP cameras, or smart elevators, are (and still may be) inherently vulnerable and easy to hack. Backdoors, crypto-mining, fake apps, and banking trojans are the largest contributors. Finally, ransomware attacks increased by 118% in 2019.

The Check Point Software Security Report 2020 [14] reviews the major 2019 cyber incidents. The leak of more than half a billion records of Facebook in an unprotected Amazon Cloud takes the lead. The second largest incident is the attack suffered by the American Medical Collection Agency, compromising personal data and payment information of more than 20 million patients. Similar to the MacAffe Security Report, it highlights the increase and sophistication of malware and ransomware attacks that are affecting also local governments and healthcare organizations. Specifically, in 2019, the botnet infection affected 28% of the organizations analyzed. These cybersecurity reports conclude that today's hyper-connected world provides more opportunities to cybercriminals. Any IT environment must be protected against future attacks. In this context, it is where DL improves traditional cybersecurity systems as it is capable of detecting novel attacks.

Security organizations are developing commercial DL-based cybersecurity solutions to protect systems against cyber threats. Symantec [15] launched an attack analytics tool which integrates Artificial Intelligence (AI) techniques to discover targeted attacks. Similarly, Vectra developed Vectra's Cognito platform [16] which uses AI techniques to detect real time attacks in IoT devices. Sophos Corporation [17] launched Intercept X tool, which applies DL models to detect threats. IBM developed the IBM's QRadar Advisor tool [18] that uses DL methods to identify malicious attacks based on cognitive analysis.

In this paper as in [19][20], cybersecurity attacks are classified in three main categories: infrastructure, software, and privacy. The *infrastructure area* comprises all the intrusion and anomaly attacks at the network level. Researchers started to use DL methods for cyber-attack detection [21] following the experience in traffic classification problems [22]. DL techniques prevent attacks by identifying patterns that are different from normal behaviors (e.g. anomaly-based network intrusion detection [14][23]). Moreover, cyberattacks share a common feature with image recognition, since more than 99% of the new attacks are a small mutant of existing attacks. In the same way that changes in images can be identified by small changes in their pixels. Thus, signatures, and patterns are automatically learned and generalized to detect future attacks [23]. In the *software area*, DL is used in malware, ransomware and botnets detection, since they are rapidly evolving to circumvent signature-based solutions [24]. The number and variety of malware attacks is increasing continuously, which makes it difficult for traditional methods (e.g., anti-virus software) to efficiently defend systems. New malware usually consists of small modifications of an existing one. Attackers improve the mechanisms of infection, obfuscation or payloads. Therefore, malicious applications of the same family have strong similarities in terms of code and behavior. This leads cybersecurity systems adopt DL in malware detection [25][26], malware classification [27][28], botnet detection [29][30] and ransomware detection [31][32]. Finally, in the *privacy area*, DL methods are used to protect user privacy. Initial works [33][34] address user privacy preservation in all types of Neural Networks (NN). Hereafter, NNs are trained with differential privacy [35][36] to avoid disclosure of private information. More recent works [37] train NNs with encrypted data.

This paper bridges the gap between DL and security, providing a comprehensive survey of DL methodologies and techniques relevant to security in mobile networks. Summarizing, this survey aims to:

- Determine the cybersecurity areas that use DL techniques.
- Identify the challenges for successful application of DL to cybersecurity.
- Provide a complete review of research work that applies DL techniques for cybersecurity in mobile networks.
- Identify the most important and promising directions for further study.

To the best of our knowledge, research papers and books in the literature, which are reviewed in Section II, only address partially the aims of this work. The rest of the paper is as follows: Section III reviews essential DL techniques applied or with potential application to cybersecurity. Section IV reviews the datasets used by cybersecurity applications and the metrics used in the evaluation process. Section V reviews recent DL research works. Section VI summarizes the lessons learned and identifies current challenges and open future directions. Finally, section VII concludes the paper.

II. Related Work

In the last years, DL and cybersecurity have crossed their paths. We can, now, find DL methods in the three cybersecurity areas (infrastructure, software and privacy). This introduction is motivated by two reasons: (1) the large majority of new cyberattacks are small mutants of existing attacks or combinations of them; and (2) DL methods have improved their accuracy in complex tasks through recent software and hardware developments. Given the significance of DL and cybersecurity, surveys and tutorials are emerging. In general, they provide a biased overview of DL methods used for a specific type of cyberattacks, or they are restricted to specific environments or applications. Table I provides an overview of existing works, a summary of its contribution and the cybersecurity area.

In the infrastructure area, Hodo et. al. [38] conduct the first review of ML and DL-based IDS. The authors define an IDS taxonomy based on the data source (host or network) and the intrusion technique (anomaly or signature based). ML and DL techniques are reviewed, as well as their performance in detecting anomalies. Their conclusion is that learning algorithms of deep networks, especially Convolutional NNs (CNN) and Deep Belief Networks (DBN), significantly outperform shallow networks in detection. Kwon et al. [7] provide a more specific survey on anomaly-based network intrusion detection. They shed more light on unsupervised and generative learning DL methods since they provide an overview of anomaly detection methodologies, and they consider data reduction, dimensionality reduction and classification. The Restricted Boltzmann Machine (RBM), DBN, CNN and Recurrent NN (RNN) methods are reviewed for network traffic analysis, being CNNs the most promising classifiers for intrusion detection. Xin et. al. [39] provide a more complete review of ML and DL methods for network intrusion detection. This survey reveals the key obstacles in this area: (1) small quantity of benchmark datasets; (2) not uniform evaluation metrics; and (3) deployment efficiency more significant than considered, (as experiments are not performed in real networks). Also, in the intrusion detection area, Ferrag et al. [40] conduct a comparative study of DL methods, mainly RNN, Deep NN (DNN), RBM, DBN, CNN, Deep Boltzmann Machine (DBM), and deep autoencoder (DAE). The methods are analyzed using two datasets: CSE-CIC-IDS2018 [41] and Bot-IoT [42]. The study concludes that CNN and DAE methods achieve the best accuracy, 97% and 98% respectively, obtaining also best performance results for both datasets.

Other surveys address DL works in the infrastructure and software security areas. Mahdavifar and Ghorbani [43] review the works that use DL models for intrusion detection, web site defacement detection, phishing detection, and malware detection and classification. This work classifies DL models in generative, discriminative, and hybrid, according the taxonomy provided by Deng et. al. [44]. Generative DL architectures are powerful at modelling the input data taking advantage of the benefits of data synthesis and pattern analysis. Discriminative architectures do not consider the data generation process, they learn the conditional probabilities of classes given the visible data, and then, they classify the data. Finally, hybrid architectures use a generative model to improve discrimination in two aspects: optimization and regulation. Malware detection uses generative architectures (e.g. Autoencoder (AE), Stacked AE (SAE), RNN) and hybrid architectures (e.g. CNN, DBN). Likewise, malware classification also uses generative (e.g. SDAE, RNN) and hybrid (e.g. DNN) architectures. In contrast, intrusion detection mostly uses AE, Long Short-term Memory (LSTM), DBN, and DNN models. Subashini et. al. [45] extend this work by reviewing ML and DL algorithms. They mostly focus on the infrastructure area for intrusion and anomaly detection, and they marginally outline the contributions in the software area for botnet and malware detection. Unlike other surveys, [45] reviews Nature Inspired computing (NIC) paradigms which are applied for fine-tuning the parameters in the security learning model to categorize attacks. This improves efficiency and performance. The study concludes that RBM, DNN, RNN, and Suport Vector Machines (SVM) are the most used models for network anomaly and intrusion detection. While SAE, SVM, CNN and DBN are most used for malware detection and classification. In the same vein, Berman et. al. [46] and Singla et. al. [47] also cover cyberattacks in the infrastructure and software areas. The two surveys review AE, CNN, RNN and Generative Adversarial Networks (GAN) models used to detect cyberattacks. They focus on a subset of attacks that include malware, botnets, and network intrusions. Singla et al. [47] highlight the lack of accuracy in IoT environments for devices with low processing capabilities.

TABLE I
SUMMARY OF EXISTING SURVEYS RELATED TO DL FOR CYBERSECURITY

Publication	**Summary**	**Scope**		
		Infrastructure	**Software**	**Privacy**
Hodo et. al. [38] *(2017)*	Overview of shallow and deep networks IDS.	✓		
Kwon et al. [7] *(2019)*	Survey of DL methods (unsupervised learning) for anomaly based intrusion detection.	✓		
Xin et al. [39] *(2018)*	Survey of ML and DL methods for intrusion detection.	✓		
Ferrag et al. [40] *(2020)*	Comparative study of DL methods for intrusion detection.	✓		
Mahdavifar and Ghorbani [43] *(2019)*	Survey of DL methods for intrusion detection and malware detection and classification.	✓	✓	
Subashini et al. [45] *(2020)*	Review of ML and DL methods for intrusion and anomaly detection, and botnet and malware detection.	✓	✓	
Berman et al. [46] *(2019)*	Survey of DL methods for cybersecurity, analyzing works in the infrastructure and software areas.	✓	✓	
Singla et al. [47] *(2019)*	Analysis of DL methods for security tasks in malware analysis, intrusion detection and botnet detection.	✓	✓	
Zachariah et al. [48] *(2017)*	Overview of malware detection techniques.		✓	
Scalas et al. [49] *(2019)*	Review of Android ransomware detection techniques.		✓	
Al-Garadi et al. [20] *(2018)*	Survey of ML and DL methods for IoT Security.	✓	✓	✓
Hemdan and Manjaiah [50] *(2020)*	Review of research works using Big Data Analytics to detect and prevent cyberattacks.	✓	✓	
Wickramasinghe et al. [51] *(2018)*	Survey of DL methods for Cyber-Phisical security applications.	✓	✓	
Sedjelmaci et al. [52] *(2020)*	Special issue on AI-based cybersecurity solutions in CPS.	✓		✓
Zeadally et al. [53] *(2020)*	Survey of cybersecurity attacks and AI-based solutions in IoT and CPS.	✓	✓	
Zhang et al. [19] *(2019)*	Survey of DL methods in mobile and wireless networking	✓	✓	✓

Other surveys focus on specific environments or applications, mainly Android applications, Cloud, IoT-Fog computing, and Cyber-Physical Systems (CPS). Surveys that review cybersecurity attacks on Android devices are conducted by Zachariah et. al. [48] and Scalas et al. [49]. The former analyzes existing malware detection techniques in the Android OS. This work reviews both, static and DL-based approaches. The latter makes an in-depth analysis of ML and DL-based ransomware detection techniques, but focusing on those that make use of system application programming interface (API) information.

Another field, gaining special relevance, where surveys are starting to emerge is IoT. Al-Garadi et al. [20] review the usage of ML and DL methods in IoT security. The authors provide an analysis of the vulnerabilities and attack surfaces. They are categorized into physical device, network and cloud services, and web and application interfaces. This survey also provides an in-depth analysis of ML and DL methods and their application to each of the IoT layers. In the same area but from a totally different perspective, Hemdan and Manjaiah [50] review research works that use Big Data Analytics to detect and prevent cyberattacks. They analyze the usage of Big Data Analytics and DL in Social Networks, Cloud Computing and IoT to predict new attacks.

Finally, in the CPS area, Wickramasinghe et al. [51] concisely review DL algorithms used in security applications. The authors analyze regularization techniques to improve the generalization capabilities of DL-based security applications. More specifically, they analyze the DL models used in the infrastructure area, mainly for intrusion and anomaly detection, and in the software area for malware detection. In the CPS domain, several industry and academic experts share their vision on AI-based cybersecurity solutions [52]. This work focusses on the infrastructure and privacy areas. Contributions of special interest encompass DoS attacks detection and Federated Learning for data privacy preservation. In the same line, Zeadally et al. [53] review cybersecurity issues in the IoT and CPS domains. They analyze cybersecurity attacks launched on different network stacks and applications, and discuss their impact. They first present an overview of non-AI security solutions, discuss their weaknesses and describe how emerging AI solutions can help to improve cybersecurity. The AI-based solutions encompass the network infrastructure and software areas focusing on IoT and CPS.

Finally, the most complete survey is elaborated by Zhang et al. [19]. The authors review works in DL for mobile and wireless networking. This survey provides a review of the state-of-the-art on DL practices in different domains including data analysis, user mobility analysis, network control, security, and signal processing. In the security field, authors summarize existing works in the three security areas: infrastructure, software and privacy. Authors provide an interesting conclusion for network attacks. They state that DL methods do not guarantee the detection of all possible new attacks and propose the improvement of DL solutions by means of: (1) transfer learning, which is the rapid transfer of knowledge from existing attacks to newer ones, and (2) lifelong learning, continuously updating the model with the features of new attacks. Nevertheless, as the survey is not focused on security it does not provide an in-depth analysis of existing works.

The works described previously in this section do not completely cover the different DL techniques used for cybersecurity in mobile networks. Some of them address a subset of the wide range of possible cyberattacks in networks, systems or applications. While other surveys are restricted to specific domains, as IoT, CPS, or Android OS. This work goes beyond these previous surveys and reviews all the works in the literature that use DL techniques to improve cybersecurity systems in mobile and wireless networking. We analyze a wide range of DL methods that are increasingly used by cybersecurity systems and that are not completely covered by previous works. For all the works analyzed in this survey, we review the implementation details (i.e. the libraries used for deploying and training the NNs and the optimization algorithms) and we analyze and compare the results reported wherever possible.

In summary, this work differentiates from earlier surveys on the following:

- It reviews all the research works in the different security areas: infrastructure, software and privacy
- It considers all different scenarios in mobile and wireless networking, including Cloud, Fog computing, IoT, CPS, etc.
- It analyzes the implementation for almost all the works (except those that do not provide it), as well as the evaluation performed. When possible, we also compare the different proposals.

Finally, to the best of the authors' knowledge this is the first survey that provides a complete review of DL methods used for cybersecurity applications.

III. Deep Learning methods used in cybersecurity applications

This section reviews DL history and summarizes the most common DL methods used in cybersecurity applications which include Multilayer Perceptron (MLP), CNN, RNN, LSTM, AE, RBM, and DBN.

A. Deep Learning overview

In the 1950s, AI started to emerge. It was in 1956 in the Dartmouth Conference [54] when Dr. McCarthy proposed that "machines can be programmed to reason simulating every aspect of learning or any other feature of intelligence". In this area there are different fields that are tightly related: AI, ML, Artificial NNs (ANNs) and DL. Figure 1 presents a taxonomy showing the relations between them.

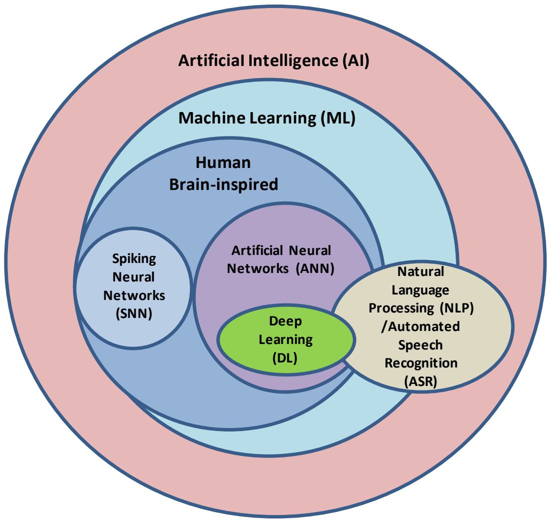

Artificial Intelligence (AI)
Machine Learning (ML)
Human Brain-inspired
Spiking Neural Networks (SNN)
Artificial Neural Networks (ANN)
Deep Learning (DL)
Natural Language Processing (NLP) /Automated Speech Recognition (ASR)

Fig. 1. AI taxonomy: ML, SNN, NLP, ASR, ANN and DL.

The aim of AI is to automatize intellectual tasks usually conducted by humans, while ML and DL are the specific methods that lead to this goal. In turn, ANNs [55] are ML algorithms inspired by biological neural networks that model complex real-world problems. ANNs can be defined as computing structures designed with simple processing elements, called artificial neurons or nodes. Neurons are fully-connected and simulate the way a human brain processes information, and solves problems. ANN consist of three or more interconnected layers. They are able to perform massively parallel computations, enabling them to improve their results as more data is inserted to the model. McCulloh and Pits [56] in 1943, were the first ones that modeled the operation of simple artificial neurons. This milestone marked the beginning of DL modern history. Artificial neurons were implemented in 1958 by Rosenblatt [57], which introduced supervised learning, through the perceptron, in the area of character recognition. The perceptron consists of a few layers of neurons connected by adaptive weights. Networks with one hidden layer fall in the shallow learning category, while networks with multiple hidden layers pertain to the DL category. In 1965, Minsky and Papert [58] identify the limitations of the perceptron. In the 1980s, Hopfield [59] presented the potential of NNs which promoted the adoption of ML techniques in many applications that people use daily (e.g.in web search, recommendation, image recognition, speech to text conversion) [21]. Initially, ML techniques exploited shallow architectures that typically contained only one layer of nonlinear feature transformations to transform the raw input into a problem-specific feature space. ML algorithms are based on either supervised, unsupervised, semi-supervised, or reinforcement learning [60]. Supervised learning algorithms make use of the training data to learn a function by mapping certain features from the training data into some output. They are applied to datasets that have features with associated labels that enable the ML model to emulate the expert's input data. Common supervised ML approaches include decision trees (DTs), SVM, Bayesian algorithms, k-nearest neighbor (KNN), and random forest (RF). Unsupervised algorithms are applied to datasets without labels. They map directly the input to the output, without depending on human intervention, learning automatically features at multiple levels of abstraction. This approach learns useful properties from the dataset structure and it detects patterns. Common unsupervised ML approaches include Principal component analysis (PCA) and K-means clustering. Semi-supervised algorithms are used for datasets with labeled and unlabeled data, where all the features are present but not all of them have associated targets. It uses unlabeled data to improve supervised learning tasks, when the labelled data is scarce or expensive. In reinforcement learning the algorithm is trained for a specific task where an overall outcome is desired. This approach is used in unknown environments. In contrast to supervised learning, the system is not trained with the sample dataset, the system learns through trial and error. ML techniques result effective for solving simple or well-constrained problems, but they present difficulties when dealing with more complex real-world applications since they have limited ability when processing natural data in their raw form.

In the cybersecurity field, researchers first used ML algorithms such as DTs, RF, SVM, Bayesian network and K-Means to detect network attacks [61][62]. However, the proposed solutions require manual feature engineering [63] and the features obtained (such as number of requests, connection time or number of bytes sent and received) are not able to fully represent the pattern behavior of network attacks. DL methods [39] that overcome ML limitations are currently used.

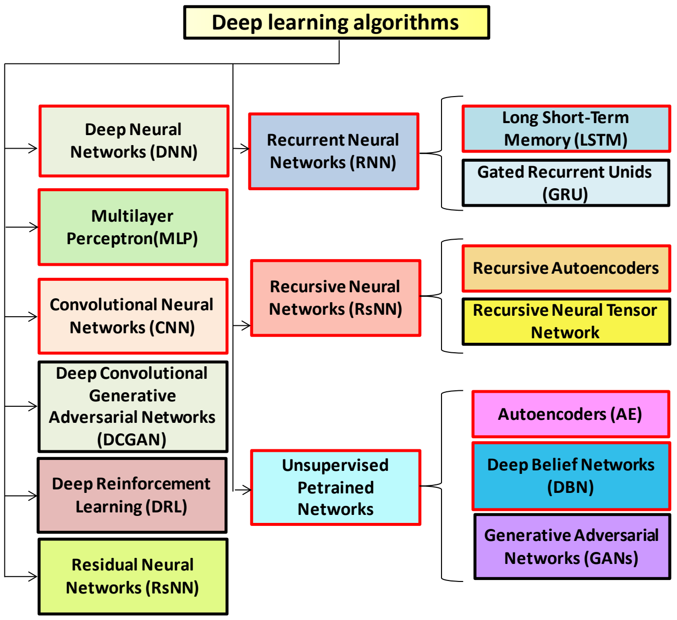
DL [64] is a sub-branch of ML that enables an algorithm to predict, classify or make decisions based on data without explicitly being programmed in a specific direction. DL algorithms are more accurate than ML algorithms because of its multilayer structure. Hierarchically, they obtain knowledge from data through multiple layers of non-linear processing units. DL tools do not rely on features defined by domain experts, in contrast to ML tools. This fosters the adoption of DL algorithms since they can extract knowledge from raw data through multiple layers of nonlinear processing units. DL is widely used by industry and academia. It extends to areas of visual recognition, audio processing, natural language understanding, pattern recognition, bioinformatics, mobile networking, and cybersecurity. DL provides encouraging results due to its high efficiency in studying complex data. Besides the improvement in learning procedures, the main factors that contribute to DNN success are: the ever-increasing computing power, the advancements in software engineering, and the massive generation of training data.

From 2016 onwards, DNN architectures and DL models are being used in a wide range of application domains such as: mobile wireless networking [19][65], network traffic control systems [22], speech processing and computer vision [66], IoT [67], recommender systems [68], and in cybersecurity. The rest of this section describes the DL methods used in cybersecurity applications.

B. DL methods

DL methods consist of different modules that transform the representation from one level of the NN to the next, which is more abstract. The first level receives the raw input data and the last one produces the outputs. The combination of several levels is needed to learn high complex models. Higher layers of representation are used for classification tasks, which consider the features of the input, as these are relevant for discrimination and suppress minor variations. The key is that feature layers self-configure directly from data (i.e. they are not designed manually by engineers). DL methods, as ML methods, can be classified in the following four main groups [64]: supervised, unsupervised, semi-supervised, and reinforcement learning.

In the cybersecurity field, supervised learning algorithms are widely used for privacy preservation and malware detection. Semi-supervised learning has been proposed for Android applications. While, unsupervised learning is the preferred for network intrusion detection, and in IoT environments. Figure 2 summarizes the DL methods used or with potential to be used in cybersecurity, and the most relevant are described in the next paragraphs.



Deep learning algorithms
Deep Neural Networks (DNN)
Multilayer Perceptron(MLP)
Convolutional Neural Networks (CNN)
Deep Convolutional Generative Adversarial Networks (DCGAN)
Deep Reinforcement Learning (DRL)
Residual Neural Networks (RsNN)
Recurrent Neural Networks (RNN)
Recursive Neural Networks (RsNN)
Unsupervised Petrained Networks
Long Short-Term Memory (LSTM)
Gated Recurrent Unids (GRU)
Recursive Autoencoders
Recursive Neural Tensor Network
Autoencoders (AE)
Deep Belief Networks (DBN)
Generative Adversarial Networks (GANs)

Fig. 2. Overview of main DL methods for cybersecurity.

1) MLP

In ANNs [69], neurons are organized in layers and connections are introduced from one layer to the next. MLP [70], also known as Fully Connected Network (FCN), is one of the first ANNs. It is considered the main architecture of DL and it is based on the simplest and oldest neural model. It consists of multiple layers of simple interconnected nodes (a.k.a neurons). More specifically, it consists of three different types of layers: one input layer, one or more hidden layers, and one output layer. The MLP is fully connected (see Figure 3), which means that all the neurons in a layer have connections to the neurons in the adjacent layers. Connection weights determine the correlation degree between the neurons' activity level (determined by the sum of the inputs of the node modified by an activation function). MLP can be used in both supervised and unsupervised DL techniques.

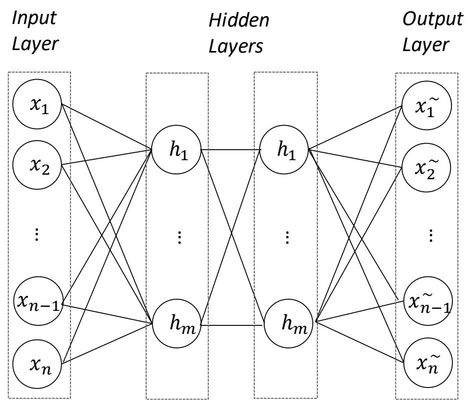


Fig. 3. MLP structure.

In the security area, MLPs have been successfully used for the detection of novel attacks in different mobile scenarios [71], especially for DDoS attacks [72][73][74].

2) CNN

CNNs [75][76], also known as ConvNets, were designed to process data in the form of multiple arrays: 1D for signals and sequences, 2D for images and 3D for video and volumetric images. CNNs take advantage of the properties of natural signals based on the following key ideas: usage of many layers, local connections, shared weights, and pooling. They also use a set of connected kernels to capture correlations between different data regions.

The architecture of a CNN (see Figure 4) consists of three different types of layers: convolutional, pooling and classification. Convolutional layers are the core of the CNN, where its units are organized in feature maps. Each unit in a feature map is connected to the local patches in the feature maps of the previous layer through a set of weights (filter bank). The result of applying these filters goes through a non-linearity transformation (usually, a rectified linear unit (ReLU)). These convolutional kernels enable close physical or temporal relationships and help reduce the memory requirements as they apply the same kernel through the entire input.

The role of the pooling layers is to merge semantically similar features into a single one by applying a specific function (e.g., the maximum over non-overlapping subsets of the feature map). Pooling layers reduce the size of the feature maps and the number of overfitting parameters. This results in a reduction of the memory requirements. Then, these layers are stacked and fed into a fully connected DNN.

The convolutional and pooling layers were inspired by the notions of simple and complex cells in visual neuroscience [77]. They have their roots in the neocognitron, which has a similar architecture, but they lack of an end-to-end supervised learning algorithm such as back-propagation. The first 1D CNN was used for the recognition of phonemes and words [78]. Subsequently, they were applied to document reading, optical recognition, and handwriting recognition systems [79]. In 2012, in the ImageNet competition, CNNs evidenced their performance in image classification by reducing the top-5 error by 39.7% [67]. Over-fitting and gradient vanishing CNN problems were reduced in GoogLeNet [80] and ResNet [81] by increasing the depth of the CNN structures, and using inception and residual learning techniques. This structure was improved by the Dense Convolutional Network (DenseNet) [82], which reuses feature maps from each layer reducing the number of layers and improving accuracy.

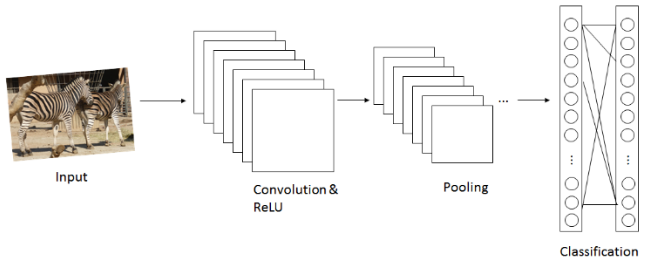


Fig. 4. CNN structure.

In cybersecurity, Dense Convolutional Networks have been successfully used for the detection of DDoS attacks [83], android-malware [84], malware traffic classification [85], encrypted traffic classification [86][87], and privacy-preserving mobile analytics [88].

3) RNN and LSTM

RNN [89] is one of the most used models for training sequential data. This type of NN is an extension of a conventional feed-forward NN with cyclic connections. RNNs are more powerful in modeling sequences. RNNs produce, at each time step (t), an output (o_t) via recurrent connections among hidden units (s_t).

A standard RNN (see Figure 5) usually is trained via a Back-Propagation Through Time (BPTT) algorithm to handle a variable-length sequence input. In a BPTT, the model is first trained, and then, for each time step, the output error gradient is recorded. However, gradient vanishing and exploding problems are frequently reported in traditional RNNs, which make them particularly hard to train [90].

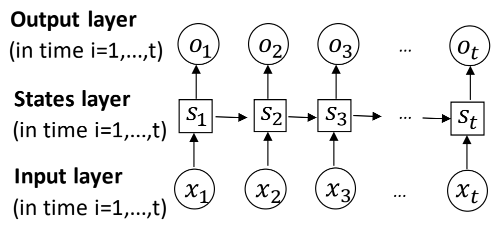


Fig. 5. RNN structure.

LSTM [91] overcomes gradient vanishing and exploding problems of RNNs introducing the LSTM cell formed by a set of gates, as depicted in Figure 6. In LSTM, three gates control the information flow. The input gate determines the ratio of the input, which affects the calculation of the cell state. The output gate determines if the previous memory cell is going to pass. The forget gate passes what the output gate stipulates.

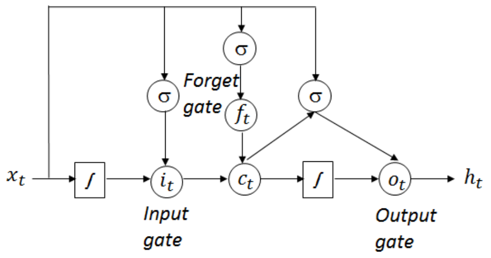


Fig. 6. LSTM Memory Cell structure.

LSTM has been successfully used in many applications in the areas of speech recognition [92], wearable activity recognition [93] and text categorization [94]. While, RNNs have achieved outstanding accuracy in tokenized predictions [95]. Nowadays, RNNs are used in intrusion detection systems [89] and in mobile networks, since they produce massive sequential data from different sources (traffic flows, the evolution of mobile network subscribers' trajectories, etc.).

4) Auto-encoders

AEs [96] and its families are one of the most applied DL models, as they produce satisfactory results in unsupervised learning. AEs learn the best parameters to reconstruct the output to the same values as, or as close to, their inputs. An AE usually has an input layer, a hidden layer, and an output layer with the same dimension of the input layer. If the hidden layer has a smaller dimension that the input layer the network is used for encoding the data, and it is known as a sparse AE [97].

AEs are designed to provide a more powerful and non-linear generalization than the PCA. To this end, back-propagation is applied and the target values are set to the same value as the inputs. AEs are also used as a non-linear transformation to discover interesting data structures, to impose different constraints on the network, and to compare the results with PCA. First, the input is transformed in a lower-dimensional space and then it is expanded to reproduce the input. For modeling non-linear dependencies in the input, once a layer is trained, its code is fed to the next layer. This code-layer is typically used for classification, as a compressed feature vector. AEs have been successfully used in the cybersecurity field for malware classification and network-based anomaly detection [23].

SAE [97] uses multiple layers of AEs to compress the information. It consists of two symmetrical DBNs, which have multiple layers for encoding and decoding. SAE achieves better accuracy, and at the same time, it reduces the computational costs and the training data required. The output of each hidden layer is used as the input of the next hidden layer. In this way, the first layer learns first-order features of the input, while the second layer learns second-order features, and so forth. Figure 7 shows the structure of a SAE. SAEs have been used in several application areas, achieving good results in object recognition and image analysis. In the security area, they are adopted for attack detection [21], and intrusion and anomaly detection [98].

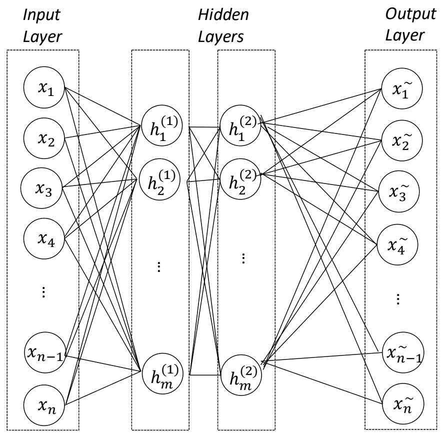


Fig. 7. SAE structure.

Deep conditional generative models are used for output representation learning and structured prediction. Output distribution is modeled as a generative model which is conditioned to the input observation. Conditional variational AEs (CVAE) [99] take advantage of developments in variational inference and directed graphical models [100]. Their input observations modulate previous Gaussian latent variables that generate the outputs. They have been successfully applied in large-scale visual recognition, and intrusion detection systems [101].

5) DBN

DBNs or Stacked RBMs can be viewed as a composition of simple RBMs or AEs where each hidden layer serves as the visible layer for the next layer.

A RBM [102] is an ANN method. It is initially designed for unsupervised learning purposes that exploits unlabeled data to learn usable patterns. RBM is an energy-based undirected generative model that makes use of a layer of hidden variables to model a distribution [103][104] over visible variables (see Figure 8). Each variable can only take a binary value (0 or 1).

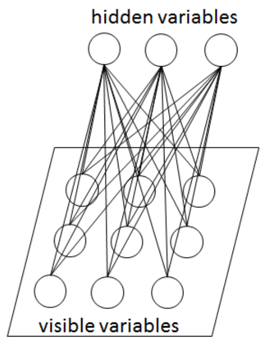


Fig. 8. Graphical model of a RBM.

RBMs are proposed as building blocks of DBNs [105]. The idea is that those hidden neurons extract relevant features from the observations. Then, these features are used as input to another RBM. Therefore, when stacking RBMs, features learned from features achieve a high-level representation.

RBMs show great performance when applied to problems involving high dimensional data as text [106] or images. Stacked RBMs or DBNs are successfully applied to time series forecasting [107], ratio matching [108], and speech recognition, and they give better results than MLPs. In the security area they are successfully used in intrusion detection [109] and malicious code detection [110].

IV. Evaluation Metrics and Datasets for Cybersecurity

The previous section has presented the most significant DL methods used in the cybersecurity field. This section describes the metrics used in the evaluation process and the datasets used in the training and testing phases.

A. *Evaluation metrics*

The DL-based cybersecurity works reviewed in this survey use the following metrics [111] in the evaluation process: accuracy (ACC), precision (p), False Alarm Rate (FAR), True Positive Rate (TPR), False Positive Rate (FPR), specificity, Receiver Operating Characteristic (ROC) curve, Area Under the Curve (AUC), and F_1 Score. These metrics can be computed from a confusion matrix, i.e. a matrix representation of the classification results (see Table II). True Positive (TP) and True Negative (TN) denote the number of attack and normal records correctly classified. Meanwhile, False Positive (FP) and False Negative (FN) denote the number of normal and attack records incorrectly classified.

From table II we can compute the metrics as detailed below.

TABLE II
Confusion Matrix

		Predicted class	
		Normal	Attack
Actual class	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

ACC is the ratio of correctly classified predictions over the total number of instances evaluated:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

p is the ratio of items correctly classified from the total of items predicted:

$$p = \frac{TP}{TP+FP} \quad (2)$$

FAR represents the ratio of items incorrectly classified as class *C* to all the items not classified as class *C*:

$$FAR = \frac{FP}{TP+FN} \quad (3)$$

TPR (a.k.a sensitivity, Detection Rate (DR), Probability of Detection(P_D) and Recall(r)) represents the ratio of items correctly classified (attack or normal) as class *C* to all the items that were class *C*:

$$TPR = \frac{TP}{TP+FN} \quad (4)$$

FPR represents the ratio of items incorrectly classified (attack or normal) as class *C* to all the items in class *C*.

$$FPR = \frac{FP}{TN+FP} \quad (5)$$

Specificity represents the ratio of items correctly classified as not class *C*, to all the items in class *C*. It is related to FPR as follows:

$$Specificity = 1 - FPR = \frac{TN}{TN+FP} \quad (6)$$

The *ROC* curve [112] results of plotting TPR over FPR. Any point in the ROC space corresponds to the performance of a classifier on a given distribution. The ROC curve provides a visual representation of the trade-off between the benefits (TPR) and costs (FPR) of classification in relation to data distributions.

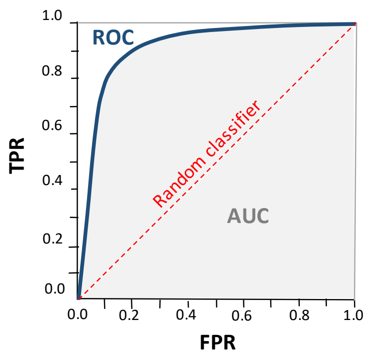


Fig. 9. ROC curve.

AUC represents the area under the ROC curve (values range between 0 and 1). It measures the degree of separability and it helps to determine if the model can effectively distinguish between classes. Figure 9 shows the relationship between these two metrics.

F_1 *score* is the harmonic mean of *p* and TPR.

$$F_1 = \frac{2TP}{2TP+FP+FN} \quad (7)$$

B. *Cybersecurity Datasets*

This sub-section provides an overview of the datasets used in the infrastructure and software areas. The datasets used in privacy applications are not presented in this section because they are not specific of security. Privacy studies use image classification datasets.

1) *Infrastructure datasets*

Infrastructure datasets consist of network traffic, which include cyberattacks detected by NIDS. NIDS datasets are largely used to develop solutions for network systems to prevent organizations from cyberattacks including, monitoring and analyzing network traffic and raising an alarm when an intrusion is detected. NIDS offers two datasets: signature-based (SNIDS) and anomaly detection-based (ANIDS). In SNIDS, attack signatures are pre-installed. Then, intrusions are detected through pattern matching. They are effective and they achieve a high detection accuracy in the detection of known attacks. ANIDS detects intrusions when they observe deviations from normal traffic patterns. They have potential on the detection of new and unknown attacks. Existing attacks, detected by these systems, can be classified in the following types:

Probing: These type of attacks [113] gather information about computer networks to discover vulnerabilities of IP, ports, and services to circumvent its security controls. For example, scanning programs (satans, nmap, mscan, etc.) can be used to discover open ports and services. Then, attackers exploit these vulnerabilities.

Denial of Service (DoS): In this class of attacks, malicious users cripple the services offered by a site, for example by flooding a site with many requests [114]. In this way, they limit or deny the system services to legitimate users. If the attack is originated by multiple coordinated hosts, it is called DDoS. There are different types of DoS attacks. Volume-based attacks, in which attackers try to consume all network bandwidth making impossible legitimate user access the site (e.g. by sending to the victim server a high number of packets). Examples include UDP or ICMP flood. Protocol attacks overfloods resources such as memory and/or processing capability in the victim. These leads to long waiting queues at intermediate network devices (load balancers, routers, firewalls, etc.). Examples include Smurf DDoS [115], where attackers attempt to flood a targeted server with ICMP packets, or SYN Flood [116], where attackers repeatedly send initial connection request SYN packets to overwhelm the available ports of the server. Finally, attacks in the application layer crash the application (e.g. webserver) by sending legitimate messages or requests. Examples include Zero-day or Slowloris attacks. Slowloris attacks [117] leave connections to a targeted Web server open as long as possible, by means of incomplete HTTP requests.

Remote-to-local (R2L): In this type of attacks, the attacker sends packets to a machine over a network, locating a vulnerability in the machine and gaining access. In R2L attacks, the attacker does not have an account on that machine. Some of these attacks (imap, named, sendmail) are caused by buffer overflows exploits in network programs. Others exploit misconfigured security policies, as ftp-write, or dictionary. While other attacks employ Trojans, for example, the xsnoop employs password capture programs.

User-to-root (U2R): In this class of attacks, the attacker begins accessing a normal user account on the system. Usually, the attacker previously has gained access through an R2L attack, for example, by sniffing passwords, or with a dictionary attack. Then, the attacker exploits the system vulnerabilities to gain root access. Examples of U2R attacks include buffer_overflow, loadmodule, perl, rootkit, ps sqlattack and xterm.

Nowadays, there are different datasets with network traffic records, which include the attacks described. They contain a large amount of data to simulate an IDS model both for training and testing. The two most popular are the KDDCUP'99 [118] and its enhanced version NSL-KDD [119]. The KDDCUP'99 dataset has been used for anomaly detection methods. It was created for the KDD Cup challenge in 1999 and consists of 4,900,000 network traffic records, each one of them with 41 features (based on basic type, content type, and traffic type features) labeled as normal or attack. Attack labels can be of one of the following four categories: Probing, DoS, U2R or R2L. Tavallaee et al. [120] statistically analyzed the KDD dataset and found important issues in the data that affect the performance of the evaluated systems. First, it contained a large number of redundant and duplicate records, which lead learning algorithms to be biased to the more frequent reports. Second, the synthetic nature of the network and data, make the dataset an inaccurate representation of real existing networks. To solve this, they proposed an enhanced version of the dataset called NSL-KDD, which consists of 125,973 training records and 22,544 testing records. Each with 41 features, as the KDDCUP'99.

Another widely used network dataset is the ISCX [121], which includes HTTP, FTP, SMTP, SSH, IMAP, and POP3 traffic. However, it does not contain HTTPS traces. It has two profiles: the Alpha-profile which conducts different multi-stage attack scenarios; and the Beta-profile, with the benign traffic (i.e., realistic network traffic with background noise). The CICIDS2017 dataset [122][123] contains attacks and normal network data, being very close to real network data. It was created by capturing traffic during 5 consecutive days. During this time, it registered many cyberattacks (DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan, and Botnet) along with normal traffic. This dataset has been labeled and more than 80 network traffic features were extracted and calculated using the CICFlowMeter software. The ADFA13 dataset [124] consists of normal training data and 10 attacks per vector. It only contains a small set of known existing cyberattacks: FTP and SSH password brute force, add new superuser, Linux Meterpreter payload, Java-based Meterpreter, and C100 Webshel. Moreover, some attacks of this dataset are poorly separated from the normal data [125]. Finally, the Winter's dataset is based on the Sperotto dataset [126], which is the first public labeled flow-based dataset. The Sperotto data was captured by monitoring a honeypot at the University of Twente for 6 days. It was divided into three categories: malicious traffic, side-effect traffic, which is not malicious by itself, and unknown and uncorrelated alerts that cannot be determined as malicious or benign traffic. This dataset has a large number of flows which make the training phase time-consuming. The Winter dataset addresses this issue generating an enhanced version of the Sperotto dataset, where duplicated data was deleted. It includes 20,000 random samples plus successful attack flows. Other datasets used for DDoS attacks are EPA-HTTP dataset [127] and CAIDA 2007 dataset [128].

2) *Software datasets*

Nowadays, DL techniques are increasingly being used for malware detection, since malware applications are evolving to circumvent their detection by existing anti-virus software. Experiments in these scenarios use the top apps in the Google Play Store [129] as normal data (benign applications); while malware data is extracted from applications in malware datasets, as Contagio [130], Genome project [131], Comodo [132], Virus Share [133], Maltrieve [134], Virus Total [135], DREBIN [136], Microsoft Malware Classification [137], netlux

[138], offensivecomputing [139], and maldozer [140]. These datasets aim to characterize existing Android malware. For example, the Genome malware project has collected more than 1,200 malware samples, which covers a large number of existing malware families. They characterized them in different aspects: installation methods, activation mechanisms, and the nature of the carried malicious payloads. The DREBIN dataset [136] contains 120,000 Android applications, but only 5000 of them are malicious. These malicious apps belong to 179 different families of malware such as: Adrd, BaseBridge, DroidDream, DroidKungFu, FakeInstaller, Geinimi, GinMaster, Kmin, Opfake and Plankton. Malware datasets are usually saved as raw program files, which provide flexibility for feature extraction and processing. Comodo Cloud Security Center includes 3,000 Android applications, being a half of them benign and the other half malicious. The malicious apps include popular malware families such as Geinimi, GinMaster, FakePlayer.

V. DL to Enhance Security in Mobile Networks

For some time now, DL and cybersecurity have crossed their paths. Nowadays, DL is being widely used to improve network security. This section provides an insight review and analysis of research works that make use of DL methods to improve cybersecurity systems. These works are organized in three main areas: *infrastructure*, *software* and *privacy*.

A. Infrastructure

Nowadays, intrusion detection (ID) is one of the main security problems. Initially, cybersecurity systems used a combination of different solutions (i.e. firewalls and IDS to protect users from cyberattacks). Anderson [141] introduced the concept of ID in 1980. Denning [142], in 1987, proposed a methodological framework for intrusion detection that became the basis of any IDS. The goal of IDSs is to identify attacks or unusual access on networks. IDSs are placed on gateways or routers to detect intrusions in the network. Intrusion attacks are usually classified, based on the KDD'99 dataset, in the following categories: DoS, Scanning, R2L and U2R. IDSs benefited from ML techniques, such as ANNs, SVMs, Naive-Bayesian (NB), RF, and Self-Organized Maps (SOM). However, these ML techniques cannot identify all the different types of intrusions [143], especially unforeseen and unpredictable attacks. Recently, IDS use DL techniques to overcome this issue. They automatically learn signatures and patterns (supervised learning) and identify patterns clearly different from normal patterns (unsupervised learning).

This section reviews, analyzes, and compares the works that use DL solutions to prevent cyberattacks. The subsections are organized as follows: (1) DL-based solutions for network intrusion detection, (2) DL-based solutions for DDoS detection, and (3) DL-based distributed solutions for cyber-attacks detection. Figure 10 provides the complete picture.

1) Network intrusion detection systems (NIDS)

In the last years, some concerns have raised in NIDSs. They have been motivated by the increasing requirement of human interaction and the decreasing levels of intrusion detection accuracy, just as ML techniques, which also have failed to detect complex attacks [143]. The emergence of the upcoming fifth-generation (5G) mobile technology, which increases transmission rates in wired networks, has demonstrated IDS ineffective to detect potential cyberattacks in their initial phases, because they fail to analyze all network packets. In the meantime, DL methods are gaining success for cyberattack detection. They are able to detect novel attacks by identifying patterns that are different from normal behavior. They have demonstrated improvements over traditional ML approaches and a strong potential for being used in modern NIDSs. This section reviews DL-based approaches that develop efficient and flexible NIDS.

Initial works [144][145][109] propose DBNs for intrusion detection. Salama et al. [144] propose a hybrid solution that combines DBN and SVM for the intrusion detection scheme. The DBN is used as feature reduction method, and SVM as classifier. The intelligent IDSs consist of three main phases: pre-processing, DBN feature reduction and classification. Pre-processing of the NSL-KDD dataset consists on mapping symbolic features to a numeric value and attack name assignment. Dimensionality reduction is achieved by DBN with back-propagation to reduce the data output size. It uses 2 RBM layers, reducing the first data from 41 to 13 features, and the second from 13 to 5. Finally, in the intrusion classification phase, the 5 output features from the DBN are forwarded to the SVM classifier. The SVM finds a decision boundary that maximizes the margin of separation between the classes. The solution, implemented using the Weka software [146], reduces the dataset size by 87%, and then classifies the reduced dataset. It provides a better classification than SVM and it also reduces the testing time (due to the reduction of the data size). This is especially important for real time applications. Subsequently, Gao et al. [145] propose a framework for network intrusion detection based on a greedy multilayer DBN, which performs efficient classification tasks. The unsupervised learning algorithm is used to pre-train and fine-tune the network, and to learn a similarity representation over the input data. The system is implemented using MATLAB 7.0, and it is evaluated, unlike [144] and [108], using the KDD CUP'99 dataset. The system performs well on intrusion recognition tasks, and the best result is obtained for a four-hidden-layer RBM with an ACC of 93.49%. Similarly, Alom et al. [109] develop an IDS based on DBN for attack detection. They test the solution using the NSL-KDD dataset, but normalized through a numerical encoding procedure. For the classification phase, the system achieves a 97.5% ACC versus the 40% of the dataset training data. The authors compare the results with Salama's [144], which achieve an ACC of 92% in 3.07 seconds. They conclude that their solution performs better, in terms of testing accuracy, and it improves the training time required by 5.5%, reducing it to 0.32 seconds.

In the same vein, other works also propose unsupervised feature learning for network-based anomaly detection, but using AEs. Yousefi-Azar et al. [22] differentiate from other AE works as: (1) they use a unique training phase and topology; and (2) the proposal is effective for two different types of cyberattacks: network intrusion detection and malware detection (see Section V.B). The proposed model uses a minimum number of features compared to similar works, so it is more computationally efficient for real time protection. It actually results optimal for small devices, so it can be used in IoT. The authors test the model using the NSL-KDD dataset [119] and they use the Sklearn library, of Python 2.7.12, to implement the classifiers. They obtain an ACC of 83.34%. This line of work continues in Shone et al. [147] that propose a solution based on stacked NDAEs for unsupervised feature learning, and RF as a classification algorithm. They implement the model in Tensor Flow [148], and they use the KDD Cup '99 and NSL-KDD datasets for evaluation. They obtain better results than previous approaches in terms of accuracy, precision and recall. They compare the model against the mainstream DBN technique proposed by Gao et. al. [145], and their model offers up to a 5% of improvement in accuracy, while the training time is reduced up to 98.81%. Niyaz et al. [149] also uses an unsupervised scheme for network intrusion detection, but they chose sparse AEs due to its high performance and ease of implementation [150]. They use the sparse AE back-propagation algorithm to find the optimal weight matrices, bias vectors, and sigmoid for the activation nodes in the hidden and output layers. To test the model, they use the NSL-KDD dataset, but preprocessed, converting nominal attributes into discrete attributes using 1-to-n encoding and eliminating the attribute num_outbound_cmds with a 0 value, achieving a total number of 121 attributes. In a second phase, they apply the new learned features representation on the training data for classification using a soft-max regression (SMR). The solution is implemented for three different types of classification: (1) 2-class (normal and anomaly); (2) 5-class (normal and four different attack categories); (3) 23-class (normal and 22 different attacks). In the implementation phase, the authors apply a 10-fold cross-validation on the training data to evaluate the accuracy of the self-taught learning (STL) classification. They also compare it to the soft-max regression when applied to the dataset without feature learning. The evaluation results show that DL has better accuracy for 2-class with respect to SMR, but for 5-class and 23-class, it does not improve significantly. Tao et al. [151] also use DAE to improve the efficiency of big network traffic classification in network security situation awareness (NSSA). Authors propose a novel approach which combines the robustness of Fisher, a traditional feature extraction method, with unsupervised learning advantages of DAE to reduce data dimensions and computation complexity. To implement the system, they used Matlab 8.0.0 and Weka 3.7.13; and the KDDCUP'99 dataset. The results show an improvement in the generalization ability of classification algorithms due to data dimensionality reduction. Lopez-Martin et al. [101] propose a network intrusion detection method based on CVAEs for an IoT network. This research work is relevant because the proposed method also performs feature reconstruction, recovering missing features from incomplete datasets. The authors call the proposed method Intrusion Detection CVAE (ID-CVAE). They use a deviation-based approach, but with a discriminative framework for traffic samples and classification. Traffic samples are labeled with the intrusion that achieves less reconstruction error, instead of using a threshold for intrusion definition. Then, the intrusion labels are included inside the CVAE decoded layers. The method is tested using the NSL-KDD dataset, and it can recover missing categorical features with 3, 11 and 70 values, with an ACC of 99%, 92% and 71% respectively.

Li et al. [110] address intrusion detection from a different point of view. Authors propose an accurate identification of malicious code to improve the efficiency of an IDS, based on DBN and AEs. A key point of the method is using AE for data dimensionality reduction (i.e., to extract the main features of the data). The AE consists of three steps: pre-training, unrolling, and fine-tuning. Then, the method uses DBN as the classifier to detect malicious code. The system is implemented using Matlab v7.11 and it is validated using the KDDCUP'99 dataset. Results demonstrate that the increase of the number of pre-training and fine-tuning iterations increases detection accuracy over a single DBN. This is because of the usage of AEs for data dimensionality reduction.

Other DL models are also used for unsupervised learning in anomaly detection. Kwon et al. [7] propose an FCN based anomaly detection system. The authors implement the system using Python Tensorflow in the Google cloud platform. For evaluation, they use the NSL-KDD dataset pre-processed normalizing numerical values and encoding categorical values as numerical values. They train the network with different hyper-parameter configuration (units, hidden layers, epochs and learning rate), and the softmax layer produces the outputs. Authors obtain promising results with an F_1 score over 90%. Kim et. al. [89] consider LSTM for implementing an IDS classifier. They chose LTSM to avoid vanishing and exploding gradient problems [90] of conventional RNNs. To test their solution, they use 10% of KDD Cup'99 for training and testing. The implementation considers softmax for the output layer and stochastic gradient descent (SGD) as the optimizer. The authors carry out two experiments. The first one, aims to find hyper-parameter values to achieve the best performance of the IDS, while the second measures the performance of the system. Hyper-parameters are parameters for model initiation.They have to be carefully chosen as they have an impact on overall performance (especially for the learning rate and hidden layer size) [152]. The results show that the detection rate and FAR have a growing trend as the learning rate is increased, having the best efficiency for a learning rate of 0.01. For smaller values of the learning rate, the system is trained too accurately, so the model detects intrusion instances, but it also considers normal instances as intrusions. The average detection rate is 98.8% and the average FAR 10%. The model detects DoS and normal instances, but U2R instances are never detected, probably because the model is trained with only 30 U2R instances.

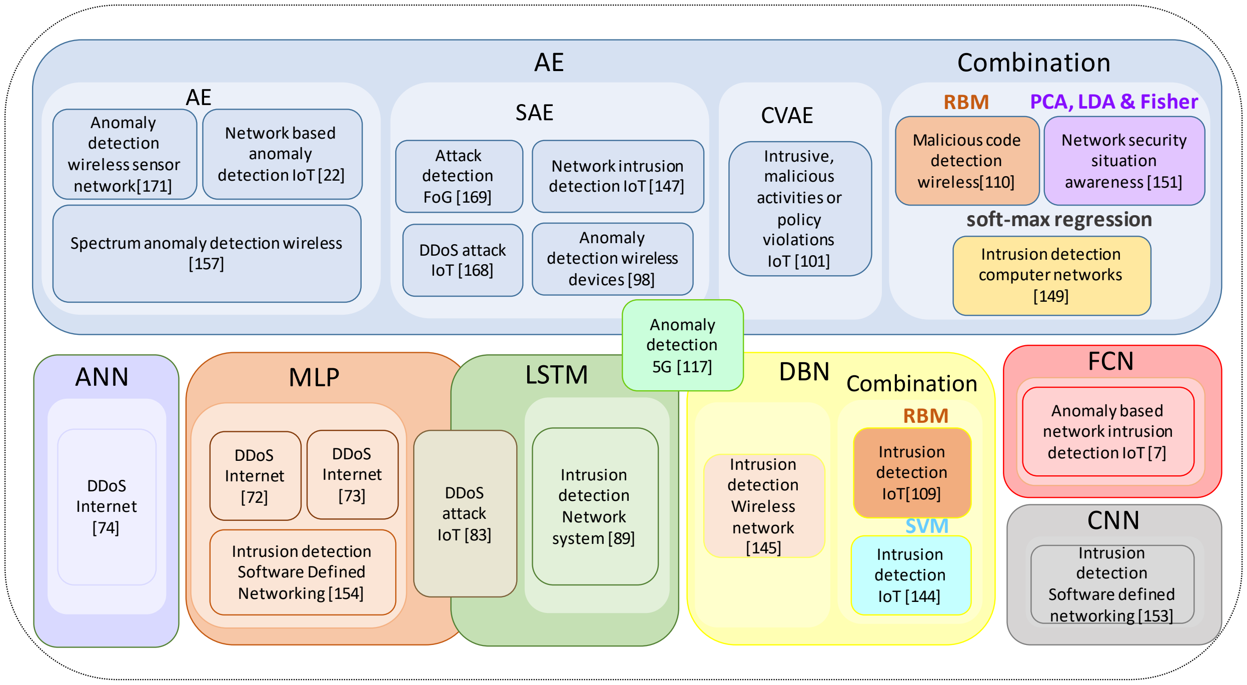

AE
Combination
AE
Anomaly detection wireless sensor network[171]
Network based anomaly detection IoT [22]
Spectrum anomaly detection wireless [157]
SAE
Attack detection FoG [169]
Network intrusion detection IoT [147]
DDoS attack IoT [168]
Anomaly detection wireless devices [98]
CVAE
Intrusive, malicious activities or policy violations IoT [101]
RBM
PCA, LDA & Fisher
Malicious code detection wireless[110]
Network security situation awareness [151]
soft-max regression
Intrusion detection computer networks [149]
Anomaly detection 5G [117]
ANN
DDoS Internet [74]
MLP
DDoS Internet [72]
DDoS Internet [73]
Intrusion detection Software Defined Networking [154]
LSTM
DDoS attack IoT [83]
Intrusion detection Network system [89]
DBN
Combination
Intrusion detection Wireless network [145]
RBM
Intrusion detection IoT[109]
SVM
Intrusion detection IoT [144]
FCN
Anomaly based network intrusion detection IoT [7]
CNN
Intrusion detection Software defined networking [153]

Fig. 10. Overview of DL methods used in the infrastructure area.

Tang et al. [153] propose a flow-based anomaly detection system using DNN, in a Software-Defined Networking (SDN) context. The DNN consists of an input layer (with dimension six), three hidden layers (with twelve, six and three neurons respectively) and an output layer (with dimension two). The system is trained and evaluated using the NSL-KDD dataset. More specifically, they use a subset of six features (duration, protocol_type, src_bytes, dst_bytes, count and srv_count). As in [89] they found an optimal hyper-parameter for the DNN. In this work, the optimal hyper-parameter is the learning rate, which is set in the range 0.1 to 0.0001. In the experiments, accuracy increases (and loss decreases) as long as the learning rate decreases in the training phase. Consequently, the best results use a learning rate of 0.0001 with a loss of 7.4% and an ACC of 91.7%. However, the testing phase results are much worse, with a loss of 20.3% and an ACC of 74.6%. Other works, as Jadidi et al. [154] consider MLP and Gravitational Search Algorithm (GSA) for flow-based anomaly detection for unknown attacks. They use an MLP with one hidden layer for anomaly detection and Winter's dataset [155] for training and testing the system. Interconnection weights of the MLP are optimized using GSA. The authors implement the GSA-based flow anomaly detection system (GFADS) using MATLAB version R2012a (7.14.0.739). They achieve a 99.43% of ACC in traffic classification. Authors compare their system to gradient descent algorithms and PSO algorithms. They conclude that GFADS is effective in the detection of attacks related to the packet header.

Other works also address cyberattacks in the IEEE 802.11 wireless network. Thing et. al. [98] propose the use of SAE for feature engineering and softmax regression in the classification task. They chose softmax because it supports multi-class classification. They test the system using a custom dataset [156], which is collected from a lab set up to emulate a typical SOHO infrastructure, with various smart devices. Different attacks are carried out in the lab to collect both attack and legitimate WI-FI signal's measurements. The attacks are divided into three different types: flooding, injection, and impersonation. The authors first normalized the data in the dataset to standardize the feature range, and facilitate the DL process. They propose to implement the system with two frameworks composed of two and three hidden layers respectively. Hidden layers consist of 256, 128 and 64 neurons. In the experiments, the system achieves an overall accuracy of 98.66%. Thus, the authors conclude that their solution correctly performs 4-class attack classification, taking in consideration novel attacks. Finally, Feng et al. [157] also propose a solution for anomaly detection in wireless networks but using a deep-structure, AE NN, for spectrum anomalies detection and frequency diagram, which acts as the feature of the learning model. Their approach relies on the reconstruct error to determine if the signal is an anomaly or not. They implement a two-layered AE NN. The network is trained with data collected by an RTL-SDR device from a real-time electromagnetic environment, and Additive White Gaussian Noise (AWGN) is selected as the anomaly. The model outperforms about 2% a conventional one-layer AE network.

DL-based solutions also are proposed in the upcoming 5G mobile technology, since IDSs fail in the detection of cyberattacks. Deep packet inspection tools cannot work properly on wired networks over 1 Gbps, for example, Snort [158] discards packets from 1.5 Gbps onwards [159]. Fernandez Maimó et al. [117] propose a two-level DL-based architecture to identify cyber threats in 5G mobile networks.

The proposed DL model consists of two levels: the first one detects anomaly traffic conditions, so-called symptoms, by means of a supervised (DBN) or semi-supervised (SAE) learning methods. The second module uses all the symptoms generated as an input to an LSTM trained in a supervised way to recognize temporal patterns of cyberattacks. The novelty of this framework is that it supports traffic fluctuation. The DL-based architecture has been implemented using TensorFlow [148], Theano [160] and PyTorch [161]. For evaluation, they use the CTU dataset [162] which consists of unknown traffic and real botnet attacks. The proposed architecture achieved an F1 score of 0.89.

2) DDoS attacks

DDoS attacks have increased dramatically in the last years [163]. They are one of the major cyberattacks in IoT networks [164]. The main objective of DDoS attacks is to make resources unavailable to intended users. DDoS compromise multiple systems across Internet with infected agents or zombies and then form networks of botnets. One of the most important DDoS attacks hits Telegram in June 2019. Different DL techniques have been used for detecting and mitigating DDoS attacks in different network environments. Initial DL-based systems adopt supervised learning. Saied et al. [74] propose a solution based on a supervised ANN (feed-forward with error backpropagation and sigmoid activation function [165]) to detect known and unknown DDoS attacks. The algorithm is trained with a customized dataset created from real-life cases and DDoS attacking patterns produced by DDoS tools. They launch known and unknown DDoS attacks, each with 20 to 120 zombies, totaling of 1160 individual attacks. Data is structured to accommodate attack patterns in a qualified format accepted by the Java NN Simulator (JNNS) [166]. 80% of the dataset is used for training, and the remaining 20% to validate the learning process. Before training the input, the values are normalized to maximize their performance in sensitive applications. They use three topological ANN structures (ICMO, TCP, and UDP) with three layers each (input, hidden and output). The experiments consider QuickProp, Back-Propagation, Backprop Weight Decay, Backprop through Time, and Sigmoid, Elliott, Softmax, BAM as an activation function. Sigmoid activation function and back-propagation learning achieve the highest detection accuracy (98%). The model increases accuracy as up-to-date patterns are fed into the system. The system learns from scenarios and detects zero-day patterns which are similar to known DDoS attacks. Other works that use supervised learning but use the MLP model are [72] and [73]. Siaterlis and Maglaris [72] explore the DDoS attack detection capability of MLP. They propose a solution that uses different metrics to detect UDP flooding attacks at the edge and train the classifiers through examples. The inputs of the MLP are several types of passive measurements taken from their university network. They use the TFN2K tool to launch UDP floods with customizable bandwidth and packet rate. By doing so, they can train their network and evaluate it in terms of "false positive" and "true positives". Network edges are protected from incoming attacks and the rest of the network from outgoing attacks. In the experiments, the system achieves a TPR above 74% and an FPR lower than 3%. Singh and De [73] continue this work but combine MLP with a Genetic Algorithm (MLP-GA). The solution uses incoming traffic, to detect application-layer DDoS attacks. They perform a behavior analysis of attackers and normal users. Then, the classification model inputs include: HTTP count, concentration of the IP addresses, constant mapping, and frame length. The model is tested with the EPA-HTTP dataset [127], CAIDA 2007 dataset [128] and an experimental dataset produced using the Slowloris attack [167]. The method achieves an ACC of 98.04% in detecting DDoS attacks with an FPR of 2.21%. The authors compare the proposed model with traditional classifiers demonstrating that they obtain better results.

Roopak et al. [83] propose a DL-based IDS to detect DDoS attacks. They implement four different classification DL models: MLP, 1d-CNN, LSTM, and CNN+LSTM. They use the CICIDS [122] dataset. They balance the DDoS attack dataset by duplicating the data, which improves the training of DL methods. The authors compare DL models to SVM, Bayes, and RF ML algorithms. The DL is implemented using Keras on Tensorflow, while the ML uses MATLAB 2017a. The CNN+LSTM model achieves the highest ACC (97.16%) while MLP achieves the lowest (86.34%). The accuracy of the ML models is in between CNN+LSTM and MLP. Therefore, the authors conclude that the best solution is the hybrid CNN+LSTM.

Other works use semi-supervised learning. Yadav et al. [168] propose a solution for detecting DDoS attacks in the application layer through traffic classification using SAEs. They construct their dataset from features extracted from their web server log (from request flooding, session flooding, and asymmetric attack). The logs are pre-processed and the features are transformed to a numeric form. This dataset is then split into two: one for training and another one for testing. The solution first learns features through the SAE, then it defines them as features of the DDoS dataset that are fed into the system. At the end, they are classified with a logistic regression classifier. The solution is implemented in Java, using WEKA (Waikato Environment for Knowledge Analysis) and Matlab. The experimental results demonstrate that the proposed method learns features from the SAE, which are beneficial for classifications. It improves the DR to 98.99% with an average FPR of 1.27%.

3) Distributed attack detection solutions

A novel solution for cyberattack detection in emerging Fog computing ecosystems is based on distributed DL. Fog computing brings Cloud Computing closer to the physical world of smart things and it requires new cybersecurity models to be resilient, adaptive, and closer to the edge. Edge nodes already provide computing, storage, communication, and control services. In the same way, they will host security services. Attacks in fog-to-things systems range from probing for gaining access to the local system to DDoS attacks. However, R2L and U2R are the most common attacks since most of the IoT devices are remotely accessed for management and updates. IoT devices are targeted through backdoors, which allow unauthorized remote entities to bypass legitimately the authentication. Similarly, devices are targeted through rootkits, which exploit programming flaws or system design, and therefore take advantage of privilege escalation. In this context, Diro et al. [169] propose the first distributed attack detection scheme in the social IoT based on multi-layer deep networks. The architecture consists of coordinating master nodes and fog nodes. The fog nodes are responsible for hosting attack detection systems and for training models, while the master nodes are responsible for collaborative parameter sharing and optimization. This speeds up data training as it is performed near the source. Plus, it can share knowledge (i.e. updated parameters) from the neighbors. The authors implement the solution using Keras on the Theano package [160] for DL, and Apache Spark [170] for distributed and parallel processing. The dataset used for testing the solution is NSL-KDD. The system is trained without labels using SAEs to extract hidden features, which are then applied to the test data. The model employs 150, 120 and 50 neurons for the first, second and third layer respectively. The system achieves an overall ACC of 99.2% when it is trained with 25 nodes working in parallel and 95.22% when it is trained with 5 nodes. DR was 99.27% which improves the shallow model by 1.77 %. ROC curves indicate that true positive values of the model are over 99%. FAR for the DL model is 0.85%, while for the shallow model is 6.57%. The authors conclude that DL models are better than ML models, and they emphasize the scalability and effectiveness of distributed parallel learning in fog nodes.

Luo et al. [171] also propose a distributed solution for anomaly detection, based on AEs for wireless sensor networks (WSN). The solution overcomes the high computation resource consumption of DL in WSN. They build an AE NN of three layers, which includes one hidden layer of neurons. The authors design a two-part algorithm which resides on sensors detecting anomalies in a fully-distributed manner. While the training model, which represents a high computation learning task, is handled by the cloud. They create their dataset by collecting data over 4 consecutive months in a real WSN indoor testbed, which consists of 8 sensor nodes that monitor temperature and relative humidity with a frequency of 2 minutes. They complete the dataset generating synthetic anomalies using Spike and Burst models [172]. They build the AE NN with 720 nodes and use k-fold cross-validation to determine the number of hidden neurons (504). They evaluate the results through the AUC and ROC curves. They perform two types of experiments. First, varying the anomaly magnitude (Δ) of spikes and bursts following a normal distribution. After these experiments, they obtain that the AUC is usually bigger than 0.8. Second, they vary the anomaly frequency. In this case, the more anomalies occurred, the more difficult is for the system to detect them. Finally, they consider an adaptive detector since the environment is constantly evolving. They configure the AE with two different setups: random and prioritized. They evaluate the TPR and FPR. TPR is better (18%) for a random scheme because the majority of the training data is historical, while the prioritized scheme has a much lower FPR, up to 60%, because it updates the weights and biases more responsively. Thus, they conclude that the anomaly detection mechanism achieves higher detection accuracy and lower FAR.

4) *Summary table – Infrastructure*

Table III summarizes the DL-based cybersecurity solutions analyzed in the infrastructure area. For all the works, we detail the attack addressed and in which scenario; the proposed DL model and learning paradigm; the dataset used in the implementation; and its performance. We report the accuracy, except for three works that use the F_1 score, TPR and FPR to validate the proposed solution.

TABLE III
SUMMARY OF DL WORKS ON INFRASTRUCTURE

Reference	Attack	Scenario	Learning paradigm	DL Model	Dataset	Performance
Salama et al. [144]	Intrusion detection	IoT	Unsupervised	DBN and SVM	NSL-KDD	ACC=92%
Gao et al. [145]	Intrusion detection	Wireless network	Unsupervised	DBN	KDDCup'99	ACC=93.5%
Alom et al. [109]	Intrusion detection	IoT Wireless network	Unsupervised	RBM-based DBN	NSL-KDD	ACC=97.5%
Yousefi-Azar et al. [22]	Network-based anomaly detection	IoT	Semi-supervised	AE	KDD Cup '99 NSL-KDD	ACC=83.3%
Shone et al. [147]	Network Intrusion detection	IoT	Unsupervised	Stacked NDAE	KDD Cup '99 NSL-KDD	ACC=98%
Niyaz et al. [149]	Intrusion detection	Computer Networks	Semi-supervised	SAE and SMR	NSL-KDD	ACC=96%
Tao et al. [151]	Network security situation awareness	Network traffic data fusion	Unsupervised	PCA, LDA, and Fisher combined with DAE	KDD Cup'99	ACC=91%
Lopez-Martin et al. [101]	Intrusive, malicious activities or policy violations	IoT networks	Semi-supervised	CVAE	NSL-KDD	ACC=99.9%

Reference	Attack	Scenario	Learning paradigm	DL Model	Dataset	Performance
Li et al. [110]	Intrusion detection (malicious code detection)	Wireless network	Semi-supervised	RBM AE	KDD Cup'99	ACC=92.1%
Kwon et al. [7]	Anomaly-based network intrusion detection	IoT	Unsupervised	FCN	NSL-KDD	ACC=90%
Kim et al. [89]	Intrusion detection	Network systems	Unsupervised	LSTM to RNN	KDDCup'99	ACC=98.8%
Tang et al. [153]	Intrusion detection	Software Defined Networking	Unsupervised	DNN	NSL-KDD	ACC=91.7%
Jadidi et al. [154]	Intrusion detection	Software Defined Networking	Semi-supervised	MLP	Winter's data Sets	ACC=99.4%
Thing et al. [98]	Anomaly detection and attack classification (Flooding, injection, impersonalisation)	Wireless devices connectivity in smart homes	Semi-supervised	Stacked AE	AWID-CLS-R-Trn AWID-CLS-R-Tst	ACC=98,6%
Feng et al. [157]	Spectrum anomaly detection	Wireless communication network	Semi-supervised	Deep-structure AE	Custom	ACC=88.5%
Fernandez Maimó et al. [117]	Anomaly detection	5G	1)Semi-supervised 2) Supervised	1) DBN or SAE 2) LSTM	CTU	F_1 score=0.89
Saied et al. [74]	DDoS attack detection (TCP, UDP and ICMP attacks)	Internet	Supervised	ANN	Custom	ACC=98%
Siaterlis and Maglaris [72]	DDoS attack detection (UDP)	Internet	Supervised	MLP	Custom	TPR>74% FPR<3%
Singh and De [73]	DDoS attack detection (application layer)	Internet	Supervised	MLP-GA	EPA-HTTP CAIDA Custom	ACC=98%
Roopak et al. [83]	DDoS attack detection	IoT	Semi-supervised	1d-CNN MLP LSTM CNN+LSTM	CICIDS2017	ACC=97.1%
Yadav et al. [168]	DDoS attack detection	IoT	Semi-supervised	Stacked AE	Custom	ACC=99.5%
Diro et al. [169]	Distributed attack detection	Social IoT	Supervised	Stacked AE	NSL-KDD	ACC=99.2%
Luo et al. [171]	Anomaly detection	Wireless sensor networks	Unsupervised	AE	Custom	TPR>80% FPR<38%

B. Software

The continuous emergence of new malware poses a significant threat to computing systems where traditional antimalware tools are ineffective. DL provides cybersecurity experts an opportunity to develop generalizable models to detect and classify existent and new malware. Malware analysis techniques can be classified as static or dynamic. Static analysis methods extract and analyze features from applications (i.e. from the binary source code or other associated files). Examples of static features are: used permissions, systems commands, and API calls. However, the weak point of these methods is that some of them are not resistant to obfuscation and they cannot deal with self-mutating malware. Static analysis is only useful in memory-limited devices. Dynamic analysis methods overcome these issues, using dynamic features, as API calls or opcodes, which make them more reliable. Consequently, DL-based cybersecurity solutions in this area fall in two main groups: those who use API calls and those who consider opcodes. Figure 11 provides the complete list of DL-based malware solutions.

1) API call solutions

System calls are one of the most important traceable events to determine malware behavior, since malware needs to use services from the operating system (OS) to execute malicious code. Any significant action requires an interaction with the OS through its APIs (e.g. opening a network connection, writing to the registry or running a thread). Therefore, tracking the system call sequence is one of the most used methods to characterize malware behavior. By inspecting these traces, different malware families can be identified.

Different approaches based on supervised or semi-supervised

DL models are used to detect malware attacks. Initial works explore RNN models to improve malware detection. Pascanu et al. [25] propose a semi-supervised solution that learns the language of malware through the instructions executed and extracting time-domain features to detect malicious files. More specifically, the high-level events that they consider are canonicalized representations of the API calls to the OS or the C run-time library. They consider RNNs and Echo state networks (ESN) to extract the features trained in an unsupervised manner. They use MLP and logistic regression for classification. They implement the solution using Theano [160]. Their dataset consists of internal Microsoft data. This data includes event streams from 250,000 malware files and 250,000 benign files. However, the dataset is not publicly available. The dataset is randomly split into 297,500 training, 52,500 validation, and 150,000 test samples. They obtain a TPR of 71.7% and an FPR of 0.1%, which outperforms standard trigram of event models by 98.3%.

Recent semi-supervised based solutions consider SAEs. Hardy et al. [173] extract API calls from the Portable Executable (PE) files. The SAEs model has two phases: unsupervised pre-training and supervised backpropagation. Ye et al. [174] follow up this work.They perform unsupervised feature learning by means of a greedy layerwise training operation. They also add supervised parameter fine tuning. They test the proposed architecture with a dataset from Comodo Cloud Security Center [175]. The dataset consists of 50,000 file samples (22,500 malware, 22,500 benign and 5,000 unknown). The proposal achieves an ACC of 95.64%.

Finally, Athiwaratkun et al. [26] improve Pascanu's solution using LSTM and Gated Recurrent Unit (GRU) instead of RNN and ESN. They also use a single-stage malware classifier based on a character-level CNN because it improves classification performance. The dataset is composed of 75,000 Windows PE format files analyzed by the Microsoft anti-malware engine. This dataset is split in 50,000 files for training, 10,000 files for validation and 15,000 files for testing. The solution is implemented using Keras with the Theano backend DL engine. The LSTM language model with temporal max pooling and logistic regression classifier shows the best results. It improves TPR by 31.3% compared to Pascanus' solution. Agrawal et al. [176] extend Athiwaratkun's [26] work by considering relevant parameters, which are input to the system API calls. These parameters provide important malicious intent information. In particular, the model includes the parameter data along with event sequences. To build up the dataset, they collect the system API calls and inputs from 75,000 files (benign and malware). They randomly split them into 50,000 for training, 10,000 for validation, and 15,000 for testing. The solution also is implemented using Keras with the Tensorflow backend. The evaluation improves the FPR over Athiwaratkun' work [26].

Another research line uses supervised models for malware identification based on API call sequences. Kolosnjaji et al. [177] combine one NN convolutional and several recurrent layers. The convolutional layer is used for feature extraction. It combines convolutional n-grams and full sequential modelling. They implement the solution using Tensorflow [148] and Theano [160]. Malware samples are collected using malware zoo [178] from three primary sources: Virus Share [133], Maltrieve [134] and proprietary collections. They obtain the labels for training the network from VirusTotal [135]. The authors report an average precision of 85.6% and an average recall of 89.4%. Tobiyama et al. [179] also propose a solution to detect malware combining RNNs (for feature extraction) and CNNs (for feature classification). This solution first records API call sequences to construct the feature extractor. They use LSTM as the learning language model. It extracts process behavior features from the RNN; and then, the CNN classifies feature images as malware or benign. The CNN consists of two convolutional layers and two pooling layers. They train and validate the proposed solution using 81 malware and 69 benign process log files. Although the dataset is really small, they obtain an AUC of 0.96.

API call based solutions are also used for malware classification (i.e. assigning a given sample its malware class). Classifying malware is important because it provides information about the attack and its motivation. Nowadays, it is especially important because of the sharp increase of malware families. Thus, automated malware classification is now the best large-scale defense for detecting malware. The first proposals on malware classifiers use sparse binary features [180][181]. The number of features is in the order of tens or hundreds of millions. Feature selection techniques reduce the number of features to enable the training of algorithms such as logistic regression. However, the number of features is still too large for complex algorithms. Three different learning paradigms are used for malware classification: supervised, unsupervised and semi-supervised. Dahl et al. [27] in 2013 propose the first work for malware classification by means of dynamic analysis using supervised learning. They develop a large-scale malware classification system that uses random projections to reduce, by a factor of 45, the dimensionality of the input space. Afterwards, they train a NN on the high dimensional input data. This enables the use of more complex supervised classification algorithms. This work considers sparse binary features based on file strings, API tri-grams, and API calls together with input values. They evaluate their proposal with 2.6 million labelled samples of 134 different malware families. They consider different DNN architectures and they also consider RBMs for the hidden layers. They obtain the best results for a one-hidden-layer DNN without RBMs (FPR=0.35%). Subsequent works use AEs for malware classification. Wang et al. [28] apply, for first time, multi-task learning to malware learning. They develop an unsupervised malware classification model based on API call sequences, which uses an RNN-AE. The RNN-AE learns in an unsupervised manner low dimensional representations of malware API call sequences. Then, it trains two decoders: one for malware classification and another for file access pattern (FAP) generation. The model is based on the multitask seq2seq [182] model and it is evaluated using the public malware API call sequence dataset [183]. The authors use a first dataset with 7430 samples for coarse-grained evaluation, and a second one with 4932 samples for fine-grained evaluation. These two data sets are split randomly for training (75%), validation (5%) and testing (20%). The model achieves a 99.2% of ACC. Yousefi-Azar et al. [22] provide a solution based on AEs for malware classification. The semi-supervised solution uses AEs to learn sufficient notion of semantic similarity between input features. The AE, trained in an unsupervised manner, has as input a feature vector generated from relevant information of the API calls. The output of the unsupervised AE is a code vector with semantic similarity between feature vectors. Finally, the resultant similarity is embedded in an abstract latent representation. The solution is evaluated using the Microsoft Malware Classification Challenge (BIG 2015) dataset available at Kaggle [137]. The SVM classifier shows the highest ACC (96.3%).

2) Opcode based solutions

Several malware detection and classification DL-based solutions use operation codes (opcodes), besides API calls, to describe program behavior. Ding et al. [184] propose a solution based on opcodes for detecting malware that uses DBNs. The unsupervised solution consists of three main modules. The PE parser that generates the opcode sequences for each executable using the n-gram model (i.e. each PE file is transformed to a n-gram vector). Then, the feature extractor finds useful n-grams providing different measures (e.g document frequency, information gain) to evaluate their classification ability. Finally, malware detection modules use a DBN to perform classification tasks. The dataset consists of PE format files. Window system files are the benign ones. Netlux [138] and Offensive Computing [139] files are the malicious ones (i.e. viruses, Trojans, worms and backdoor attacks). The proposal outperforms other learning techniques such as SVM, KNN and decision tree. It achieves an ACC of 96.7%. The authors also use the DBN as an AE. Thus, they remove the classification layer and use the top hidden layer as the output layer. This improves slightly the classification performance.

H. HaddadPajouh et al. [185] propose a semi-supervised solution that uses RNNs for detecting malware in ARM-based IoT application opcodes. The solution has three stages. In the first stage, it extracts the opcodes from the dataset. In the second stage, it obtains the feature vector from the opcodes. The third stage performs training, evaluation, and tuning for optimal results. The dataset comprises 281 malware samples collected from 32-bit ARM-based malware in the Virus Total Threat Intelligence platform [186]. The dataset has 270 benign samples collected of Raspberry Pi II applications collected from the Linux Debian repositories [187]. The solution is implemented using Google Tensor Flow [148] and Scikit-learn [188]. The model is evaluated with different LSTM configurations and the the 2-layer configuration achieves the highest ACC (98.18%).

3) Android Malware detection and classification

Mobile applications have become the most common way to access personalized computing services (e.g. email, banking, shopping, automated home control). They have become attractive targets for hackers, which take advantage of the update mechanisms to infect mobile apps. Android devices are the target in 99% of all mobile device malware [189]. The attackers produce malicious applications, usually modifying existing applications. Malware can be organized in families, where each application of the same family has a similar malicious behavior. Malicious applications gather user private data (e.g. passwords, banking credentials, contacts list). GDATA reported over 2 million of new malware Android applications in the first half of 2018 and 1.2 million in the second quarter [190]. Consequently, a significant part of malware detection and classification works in the literature focus on Android OS. The first works in this area use signature-based methods [191][192] to characterize malware using specific patterns in the bytecode and API calls. But, they are easily bypassed by byte-code transformation attacks [193]. The Fraunhofer Institute for Applied and Integrated Security study on Android anti-malware [194] concludes that most antimalware software is easily bypassed. An example is the repackaging, where an attacker decompiles a trusted application and obtains the source code. Then, the attacker adds the malicious payload and recompiles the application. Finally, the attacker makes the malicious application available on a different market. Signature malware detection techniques [195] are in many cases ineffective and the process of obtaining malware is challenging and time consuming. The malware identification period is called zero-day window and it is the moment in which malware causes the worst damage. Furthermore, Android applications can only access its own disk space. Thus, any antimalware software cannot monitor the full file system. Consequently, it is easy for applications to download and run updates without strict controls. ML methods can extract malware features through both static [196] and dynamic analysis [197]. This enables ML solutions to discriminate between benign apps and malware. Nowadays, DL methods are used for the definition of generalizable models to detect and classify malware efficiently.

There are a large number of solutions for detecting malware in Android devices. Initial works, as DroidAPIMiner [198], APKAuditor [199], and SherlockDroid [200] focus on the classification of Android malware using single-level features. However, single-level features do not reflect the overall characteristics of Android malware. To overcome these limitations, Hubner et al. [201] propose Drebin to conduct Android malware classification based on several types of features. But, the huge number of features of the classification process increases significantly the response time and resource usage. To overcome these limitations, Su et al. [202] develop DroidDeep that considers static information (e.g. permissions, API calls, component deployment) to characterize the behavioral pattern of Android apps and extract multi-level features (almost 30,000). The solution is based on the DBN model, which is fed with the extracted features for classification. They choose DBN because it is a greedy and fast algorithm which learns a reduced set of features. Finally, a detector based on the SVM algorithm feeds from the learned features. They perform experiments with 3,986 benign apps (from Google Play Store) and 3,986 malware (from Drebin [136], Android Malware Genome Project and the Contagio Community [130]). They obtain 99.4% of malware detection, outperforming the other proposals. DroidDeep also obtains a better runtime efficiency, so it can be adopted by real-world Android devices.

Yuan et al. [203] also use semi-supervised learning. In their case, the use an RBM model. They use static and dynamic analysis to extract relevant features (required permission, sensitive API and dynamic behavior) from each app. They achieve an ACC of 96%. Their dataset consists of 500 samples: 250 malware samples from contagio [130] and 250 top apps in the Google Play Store. They increase ACC up to 19% when compared to traditional ML tools (C4.5, SVM, Naïve Bayes, Logic Regression, Multi-layer perceptron). In their follow up work on malware characterization [112], they extract 192 features from both, static and dynamic analysis, but using a DBN-based DL model similar to Su et al. [202]. They design and implement the DroidDetector. They evaluate it with 20,000 benign applications crawled from the Google Play Store and 1760 malware apps (500 collected from Contagio Community and 1260 from the Genome Project). The results further improve detection ACC by 2%. Zhu et al. [204] also choose DBN as the DL model to design DeepFlow. Their solution detects malware in Android applications directly from the data flows in these applications. They test the solution using 3,000 benign apps from the Google Play Store and 8,000 malicious apps from the Android Malware Genome Project and VirusShare. DeepFlow first extracts all the sensitive data flows using FlowDroid static analysis tool [205] and then, it categorizes the extracted flows using SUSI technique [206] to obtain the features. The extracted flows are the input to the DBN model for classification. This model is trained on two crawler modules, one for malware and the other one for benign-ware. DeepFlow has a high accuracy in detecting novel malware with an F_1 score of 95.05%.

Hou et al. [207] also use semi-supervised learning, but based on SAEs. They propose a solution to improve the weakness of signature-based methods, which employ repackaging and obfuscation techniques to bypass them. This work proposes a novel dynamic analysis method, so-called Component Traversal, which can automatically execute code routines for each Android app. Based on the Linux system kernel calls, they construct weighted directed graphs. Finally, they apply SAEs on the graph-based features for detecting newly unknown Android malware. To evaluate the performance of their solution they use a real sample collection from Comodo Cloud Security Center. The SAE model is tested for different number of hidden layers and different number of neurons in each layer. The design with 3 hidden layers and 200 neurons per layer achieves the best ACC results (93.68%). They compare their solution with typical shallow learning methods (SVN, ANN, NB, DT). Detection performance improves, at least, by 5.44%.

Supervised methods are an alternative for Android malware detection and classification. Martinelli et al. [84] use it for malware classification. They characterize malware applications behavior as trusted or malware. They use sequences of system calls to capture the app behavior and they are generic enough to be robust to camouflage techniques. They use CNN for NLP classification tasks. The CNN network learns for each input (syscall sequence) a set of confidence scores encoded as vectors with a length of two (to match the predefined classes: trusted and malicious). Then, the network assigns a positive or negative label according to the highest confidence score to the input using softmax. They implement and test the model in Tensor Flow. For evaluation, they build a dataset of traces collected from 7,100 real-world Android applications, 3,536 legitimate, from the GooglePlay, and 3,564 malware apps of several different malware families from the Drebin repository. 20% of the dataset is used for training the model and 80% for testing purposes. 105 different syscalls are analyzed. The ACC of the model ranges between 0.85 and 0.95 for the training test, and between 0.75 and 0.8 for the test. Karbab et al. [140] continue this work developing MalDozer, an Android malware detector based on CNN. Maldozer performs, additionally, malware family classification. The authors develop the NN using Tensorflow, and they test it using three different datasets for the detection task: Malgenom with 37,627 benign and 1,258 malware apps, Drebin with 37,627 benign and 5,555 malware apps and Maldozer with 37,627 benign and 20,089 malware apps. The results demonstrate the correct functionality of MalDozer for detection and attribution to a malware family with an F1-Score between 96% and 99% and FPR between 0.06% and 2%. McLaughlin et al. [208] also use a CNN to design an android malware detection system. But in this work, the network automatically learns the indicative features of malware from the raw opcode sequence from disassembled programs. The training pipeline of the system is simpler than previous n-gram based solutions, since the network is trained end-to-end to learn appropriate features and at the same time it performs the classification. The proposal is evaluated with three different datasets. The first one, from Android Malware Genome project with 863 benign and 1,260 malware apps. The second one, from McAfee Labs, with 3,627 benign and 2,475 malware apps. And the third one, also from McAfee Labs, with 9,268 benign and 9,902 malware apps. All the datasets are divided into 90% for training and validation, and 10% for testing. The architecture has one single convolutional layer for all the experiments performed. The solution is developed using Torch [209]. In the training phase, the network parameters are optimized by RMSProp [210]. Before training, the network is efficiently executed on a GPU to scan a large number of files. ACC ranges between 0.87 and 0.98 for the three different datasets. This solution is also computationally efficient since it classifies 3,000 files per second approximately. Lee et al. [211] propose SeqDroid to detect malicious Android applications. They focus on obfuscated malware using stacked RNNs and CNNs. They improve learning performance by combining feature vectors of Android’s metadata (e.g. package, developer names and capability information). To validate the solution, two million APK samples are collected from VirusTotal [135]. 20% of them are selected for validation. They compare the RNN-CNN model with ngram-based models. It improves classification performance by 16%. When compared to RNN models, it reduces training time by 50% maintaining the same classification performance. Finally, Kim et al. [212] also consider a supervised approach. They develop a framework for Android malware detection based on a multimodal DL method that uses various kinds of features. The proposed framework is configurable. It can add new type of features and supports dynamic features. It supports seven types of features: string, method opcode, method API, shared library function opcode, permission, component, and environmental. The authors evaluate the performance of the framework with 41,260 samples (20,000 malware samples from VirusShare, 1,260 from the Malgenome project and 20,000 benign samples from Google Play). The multimodal NN is implemented using the Keras library [160], clustering uses Scikit-learn [188] and ML algorithms use Tensorflow [148]. They test the solution for different combinations of features (between one and seven). The authors observe that accuracy increases each time a new feature is added to the model. ACC moves from 89% when only one feature is considered to 98% when all features are factored in.

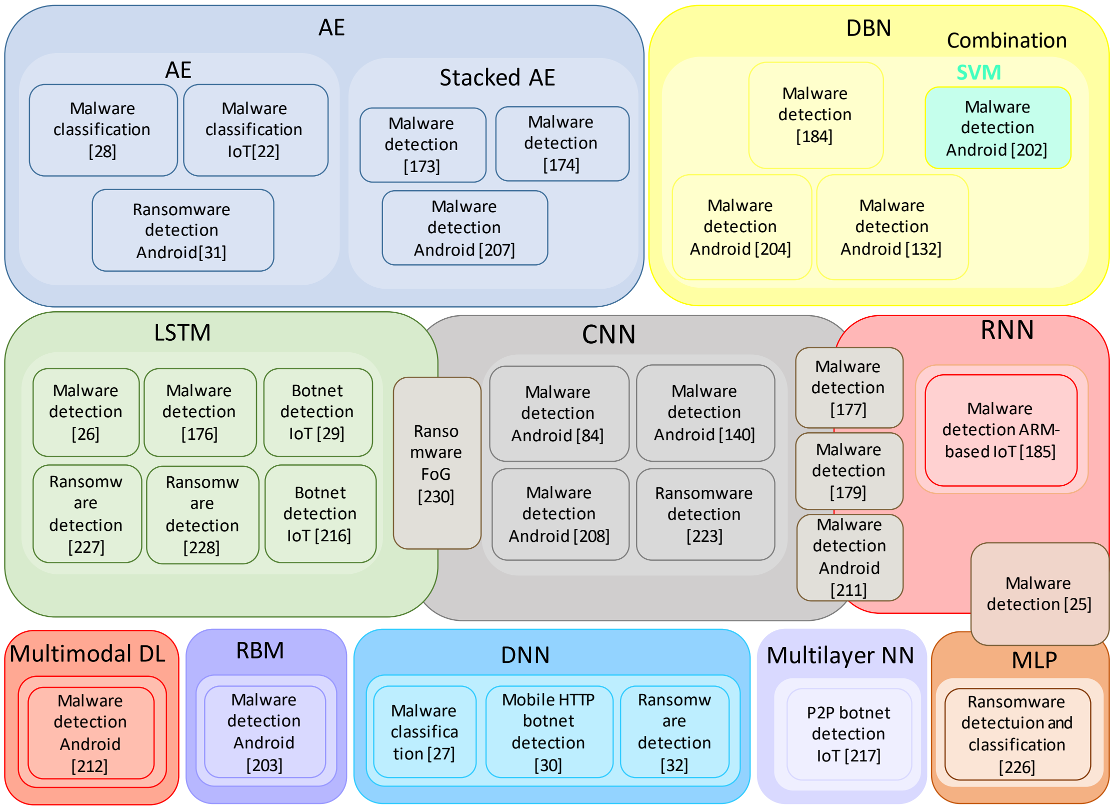

AE
AE
Malware classification [28]
Malware classification IoT[22]
Ransomware detection Android[31]
Stacked AE
Malware detection [173]
Malware detection [174]
Malware detection Android [207]
DBN
Combination
SVM
Malware detection [184]
Malware detection Android [202]
Malware detection Android [204]
Malware detection Android [132]
LSTM
Malware detection [26]
Malware detection [176]
Botnet detection IoT [29]
Ransomware detection [227]
Ransomware detection [228]
Botnet detection IoT [216]
Ransomware FoG [230]
CNN
Malware detection Android [84]
Malware detection Android [140]
Malware detection Android [208]
Ransomware detection [223]
Malware detection [177]
Malware detection [179]
Malware detection Android [211]
RNN
Malware detection ARM-based IoT [185]
Malware detection [25]
Multimodal DL
Malware detection Android [212]
RBM
Malware detection Android [203]
DNN
Malware classification [27]
Mobile HTTP botnet detection [30]
Ransomware detection [32]
Multilayer NN
P2P botnet detection IoT [217]
MLP
Ransomware detectuion and classification [226]

Fig. 11. Overview of DL methods used in the software area.

4) *Botnets*

Botnets represent one of the most dangerous kind of malware because, unlike common malware, they are not managed by predictable algorithms. Botnets are designed to infect different devices and to remain, as long as possible, active and undetectable [213]. They have a more complex pattern, because they are controlled by humans via command and control (C&C) servers or peer-to-peer (P2P) networks. Plus, their design differs from one another. Initial works in this area [214] identify two common features for botnets: (1) the commands used by the botmaster to communicate to bots, and (2) the way that bots send stolen data to the botmaster. The first works use rule-based behavioral analysis to detect the bots. Nevertheless, these models are ineffective because malicious behaviors often take place over long time scales. More recent works start to analyze network traffic for botnet detection. They generalize common patterns followed by botnets during their life cycle to detect unseen botnet traffic. Oulehla et al. [214] propose the use of NNs to obtain features from botnet behaviors. Torres et al. [29] propose the usage of LSTM to detect botnets. They analyze two different strategies, undersampling and oversampling for imbalanced traffic, because the FAR value increases considerably if no sampling technique is used. As both techniques improve detection, undersampling is preferred because it is more computationally efficient. The LSTM network is trained using stratified 10-fold cross validation. The solution is evaluated with two datasets resultant from network traffic captures in the CVUT University [215]. Using only TCP data, it achieves a TPR of 96.8% and an FPR of 1.11%. In the same vein, McDermott et al. [216] uses a Bidirectional LSTM RNN (BLSTM-RNN) model in conjunction with word embedding for botnet detection in IoT networks. This work performs detection at the packet level and it uses word embedding for text recognition and conversion, which the authors have proven useful for predicting attack vectors. They create a customized dataset of Mirai botnet traffic using the traffic of IoT cameras in a laboratory. The dataset includes benign, scan, infect and control traffic. The model is compared to LSTM-RNN. They achieve better accuracy for three of the four attack vectors considered (UDP flood, DNS flood and SYN flood attacks). ACC ranges between 98% and 99%. However, for ACK attacks the results are not that high.

Alauthaman et al. [217] also consider supervised learning. In this work, the authors propose a method for P2P botnet detection based on the traffic reduction technique. The method uses an adaptive multilayer feedforward NN in conjunction with decision trees. First, the classification and regression tree select relevant features. Then, the NN training model has the relevant features selected by a resilient back-propagation learning algorithm. To test the proposed model, two datasets are used: (a) ISOT [218], which contains malicious traffic from the Honeynet French chapter and benign traffic from the Traffic Lab at Ericsson Research in Hungary and from the Lawrence Berkeley National Laboratory (LBNL); and (b), ISCX [121]. The experiments show a detection rate of 99.8% with an FPR of 0.75%.

Recently, botnets employ the HTTP protocol, since botmasters can easily hide their activities amongst the benign web traffic. Examples of mobile Botnets are Zitmo, DroidDream, and AnserverBot [219]. Eslahi et al. [30] propose a detection approach for HTTP Botnets on Bring Your Own Device (BYOD) networks. The model consists of three main stages: Data processing (collection, reduction and filtering); feature processing; and pattern recognition. Periodic behaviors of HTTP Botnets are classified using the three metrics. The pattern recognition engine employs a Feedforward Backpropagation NN. The model is evaluated using 1000 instances from two datasets: the bots from Genome [131] and the Drebin dataset [136]. It achieves an ACC of 97.81%.

5) Ransomware

In the last years, ransomware has been growing largely causing millions of dollars of losses to industry and consumers. This type of malware installs covertly on the victim's device to demand a ransom payment, usually through crypto-currencies such as Bitcoin, to restore the infected resources. It is designed to infect, encrypt and prevent access to the system or files and lock-down hosts. There are two main types of ransomware: locker ransomware that denies user access to the system; and crypto ransomware that encrypts the files and folders of the user, but the user can access the system. New ransomware versions are appearing constantly due to the large revenues obtained by the cyber criminals [220]. These new versions easily bypass intrusion detection tools, because in most cases they are created by polymorphic and metamorphic algorithms. Initially, signature-based methods were used to detect ransomware, similar to matching binary patterns in anti-virus software. However, these methods fail when ransomware changes its behavior or uses packers to camouflage. Since the network behavior of different ransomware families is similar, ML techniques have been used for ransomware identification [221][222]. Yet, they rely on human intervention. Subsequent works use supervised DL methods to overcome human interaction dependencies, avoiding the error prone human element. Moreover, DL methods can discover the threat when the infection process starts. Hill and Bellekens [223] use Dynamic CNN (DCNN) to classify cryptographic primitives in binary executables. The solution, so-called CryptoKnight, classifies unknown software, from the cryptographic execution patterns learned. Unlike a standard CNN, it allows inputs of different lengths as the DCNN to use k-max pooling, where k scales with the input length. The solution is tested using a dataset created by the authors. They elaborate a methodology that achieves procedural generation including elements that provide some obfuscation without altering the intended control flow. The solution is able to classify the sample algorithms with 91% of ACC without extensive hyper-parameter optimization.

Tseng et. al. [32] also uses supervised learning but their work is based on deep packet inspection over network traffic. The architecture is based on a DNN consisting of 7 layers with the ReLU activation function to speed up the training process. The solution is implemented using Tensorflow [148] to build the NN and the dpkt library [224] to decode the payload in the original pcap files. To test the solution, they build their own dataset capturing 23 families of ransomware pcap files, as CryptXXX, CryptoWal, or TeslaCrypt from malware traffic analysis websites [225]. The dataset includes files corresponding to new ransomware not used for training. This approach is used to validate that the solution can predict new ransomware. The network achieves an ACC of 93%. In the same line, Vinayakumar et al. [226] propose a solution based on MLP to detect and classify ransomware with the help of API invocations. Dynamic analysis usually considers API calls made by the executable to identify the behavior of the application. The solution detects if a .EXE file is ransomware or benign and it classifies the ransomware to its corresponding category. The architecture is implemented using Tensorflow [148] and it is trained using backpropagation with a non-linear activation function. The solution is tested with a dataset generated by the authors, which includes 7 different ransomware families and 131 API calls. The solution achieves an ACC of 100%, which improves shallow network performance (96% to 98%). Similarly, Maniath et. al. [227] also propose a solution based on API calls to determine the behavior of applications. Their solution includes LSTM models to detect ransomware from executables. The LSTM network is implemented using Tensorflow. The solution is tested using a dataset elaborated by the authors that consists of 157 ransomware and benign samples collected from Microsoft Windows and online repositories. The solution achieves an ACC of 96.67%. Agrawal et al. [228] improved this work enhancing LSTM cells with the Attended Recent Inputs (ARI) mechanism. They observe that ransomware executables have high repetition of small local patterns due to their repetitive encryption. Therefore, they use methods that utilize repeating behaviors but, at the same time, maintain outer sequence event learning. ARI cells learn from recent history while processing the input sequence. The ARI-LSTM network is implemented using Keras [160] with a Tensorflow [148] backend for training using backpropagation with the Adam optimizer [229]. They construct a dataset of 26,300 samples of ransomware and benign executables for Windows OS. The framework achieves an ACC of 91%, which improves by 4% the performance of the LSTM model when detecting ransomware. LSTM networks are also used for detecting ransomware in fog computing. Homayoun et. al. [230] design the Deep Ransomware Threat Hunting and Intelligence System (DRTHIS) which uses LSTM in conjunction with CNN for ransomware detection and classification. The solution is implemented using Keras [160]. They train and evaluate the performance of DRTHIS with a dataset consisting of 220 Locky [231], 220 Cerber [232] and 220 TeslaCrypt ransomware samples plus 219 goodware samples. It achieves an F-measure of 99.6% with a TPR of 97.2%. The solution is also capable of detecting unknown ransomware. It classifies 99% of Cryptowall, 75% of TorrentLocker and 92% of Sage samples.

Other works provide solutions for the Android mobile platform. Gharib and Ghorbani [31] propose a real-time hybrid ransomware detection framework based on DAEs to reduce and learn new features. They also use, Binary and Multiple Sequence Alignment (MSA) techniques to profile malware families by analyzing dynamic system call sequences. The DNA-Droid framework evaluates an input sample using static analysis and if it is suspicious, its run-time behavior is monitored. In this way, ransomware is detected at an early stage before the infection process starts. The DNA-Droid is implemented using Scikit-learn [188] and Tensorflow [148] libraries. It is tested using a dataset developed by the authors, which contains a large collection of Android ransomware samples of eight different families (1,928 samples) and a set of 2,500 benign samples. The solution achieves an ACC of 98.1% in the best case.

6) Summary table –Software

Table IV below summarizes the DL-based cybersecurity solutions analyzed in the software area.

TABLE IV
SUMMARY OF DL WORKS ON SOFTWARE

Reference	Attack	Scenario	Learning paradigm	DL Model	Dataset	Performance
Pascanu et al. [25]	Malware detection (API calls based)	Publicly available malware applications	Semi-supervised	RNN and MLP	Microsoft dataset	TPR=71.7% FPR=0.1%
Hardy et al. [173]	Malware detection (API calls based)	Publicly available malware applications	Semi-supervised	SAEs	Comodo Cloud Security Center	ACC=96%
Ye et al. [174]	Malware detection (API calls based)	Publicly available malware applications	Semi-supervised	SAEs	Comodo Cloud Security Center	ACC=95.6%
Athiwaratkun et al. [26]	Malware detection (API calls based)	Publicly available malware applications	Semi-supervised	LSTM	Windows PE format files	FPR=1%
Agrawal et al. [176]	Malware detection (API calls based)	Publicly available malware applications	Semi-supervised	LSTM	Custom	TPR<80%
Kolosnjaji et al. [177]	Malware detection (API calls based)	Publicly available malware applications	Supervised	RNN and CNN	Virus Share Maltrieve	ACC= 89.4%
Tobiyama et al. [179]	Malware detection (API calls based)	Publicly available malware applications	Supervised	RNN and CNN	Custom	AUC=0.96
Dahl et al. [27]	Malware classification (API calls based)	Publicly available malware applications	Supervised	DNN	Custom	FPR=0.35%
Wang et al. [28]	Malware classification (API calls based)	Publicly available malware applications	Unsupervised	AE	Public malware API call sequence	ACC=99.2%
Yousefi-Azar et al. [22]	Malware classification (API calls based)	Publicly available malware applications (IoT)	Semi-supervised	AEs	Microsoft Malware Classification Challenge	ACC=96.3%
Ding et al. [184]	Malware detection (opcode based)	Publicly available malware applications	Unsupervised	DBN	netlux offensivecomputing Microsoft	ACC=96.7%
HaddadPajouh et al. [185]	Malware detection (opcode based)	ARM-based IoT applications	Semi-supervised	RNN	Virus Total Threat Intelligence platform Linux Debian repositories	ACC=98.2%

Reference	**Attack**	**Scenario**	**Learning paradigm**	**DL Model**	**Dataset**	**Performance**
Su et al. [202]	Android malware detection	Android apps	Semi-supervised	DBN SVM	Google Play Store Drebin Android Malware Genome Project Contagio Community	ACC=99.4%
Yuan et al. [203]	Android malware detection	Android apps	Semi-supervised	RBM	Google Play Store Contagio Community	ACC=96%
Yuan et al. [132]	Android malware detection	Android apps	Semi-supervised	DBN	Google Play store Contagio Community Genome Project	ACC=98%
Zhu et al. [204]	Android malware detection	Android apps	Semi-supervised	DBN	Google Play Store Android Malware Genome Project VirusShare	ACC =98%
Hou et al. [207]	Android malware detection	Android apps	Semi-supervised	SAE	Comodo Cloud Security Center	ACC=93.7%
Martinelli et al. [84]	Android malware detection and classification	Android apps	Supervised	CNN	Google Play store Drebin repository	ACC=95%
Karbab et al. [140]	Android malware detection	Android apps	Supervised	CNN	Malgenom Drebin Maldozer	F_1 Score=99%
McLaughlin [208]	Android malware detection	Android apps	Supervised	CNN	Android Malware Genome project McAfee Labs	ACC=98%
Lee et al. [211]	Android malware detection	Android apps	Supervised	CNN and RNN	VirusTotal	ACC=99%
Kim et al. [212]	Android malware detection	Android apps	Supervised	Multimodal DL method	Google Play App Store VirusShare Malgenome	ACC=98%
Torres et al. [29]	Botnet detection	IoT	Supervised	LSTM	Custom	TPR=96.8% FPR=1.11%
McDermott et al. [216]	Botnet detection	IoT	Unsupervised	BLSTM-RNN	Custom	ACC=99%
Alauthaman et al. [217]	P2P Botnet detection	IoT	Supervised	Multilayer feedforward NN	Custom	ACC=99.8%
Eslahi et al. [30]	Mobile HTTP Botnet detection	BYOD networks	Supervised	Feedforward Backpropagation NN	Genome project Drebin dataset	ACC=97.8%
Hill and Bellekens [223]	Ransomware detection	Computer networks	Supervised	DCNN	Custom	ACC=91%
Tseng et. al. [32]	ransomware detection	Computer networks	Supervised	DNN (7 layers, ReLU)	Custom	ACC=93%
Vinayakumar et al. [226]	ransomware detection and classification	Computer networks	Supervised	MLP	Custom	ACC=100%
Maniath et. al. [227]	ransomware detection	Computer networks	Supervised	LSTM	Custom	ACC=96.6%
Agrawal et al. [228]	ransomware detection	Computer networks	Supervised	ARI-LSTM	Custom	ACC=91%

Reference	Attack	Scenario	Learning paradigm	DL Model	Dataset	Performance
Homayoun et. al. [230]	ransomware detection and classification	FoG	Supervised	LSTM CNN	Custom	ACC=99%
Gharib and Ghorbani [31]	ransomware detection	Android mobile platform	Supervised	DAE	Custom	ACC=98.1%

C. *Privacy*

Nowadays, mobile devices have become common in our daily life. Users can benefit from a wide range of services offered by these devices (e.g recommendation systems, targeted advertising, health monitoring, and security surveillance). Most of these services are for free but they collect high sensitive user data (e.g. personal data, photos, videos or banking data). These services also access sensitive external data (e.g. surveillance systems or medical information). The principal beneficiaries are companies that exploit DL-based systems. These companies benefit from the vast amounts of data collected from their users, which lead them to have the monopoly of DL models. This poses important privacy issues for user personal data. Therefore, a great number of works are emerging in the literature to address these concerns. Figure 12 provides the complete picture.

Shokri and Shmatikov [33] is one of the first works on privacy. They develop a system for collaborative DL that preserves user privacy in all types of NNs. The system is based on MLP and CNN models and it enables multiple users to learn NN models based on their inputs, while benefiting from other user data without sharing the inputs. The DL algorithms are based on SGD because they can be parallelized and executed asynchronously. Furthermore, the model parameters can be selectively shared when training the model. Plus, they can be tuned to control the tradeoff between accuracy and privacy. The solution is evaluated using two datasets, MNIST [233] and SVHN [234] (both typically used for image classification). For the MNIST data set, the system achieves an ACC of 99.14% when participants share the 10% of their data. This result matches the centralized privacy-violating model (ACC=99.17%). For the SVHN dataset, ACC is 93.12%. Phong et al. [34] improve this work by ensuring that the system does not leake user data to the server while maintaining accuracy. They also improve the system security through homomorphic encryption with no impact on accuracy. Abadi et al. [37] propose a completely different approach. NNs are trained with differential privacy [235] to avoid the disclosure of private DL datasets information. The algorithms are based on a differential privacy enhanced SGD. They implement the solution using TensorFlow [148]. They test it using two popular image datasets: MNIST [233] and CIFAR-10 [236]. The system achieves an ACC of 97% for MNIST and 73% for CIFAR-10 in both cases for a differential privacy (8,10-5). Therefore, this work demonstrates that DNNs can be trained at a manageable cost in software complexity. Hitaj et al. [36] train DL structures locally and they only share a subset of the parameters obfuscated via differential privacy. The authors propose a solution to avoid attacks on collaborative DL using GANs. Their solution stops attackers from inferring sensitive information from the victim device. To this end, they first devise a novel class of inference attacks that are more generic than existing information extraction mechanisms. Then, they run the active inference attacks on the distributed collaborative learning system based on CNNs implemented by Shokri and Shmatikov [33]. To test the approach, they use the MNIST [233] and AT&T [237] datasets. They achieve an ACC of 97%.

Recent works address privacy issues in IoT. Osia et. al. [238][88] propose a hybrid framework, based on the CNN model, for efficient privacy preserving analytics. The proposal is based on the Siamese architecture [239]. It splits the NN into the IoT device and the Cloud. Feature extraction runs in the IoT device and classification in the Cloud. In this way, user raw data is not uploaded to the Cloud. Hence, it provides strong privacy guarantees to the system. The main innovation of this work is the feature extractor module that achieves an acceptable trade-off among accuracy, privacy and scalability. The solution is evaluated for two widely used classification tasks: gender classification and activity recognition. In gender classification, the datasets used are IMDB-Wiki [240] and LFW [241]. It achieves an ACC of 94%. In activity recognition, the dataset used is the MotionSense [242]. It achieves an ACC of 93%. In the same vein, Servia-Rodriguez et. al. [243] focus on Internet services that collect extensive user data, which can become invasive and comprise user privacy. The authors propose an alternative model that avoids the flow of user data to the Cloud. They propose to train the NN on distributed devices, which enables users to keep all rights over their data. The model is based on a two-step process that consists on a first analysis of a small dataset provided by voluntary users. The result of the analysis becomes a shared model. Then, they retrain the model locally (local model) using personal user data. At the end, each user has his/her own personal model. They evaluate their model for two learning tasks: supervised and unsupervised. The supervised model recognizes user activity from accelerometer traces. The system is based on MLP and trained using the WISDM Human Activity Recognition Dataset [244]. The authors compare the performance of the model between the shared, local, and personal models. Training the model with samples of other individuals, not only with samples of one user, achieves the best results. The unsupervised model uses the Latent Dirichlet Algorithm (LDA) to identify topics in a large set of documents. The model is trained using the NIPS dataset [245] and the Wikipedia latest English dump in January 2017 [246]. It achieves higher accuracy for the personal model than for the local one.

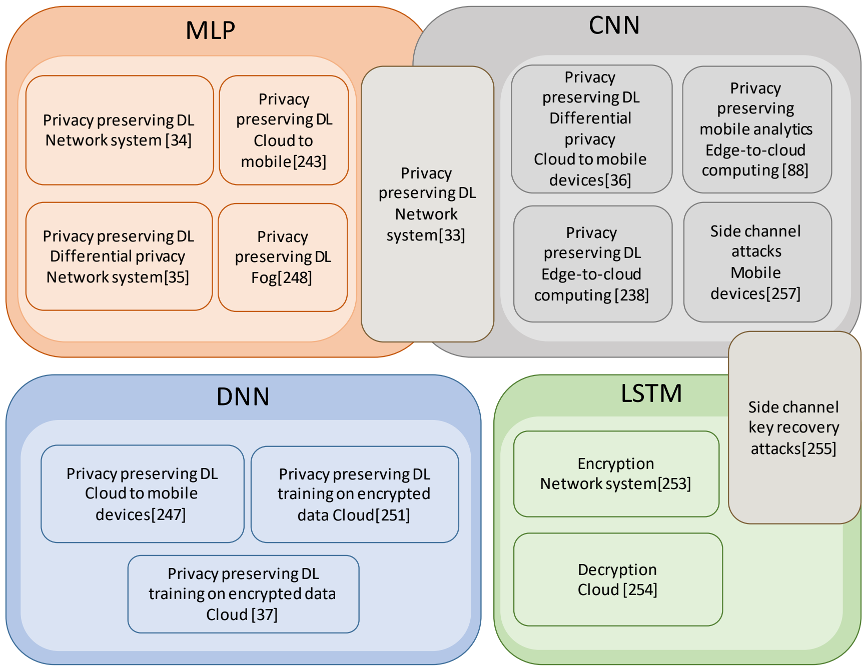

MLP
Privacy preserving DL Network system [34]
Privacy preserving DL Cloud to mobile[243]
Privacy preserving DL Differential privacy Network system[35]
Privacy preserving DL Fog[248]
Privacy preserving DL Network system[33]
CNN
Privacy preserving DL Differential privacy Cloud to mobile devices[36]
Privacy preserving mobile analytics Edge-to-cloud computing [88]
Privacy preserving DL Edge-to-cloud computing [238]
Side channel attacks Mobile devices[257]
DNN
Privacy preserving DL Cloud to mobile devices[247]
Privacy preserving DL training on encrypted data Cloud[251]
Privacy preserving DL training on encrypted data Cloud [37]
LSTM
Encryption Network system[253]
Decryption Cloud [254]
Side channel key recovery attacks[255]

Fig. 12. Overview of DL methods used in the privacy area.

Wang et al. [247] propose a similar solution to that of Servia-Rodriguez [243], but they consider Cloud resources, in addition to mobile devices, when partitioning the DNN. They develop the Arden framework for DNN-based private inference in mobile cloud. The solution uses a lightweight privacy-preserving mechanism, which consists of arbitrary data nullification and random noise addition, to protect sensitive information. Furthermore, authors propose a noisy training method, which injects deliberately noise into the training data to mitigate a negative impact on the performance of the cloud side. The framework is tested using the MNIST [233] and SVHN [234] datasets. It achieves an ACC of 98.02% and 88.12% respectively for each dataset. The experiments demonstrate that the Arden framework preserves user privacy and it also improves the inference performance reducing resource consumption over 60%. Finally, Lyu et al. [248] propose a novel approach that embeds Fog computing into DL to speed up computation and protect privacy in IoT. They devise a Fog-embedded privacy-preserving DL framework (FPPDL) to reduce computation and communication costs while preserving privacy. Privacy is preserved by a two-level protection mechanism. First, it uses Random Projection (RP) to protect privacy. They perturbe original data but preserve certain statistical characteristics of the original data. Second, Fog nodes train fog-level models applying Differentially Private SGD. The framework is implemented using MLP with two hidden layers using ReLU activations. It is evaluated using three different datasets typically used in image classification: MNIST [233], SVHN [234] and multiview multicamera dataset [249]. The solution achieves an ACC of 93.31% and 84.27% for the MNIST and SVHN datasets respectively. The results are slightly lower than for the centralized framework, but the solution reduces significantly communication and computation costs.

Another approach to guarantee user privacy in DL-based services is to train the network on encrypted data. Cryptographic techniques, as fully homomorphic encryption (FHE) [250], enable the processing of encrypted data. However, they are too slow for training DNN models due to the computational complexity and operations involved. Gilad-Bachrach et al. [251] propose Crypto-Nets that perform the inference phase of a NN on encrypted data. The solution is evaluated with the MNIST dataset. It achieves an ACC of 99%. On average, it achieves a sustained rate of 60,000 predictions/hour. However, this work has much room for improvement, especially in terms of throughput and latency. Nandakumar et. al. [37] improve this work and build the first fully homomorphic computationally efficient DL service for training on encrypted data. The key objective of this work is to outsource DL tasks to an external service, with the appropriate expertise and computational resources, without comprising user data. To this end, data is encrypted using a private key and it is subsequently shared with the service provider. Then, the service provider can train the model but it cannot learn anything about the data. The resultant model is only useful to the users with access to the private key. The solution is based on a DNN with two hidden layers, which use the SGD algorithm. The solution is implemented using the FHE toolkit HElib [252] and it is evaluated using the MNIST dataset. It achieves an ACC of 96%. The authors report a 50x speed-up in computation time when: (a) choosing the appropriate data representation; (b) simplify the network with minimum accuracy degradation; (c) pack data within the cipher text to minimize the number of operations; and (d) enable the parallelization of FHE computations. However, training in the encrypted domain is still too slow. It is about four or five orders of magnitude slower that training non-encrypted data.

DL models also have been explored to prevent information leakage and password guessing attacks. Recently, Liu et al. [253] propose a general DL model, so-called PL, to train datasets and generate passwords combining an LSTM with probabilistic context-free grammars (PCFG). The authors develop GENPass, a password-guessing generator based on the LSTM model. GENPass consists of several generators that create passwords from datasets and a classifier that checks that the output is not specific to a certain dataset. GENPass improves generality by implementing the adversarial idea in password generation. GENPass is implemented using Tensorflow [148] and it is trained and tested using a dataset that collects leaked English passwords from 4 websites: Myspace, phpBB, RockYou and LinkedIn. The results show that the matching rate improves by 20% compared to simply combining those datasets in cross-sites tests when learning from a single dataset.

Other works study different decryption DL methods. Greydanus et al. [254] propose the LSTM RNN to learn decryption algorithms. The novelty of this work resides in the proposed model, which can be applied to any polyalphabetic cipher. The solution can learn three different ciphers: Vigenere, Autokey, and Enigma. Once trained, the model has good performance on unseen keys and much longer ciphertext sequences. The model achieves an ACC of 99% for the three ciphers. Consequently, it is useful in cryptanalysis.

Other works in the literature address privacy vulnerabilities in side channel attacks. Maghrebi et al. [255] propose DL techniques for side channel key recovery attacks [256]. The authors compare the effectiveness and efficiency against different implementations of their proposed DL-based, ML-based and template-based attacks. The DL methods considered are AE, CNN and LSTM; while the ML methods are MLP and RF. The results demonstrate that DL attacks have better efficiency than common template and ML attacks in breaking unprotected and protected (AES) implementations. Ning et al. [257] address privacy vulnerabilities in mobile devices due to the malicious use of unsupervised sensor data. Today's smartphones integrate a wide variety of sensors (e.g. GPS, microphone, accelerometer, gyroscope, magnetometer, proximity, ambient). The authors report a new vulnerability due to the malicious use of unsupervised magnetometer and motion sensor data, where attackers sniff mobile applications and infer (using a CNN) the apps installed on the device and how frequently they are used. They achieve an ACC of 98%. The 6-layer CNN architecture is implemented using Tensorflow [148]. Each convolutional layer has 64 filters and the densely connected layer uses 128 neurons with ReLU activations. A dataset was created with the top 15 most used applications to validate the solution. Finally, this work proposes a noise injection scheme to effectively mitigate such attacks. The noise injection reduces the App sniffing ACC to 15%. Thus it mitigates the privacy leakage risk.

1) *Summary table –Privacy*

Table V summarizes the work analyzed in the privacy area. System performance is reported for all of the surveyed works, except for Maghrebi [255] (they do not use standard DL evaluation metrics).

TABLE V
SUMMARY OF DL WORKS ON PRIVACY

Reference	Objective	Scenario	Learning paradigm	DL Model	Dataset	Performance
Shokri and Shmatikov [33]	Privacy preserving DL	Network systems	Supervised	MLP CNN	MNIST	ACC=99.1%
					SVHN	ACC=93.1 %
Phong et al. [34]	Privacy preserving DL	Network system	Supervised	MLP	MNIST	ACC=99%
					SVHN	ACC=93 %
Abadi et al. [35]	Privacy preserving DL Differential privacy	Network system	Supervised	MLP	MNIST	ACC=97%
					CIFAR-10	ACC=73%
Hitaj et. al. [36]	Privacy preserving DL Differential privacy	Cloud to mobile devices	Supervised	CNN	MNIST AT&T	ACC=97%
Osia et. al. [238]	Privacy preserving mobile analytics	Edge-to-cloud computing IoT	Supervised	CNN	IMDB-Wiki LFW	ACC=94%
					MotionSense	ACC=93%
Osia et al. [88]	Privacy preserving DL	Edge-to-cloud computing (IoT)	Supervised	CNN	IMDB-Wiki LFW MotionSense	ACC=93%
Servia-Rodriguez et. al. [243]	Privacy preserving DL	Cloud to mobile devices	Supervised Unsupervised	MLP	WISDM Human Activity Recognition NIPS Wikipedia latest English dump	ACC=88%
Wang et. al. [247]	Privacy-preserving DL	Cloud to mobile devices	Supervised	DNN	MNIST	ACC=98%
					SVHN	ACC=88.1%
Lyu et. al. [248]	Privacy-preserving DL	Fog Computing IoT	Supervised	MLP	MNIST	ACC=93.3%
					SVHN	ACC=84.2%
Gilad-Bachrach et. al. [251]	Privacy-preserving DL Training on encrypted data	Cloud	Supervised	DNN	MNIST	ACC=99%
Nandakumar et. al. [37]	Privacy-preserving DL Training on encrypted data	Cloud	Supervised	DNN	MNIST	ACC=96%
Liu et al. [253]	Guessing passwords (Adversarial generation)	Network system	Supervised	LSTM	Custom	ACC=80%

Reference	Objective	Scenario	Learning paradigm	DL Model	Dataset	Performance
Greydanus et. al. [254]	Enigma learning	Cloud	Supervised	LSTM	Custom	ACC=99%
Maghrebi et al. [255]	side channel key recovery attacks	Mobile devices	Supervised	AE CNN LSTM	Custom	NA
Ning et al. [257]	side channel attacks	Mobile devices	Supervised	CNN	Custom	ACC=98%

VI. LESSONS LEARNED AND FUTURE DIRECTIONS

DL methods provide promising results to improve detection and accuracy of existing cybersecurity systems. They succeed in detecting new and complex attacks as they overcome the limitations of traditional and ML-based security systems. DL models provide resilience to new cyberattacks as novel attacks are usually small mutations of previously known attacks. They have self-taught abilities. This enables them to discover hidden patterns, different from normal behaviors, from the training data. The continuous update of the underlying model provides the capacity to learn the features of new attacks.

An important lesson learned for cybersecurity systems is that effective DL techniques are able to quickly transfer knowledge of existing attacks to improve the detection of newer ones, learn the features of newcomers and update the underling model. Research in this direction should consider transfer learning and lifelong learning. Deep lifelong learning [258] aims to build a solution that continuously adapts to new environments retaining the maximum knowledge from previous learning experiences. The model shows good results in non-stationary image data [259] or computer games [260] where it outperforms traditional DL algorithms. Deep transfer learning [261] uses previous knowledge of a specific domain to accelerate new learning processes since it does not learn from scratch. Transfer learning can help cybersecurity solutions as it reduces the time to respond to new threats.

Figure 13 provides a complete view of cybersecurity challenges in mobile networks and the learning paradigm most commonly used by existing cybersecurity systems.

A. Infrastructure area

1) Lessons learned

In the infrastructure area, most of the works in the literature address network intrusion detection. The architectures proposed for the cybersecurity systems are mainly based on unsupervised or semi-supervised learning paradigms. AEs are the preferred choice for intrusion and anomaly detection. KDDCUP'99 and NSL-KDDare the most widely used datasets. Yet, they do not represent perfectly existing real networks but, still, they are useful for comparing DL-based NIDS. A fair comparison of all the works in this survey is difficult due to two main factors. First and foremost, the datasets used are not always the same or they use different subsets of the same dataset. The second is that not all the works consider the same evaluation metrics when evaluating the solution. Nevertheless, some trends are visible for intrusion and anomaly detection works. All DL-based architectures outperform in terms of detection accuracy traditional IDS and ML-based IDS, even when considering real world data with embedded noise. Moreover, preprocessing the datasets enables higher accuracy, as well as higher efficiency in training and testing phases. Similarly, minimizing the number of features, by means of feature reduction methods, reduces data dimensions and computation complexity while preserving detection and classification accuracy. Finally, several works define the optimal hyper-parameters values to improve the efficiency of the system.

On the other hand, the analysis performed demonstrates that the ability of the architectures to detect intrusions depends on the type of attack and on the number of classes considered in the classification tasks. Another important conclusion is that the further an algorithm is trained with the latest features of attacks, the higher the detection of known and unknown attacks is. Consequently, attack detection systems should be frequently trained with updated samples.

In the IoT domain, the works analyzed demonstrate that parallel learning improves accuracy and efficiency of cyberattack detection. At the same time, performing the training and testing phases of the DL models on the IoT device leads to device dependent solutions.

The solutions for DDoS attacks are mainly based on supervised learning, being MLP the most used model. An interesting conclusion from this analysis is that the models provide better results if they are trained with up-to-date patterns. If so, they learn from new scenarios and detect zero-day patterns. Nevertheless, a major drawback of supervised learning is that it requires a high amount of labeled data, which is expensive to collect and it is not always is available. This will certainly lead to consider semi-supervised learning paradigms in the near future, which can effectively deal with the huge amount of unlabeled data for DDoS attacks. Distributed attack detection solutions also adopt supervised learning paradigms (mostly based on SAE). Surveyed works demonstrate that distributed attack detection can detect cyberattacks effectively and they can even improve centralized DL performance. The main advantage relies on parameter sharing during the training phase as it reduces local minima.

2) Future directions

In the cybersecurity field, and more specifically for intrusion and anomaly detection, an interesting research direction is to investigate if publicly available datasets are enough to train the learning algorithms to be generalized for new inputs in the given domain. In new areas, such as IoT, CPS or 5G, the major challenge when developing a DL-based solution is the generation of a realistic and high-quality training dataset. As datasets are the basis for obtaining the model knowledge, they should contain information that completely reflects real-world attacks. The completeness of the dataset has also a direct influence in the performance of DL-based cybersecurity models. In Cloud or IoT environments, crowd-sourcing methods are currently being introduced for generating rich threat datasets, which should be continuously updated with new attacks. However, the great diversity of IoT devices make it technically challenging. Another interesting research direction in IoT is to generalize the architectures to be applicable to devices of different vendors, and even to devices with different functionalities. Those architectures shall also consider the challenge of applying sophisticated security mechanisms to computationally limited devices. We foresee that DL models will be designed to achieve a trade-off between attack detection accuracy and the computing capabilities of the device.

Finally, existing DL-based security solutions appear not sufficient for the upcoming 5G mobile technology, where networks will have higher transmission rates. Novel DL models are in dire need in order to prevent systems from attacks considering the specific requirements (i.e. large-scale streaming, heterogeneous and low-quality data) without compromising the accuracy in detection with minimal response time.

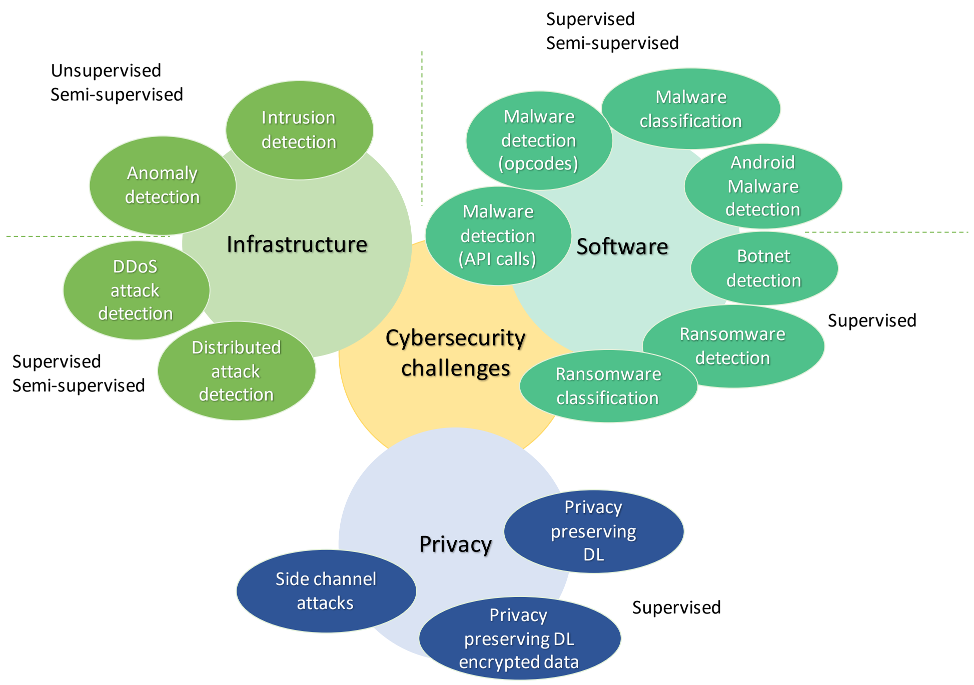


Fig. 13. Cybersecurity challenges in mobile networks.

B. *Software area*

1) *Lessons learned*

In the software area, DL-based solutions also outperform ML-based ones for malware detection and classification, botnet detection and ransomware detection and classification.

DL-based architectures are tested with different datasets, in this case even with a greater number of variants than in the infrastructure area. Furthermore, as obtaining legitimately malicious data is difficult, the ratio of benign and malicious data in the training dataset is unbalanced. This imbalance affects the performance of the solution. Therefore, standardized high-quality datasets are crucial for future improvements on DL-based cybersecurity systems. They are necessary to develop generalizable models to detect new malware.

Malware detection solutions fall in two main groups depending on the use of API calls or opcodes. API-call based solutions adopt supervised learning (mostly, RNNs and CNNs), or semi-supervised learning (AEs and LSTMs). Semi-supervised learning achieves better system performance while maintaining accuracy. Supervised solutions need labelled datasets which are scarce and they do not completely consider the wide range of malware attacks. Opcode based malware detection solutions consider unsupervised learning (RNNs) and semi-supervised learning (DBNs). Malware classification uses API-call based solutions, being AEs the preferred choice.

Android OS is the primary focus of malware solutions since 99% of all mobile device malware targets Android devices. Initially, the supervised learning paradigm (especially CNNs) was widely adopted. However, the vast amount of unlabeled data in this area, lead researchers to consider semi-supervised DL models. DBN has received special attention as it achieves a high accuracy with good runtime efficiency. Malware applications behavior is characterized in terms of sequences of systems calls. This captures the application behavior and, at the same time, it is robust to obfuscation techniques.

Finally, botnet detection and ransomware detection and classification use supervised models. Botnet detection solutions analyze network traffic to generalize common patterns of botnets along their life cycle. Through the works in this survey, we can see that undersampling techniques improve botnet detection and reduce computational cost. In contrast to traditional intrusion detection tools, supervised DL models can detect ransomware with behavioral changes and ransomware that uses packers to camouflage. Moreover, they discover the threat when the infection process starts. The most studied DL models for ransomware detection tools are CNNs and LSTMs.

2) *Future directions*

In the software area, the most important research direction is the development of publicly available high-quality datasets. These datasets should contain a large number of different possible attack types. Nowadays, different alternatives are under consideration to generate realistic and high-quality training datasets. One of the research directions is data augmentation to expand the limited available data by generating new samples from existing ones. The key challenge in malware data augmentation is to produce new samples that preserve the adequate data distribution for each class. This will improve classification accuracy of DL methods since improving the coverage of collected data translates to better detection capabilities of new, and existing, malware attacks. Works in the software area also highlight the need to continuously train the DL algorithms with the latest features of malware attacks to significantly increase detection accuracy.

The majority of the works design the proposed solution to have high accuracy detection but they do not consider computational costs, which is fundamental in real security products. Small computational costs are critical for mobile or edge devices with limited hardware. Consequently, DL-based solutions should be more efficient. For example, they should consider lightweight NNs and improve the preprocessing of input data. Acceleration at the edge should be explored, for example, by using network pruning techniques [262], as it has shown success in other areas as image recognition. These model optimization techniques eliminate unnecessary values in the weight tensor. It contributes to the development of more efficient NNs by reducing the computational cost of training.

Current works in the software area address the different types of attacks in an isolated way. Future research directions should consider interconnections between different malicious activities. For example, they should consider the cybersecurity attack lifecycle in terms of recognition, initial compromise, command and control target attainment and actions on the objective.

C. *Privacy area*

1) *Lessons learned*

Privacy is one of the major concerns in mobile networks. This area of cybersecurity is in its infancy and it requires further investigation. DL-based privacy preservation works mainly follow three different approaches: collaborative DL, differential privacy, and training on encrypted data. The vast majority of these works use supervised learning. Collaborative DL solutions use the MLP model and participants only share a subset of their data. If differential privacy is used, it avoids the disclosure of datasets. In this case, the DL models chosen are MLP and CNN. Finally, the proposals that train on encrypted datasets use DNNs (usually, with two hidden layers). These works achieve acceptable levels of accuracy. However, they still are too slow. They are about four or five orders of magnitude slower than training with non-encrypted data. Therefore, there is still room for improvement in all privacy areas.

Different proposals are starting to emerge in IoT. Privacy preserving analytics use preferably CNNs. In this area, semi-supervised learning is widely used. In the context of Fog computing, the majority of the works focus on reducing the computation and communication costs in IoT devices, comprising detection accuracy. Although these approaches are evolving and improving every day, it deserves further study in order to be able to adapt to constant changes.

2) *Future directions*

Privacy protection solutions are in an initial stage. It requires significant progress, especially, in the latency and throughput of NN training on encrypted data. Current systems outsource DL tasks to an external service with the appropriate expertise and computational resources without comprising user data; and thus, making the solution computationally efficient. However, it should consider also new alternatives (e.g. quantum computing techniques) to make the solution competitive.

Other future directions are shared with the infrastructure and software area: parallel learning and computational cost optimization. Several efforts are on the way, such as network pruning and the interplay between different malicious activities. Yet, this area is still in its infancy.

VII. CONCLUSION

Nowadays, the number of cyberattacks is increasing day by day, in number and in complexity, as technology evolves. In such a complex technological environment, traditional cybersecurity systems fail in the detection of complex unknown attacks such as zero-day attacks and new malware variants. ML techniques have been adopted by cybersecurity systems to address these challenges but with little success against unforeseen or unpredictable attacks. Meanwhile, DL techniques improve learning procedures and provide encouraging results in a wide range of applications, including cybersecurity. The success of DL relies, to a great extend, on the new achievements in software engineering and the massive generation of training data. This survey paper reviews DL methods applied to detect and classify all types of cyberattacks. To this end, a comprehensive analysis of DL techniques is done covering all cybersecurity aspects: intrusion detection, software attack detection and privacy preservation. For all the works reviewed, we analyze the architecture, giving a special attention to the DL method(s) used, its implementation, the data sets used for testing, and the results achieved. Whenever possible, we have compared the performance of the different proposals. It is worth noting that this was the most difficult part because most of the works do not use the same dataset for testing the model, especially in the software domain. While, others use specific subsets of the same dataset, especially in the infrastructure domain, in which most of the works use a subset of the KDDCUP'99 or NSL-KDD datasets. In the evaluation process, accuracy is usually reported, although several works use other metrics.

Finally, this paper provides a complete analysis and classification of DL methods used in cybersecurity. The major contributions of this paper are that it addresses all cybersecurity areas, including infrastructure, software and privacy, and it considers all different scenarios in mobile networks. Finally, it provides the relevant details of each proposal. To conclude, this survey paper aims to be a useful guide for researchers that start its work on DL-based cybersecurity systems.

LIST OF ACRONYMS

Acronym	**Description**
ACC	Accuracy
AE	Autoencoder
AI	Artificial Intelligence
ANIDS	Anomaly Detection-Based
ANN	Artificial Neural Network
API	Application Programming Interface
ARI	Attended Recent Inputs
ASN	Autonomous Systems
ASR	Automated Speech Recognition
AUC	Area Under the Curve
AWGN	Additive White Gaussian Noise
BLSTM-RNN	Bidirectional LSTM RNN
BM	Boltzmann Machine
BYOD	Bring Your Own Device
BPTT	Back-Propagation Through Time
CNN	Convolutional Neural Network
CVAE	Conditional Variational Autoencoder
CPS	Cyber-Physical System
C&C	Command and control
DAE	Deep Autoencoder
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DCNN	Dynamic Convolutional Neural Network
DNN	Deep Neural Network
DoS	Denial of Service
DDoS	Distributed Denial of Service
DL	Deep Learning
DR	Detection Rate
DRTHIS	Deep Ransomware Threat Hunting and Intelligence System
DT	Decision Trees
ESN	Echo State Network
FAP	File Access Pattern
FAR	False Alarm Rate
FHE	Fully Homomorphic Encryption
FN	False Negative
FP	False Positive
FPPDL	Fog-embedded privacy-preserving DL
FPR	False Positive Rate
GAN	Generative Adversarial Network
GFADS	GSA-based flow anomaly detection systematic
GRU	Gated Recurrent Unit
GSA	Gravitational Search Algorithm
HMM	Hidden Markov model
ID-CVAE	Intrusion detection CVAE
IDS	Intrusion Detection Systems
IoT	Internet of Things
ISTR	Internet Security Report
IT	Information Technology
JNNS	Java Neural Network Simulator
KNN	k-Nearest Neighbor
LBNL	Lawrence Berkeley National Laboratory
LDA	Latent Dirichlet Algorithm
LotL	Living off the Land
Acronym	**Description**
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
MLP-GA	Multilayer Perceptron with a Genetic Algorithm
MSA	Multiple Sequence Alignment
NB	Naive-Bayesian
NIC	Nature Inspired Computing
NIDS	Network Intrusion Detection Systems
NLP	Natural Language Processing
NN	Neural Networks
NSSA	Network Security Situation Awareness
OS	Operating System
p	precision
PCA	Principle Component Analysis
PCFG	Probabilistic Context-Free Grammars
PE	Portable Executable
PE	Portable Executable
P2P	Peer-to-peer
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RP	Random Projection
R2L	Remote-to-local
SAE	Stacked Autoencoder
SDN	Software-Defined Networking
SGD	Stochastic Gradient Descent
SMR	Soft-Max Regression
SNIDS	Signature-based
SNN	Spiking Neural Networks
SOM	Self-Organized Maps
STL	Self-taught learning-based
SVM	Suport Vector Machines
TN	True Negative
TP	True Positive
TPR	True Positive Rate
U2R	User-to-root
WSN	Wireless Sensor Networks
5G	Fifth-generation

ACKNOWLEDGMENTS

This work is supported by the Generalitat de Catalunya under grant 2017SGR962 and the DRAC project (001-P-001723).

REFERENCES

[1] Help Net Security, "Number of connected devices reached 22 billion, where is the revenue?" May, 2019. [Online] Available: https://www.helpnetsecurity.com/2019/05/23/connected-devices-growth/. Accessed on: Feb. 2, 2021.

[2] L. Kappelman et al. "The 2019 SIM IT Issues and Trends Study", MIS Quarterly Executive, (19:1), Article 7. 2020.

[3] Cisco Cybersecurity Report Series 2020 - Securing What's Now and What's Next. [Online] Available: https://www.cisco.com/c/en/us/products/security/cybersecurity-reports.html. Accessed on: Feb. 2, 2021.

[4] NDIA 2019 Cybersecurity Report. [Online] Available at: https://www.ndia.org/policy/cyber/2019-cybersecurity-report. Accessed on: Feb. 2, 2021.

[5] A.L. Buczak, E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection", IEEE Communications Surveys & Tutorials, vol. 18, no. 2, 2016, 1153-1176.

[6] D. Schatz, R. Bashroush, and J. Wall. "Towards a More Representative Definition of Cybersecurity", Journal of Digital Forensics, Security and Law, vol. 12, no. 2, 2017, 53-74.

[7] D. Kwon, H. Kim, J. Kim, S. Suh, I. Kim and J. Kim, "A survey of deep learning-based network anomaly detection", Cluster Computing, vol. 22, no. 5, January 2019, pp. 949-S961.

[8] N. Sultana, N. Chilamkurti, W. Peng, R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches", Peer-to-Peer Networking and Applications, Vol 12, 2019, pp. 493–501.

[9] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, vol. 521, May 2015, pp. 436-444.

[10] K. Bissell, R. Lasalle, P. Dal Cin, "Innovate for cyber resilience lessons from leaders to master cybersecurity execution", 2020. [Online] Available: https://www.accenture.com/_acnmedia/PDF-116/Accenture-Cybersecurity-Report-2020.pdf. Accessed on: Feb. 2, 2021.

[11] Symantec, vol. 24, February 2019, 1-59. [Online] Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf. Accessed on: Feb. 2, 2021.

[12] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, "DDoS in the IoT: Mirai and Other Botnets", Computer, vol. 50, no. 7, July 2017, pp. 80 - 84.
[13] McAfee mobile threat report. [Online] Available: https://www.mcafee.com/enterprise/en-us/threat-center/mcafee-labs/reports.html. Accessed on: Feb. 2, 2021.
[14] Check Point Software Security Report 2020. [Online] Available: https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf. Accessed on: Feb. 2, 2021.
[15] Symantec, [Online] Available: https://securitycloud.symantec.com. Accessed on: Feb. 2, 2021.
[16] Vectra's Cognito platform, [Online] Available: https://www.vectra.ai/product/what-it-is. Accessed on: Feb. 2, 2021.
[17] Sophos, [Online] Available: https://www.sophos.com/products/endpoint-antivirus.aspx. Accessed on: Feb. 2, 2021.
[18] IBM's QRadar Advisor tool, [Online] Available: https://www.ibm.com/support/knowledgecenter/SS42VS_SHR/com.ibm.apps.doc/c_Qapps_intro.html. Accessed on: Feb. 2, 2021.
[19] C. Zhang, P. Patras and H. Haddadi, "Deep learning in mobile and wireless networking: A survey", IEEE Communications Surveys & Tutorials, vol. 21, no. 3, March 2019, pp. 224-287.
[20] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," in IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1646-1685, 2020.
[21] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in Fog-to-Things computing", IEEE Communications Magazine, vol. 56, no. 2, February 2018, pp.169-175.
[22] M. Yousefi-Azar, V. Varadharajan, L. Hamey and U. Tupakula, "Autoencoder-based feature learning for cyber security applications", 2017 International Joint Conference on Neural Networks (IJCNN), July 2017, pp. 3854-3861.
[23] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue and K. Mizutani, "State-of-the-Art Deep Learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems", IEEE Communications Surveys & Tutorials, vol. 19, no. 4, May 2017, pp. 2432-2455.
[24] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and evolution", 2012 IEEE Symposium on Security and Privacy, May 2012, pp. 95-109.
[25] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu and, A. Thomas, "Malware classification with recurrent networks", in Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2015, pp. 1916-1920.
[26] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN", in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2017, pp. 2482-2486.
[27] G. E. Dahl, J. W. Stokes, L. Deng, D. Yu, "Large-scale malware classification using random projections and neural networks", in Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2013, pp. 3422-3426.
[28] X. Wang and S.-M. Yiu, "A multi-task learning model for malware classification with useful file access pattern from API call sequence", ArXiv, October 2016, pp. 1-7.
[29] P. Torres, C. Catania, S. Garcia and C. Garcia-Garino, "An Analysis of Recurrent Neural Networks for Botnet Detection Behavior", in Proceedings of the 2016 IEEE Biennial Congress of Argentina (ARGENCON), June 2016, pp. 1-6.
[30] M. Eslahi, M. Yousefi, M. Var Naseri, Y. M. Yussof, N. M. Tahir and H. Hashim, "Mobile botnet detection model based on retrospective pattern recognition", International Journal of Security and Its Applications, vol. 10, no. 9, 2016, pp. 39-44.
[31] A. Gharib and A. Ghorbani, "DNA-Droid: A Real-Time Android Ransomware Detection Framework", International Conference on Network and System Security NSS 2017, Lecture Notes in Computer Science, vol. 10394, 2017, pp. 184-198.
[32] A. Tseng, Y. Chen, Y. Kao and T. Lin, "Deep learning for ransomware detection", IEICE Technical Report, vol. 116, no. 282, IA2016-46, October 2016, pp. 87-92.
[33] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning", in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, October 2015, pp. 1310-1321.
[34] L. T. Phong, Y. Aono, T. Hayashi, L. Wang and S. Moriai, "Privacy-preserving deep learning: Revisited and enhanced", in L. Batten, D. Kim, X. Zhang and G. Li (eds) Applications and Techniques in Information Security, Communications in Computer and Information Science, vol. 719, June 2017, pp. 100-110.
[35] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar and L. Zhang, "Deep learning with differential privacy", in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, October 2016, pp. 308-318.
[36] B. Hitaj, G. Ateniese and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning", in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), November 2017, pp. 603-618.
[37] K. Nandakumar, N. Ratha, S. Pankanti and S. Halevi, "Towards deep neural network training on encrypted data", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1-9.
[38] E. Hodo, X. Bellekens, A. W. Hamilton, C. Tachtatzis and R. C. Atkinson, "Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey", ArXiv, 2017, pp. 1-43.
[39] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou and C. Wang, "Machine learning and deep learning methods for Cybersecurity", IEEE Access, vol. 6, 2018, pp. 35365 - 35381.
[40] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study". Journal of Information Security and Applications, vol. 50, 2020.
[41] CSE-CIC-IDS2018 Dataset. [Online] Available: https://www.unb.ca/cic/datasets/ids-2018.html. Accessed on: Feb. 2, 2021.
[42] Bot-IoT Dataset. [Online] Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. Accessed on: Feb. 2, 2021.
[43] S. Mahdavifar, A. Ghorbani, "Application of deep learning to cybersecurity: A survey", Neurocomputing, vol 347, 2019, pp. 149-176.
[44] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning", APSIPA Transactions on Signal and Information Processing, vol 3, no. 2, 2014, pp. 1-29.
[45] P. Subashini, et al., "Review on Intelligent Algorithms for Cyber Security", Handbook of Research on Machine and Deep Learning Applications for Cyber Security, 2020.
[46] D. S. Berman, A. L. Buczak, J. S. Chavis and C. L. Corbett, "A Survey of Deep Learning Methods for Cyber Security", Information, vol. 10, no. 4, January 2019, pp. 122-157.
[47] A. Singla and E. Bertino, "How deep learning is making information security more intelligent", IEEE Security & Privacy, vol. 17, no. 3, May-June 2019, pp. 56-65.
[48] R. Zachariah, K. Akash, M. S. Yousef and A. M. Chacko, "Android malware detection a survey", in Proceedings of the IEEE International Conference on Circuits and Systems (ICCS), December 2017, pp. 238-244.
[49] M. Scalas, D. Maiorca, F. Mercaldo, C. A. Visaggio, F. Martinelli and G. Giacinto, "On the effectiveness of system API-related information for Android ransomware detection", Computers & Security, vol. 86, September 2019, pp. 168-182.
[50] Hemdan, D. H. Manjaiah, "Digital Investigation of Cybercrimes Based on Big Data Analytics Using Deep Learning", Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications, 2020.
[51] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe and M. Manic, "Generalization of Deep Learning for Cyber-Physical System Security: A Survey", in Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society, October 2018, pp. 745-751.
[52] H. Sedjelmaci, F. Guenab, S. Senouci, H. Moustafa, J. Liu and S. Han, "Cyber Security Based on Artificial Intelligence for Cyber-Physical Systems", in IEEE Network, vol. 34, no. 3, pp. 6-7, May/June 2020.
[53] S. Zeadally, E. Adi, Z. Baig and I. A. Khan, "Harnessing Artificial Intelligence Capabilities to Improve Cybersecurity", in IEEE Access, vol. 8, pp. 23817-23837, 2020.
[54] J. Moor, The Dartmouth College Artificial Intelligence Conference: the next fifty years. AI Mag. 2006.
[55] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", Journal of Microbiological Methods, vol. 43, no. 1, 2000, pp. 3-31.
[56] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", The Bulletin of Mathematical Biophysics, vol. 5, no. 4, December 1943, pp. 115-133.

[57] F. Rosenblatt, "The percepton: A probabilistic model for information storage and organization in the brain", Psychologial Review, vol. 65, no. 6, 1958, pp. 386-408.

[58] M. Minsky and S. Papert "A Review of Perceptrons: An introduction to computational geometry", Information and Control, vol. 17, 1970, pp. 501-522.

[59] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems", Biological Cybernetics, vol. 52, 1985, pp. 141-152.

[60] R. Y. Choi et al., "Introduction to Machine Learning, Neural Networks, and Deep Learning", Translational Vision Science & Technology, Vol. 9, 2020.

[61] M. E. Karsligel, A. G. Yavuz, M. A. Güvensan, K. Hani , and H. Bank, “Network intrusion detection using machine learning anomaly detection algorithms”, in Proc. 25th Signal Process. Commun. Appl. Conf. (SIU), May 2017, pp. 1-4.

[62] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection” in Proc. IEEE Symp. Secur. Privacy, May 2010, pp. 305-316.

[63] A. Diro and N. Chilamkurti, “Leveraging LSTM networks for attack detection in Fog-to-Things communications”, IEEE Commun. Mag., vol. 56, no. 9, pp. 124 130, Sep. 2018.

[64] S. Pouyanfar, "A Survey on Deep Learning: Algorithms, Techniques, and Applications", ACM Computing Surveys, vol. 51, no. 5, 2018.

[65] Q. Mao, F. Hu and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey", IEEE Communications Surveys & Tutorials, vol. 20, no. 4, 2018, pp. 2595-2621.

[66] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, "A survey of deep neural network architectures and their applications", Neurocomputing, vol. 234, April 2017, pp. 11-26.

[67] M. Mohammadi, A. Al-Fuqaha, S. Sorour and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey", IEEE Communications Surveys & Tutorials, vol. 20, no. 4, 2018, pp. 2923-2960.

[68] S. Zhang, L. Yao, A. Sun and Y. Tay, "Deep learning based recommender system: A survey and new perspectives", ACM Computing Surveys, vol. 52, no. 1, 2019, pp. 1-5.

[69] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wila-mowski, “Selection of proper neural network sizes and architectures: a comparative study”, IEEE Transactions on Industrial Informatics, vol. 8, no. 2, February 2012, pp. 228-240.

[70] S.K. Pal, S. Mitra, "Multilayer perceptron, fuzzy sets, and classification", IEEE Transactions on Neural Networks, vol. 3, no. 5, 1992, 683 - 697.

[71] T. Teoh, G. Chiew, E. J. Franco, P. C. Ng, M. P. Benjamin, Y. J. Goh, "Anomaly detection in cyber security attacks on networks using MLP deep learning", 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), July 2018, pp. 1-5.

[72] C. Siaterlis and V. Maglaris, "Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics", 10th IEEE Symposium on Computers and Communications (ISCC), June 2005, pp. 469-475.

[73] K. J. Singh and T. De, "MLP-GA based algorithm to detect application layer DDoS attack", Journal of Information Security and Applications, vol. 36, October 2017, pp. 145-153.

[74] A. Saied, R. E. Overill and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks", Neurocomputing, vol. 172, no. 8, January 2016, pp. 385-393

[75] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning", MIT Press, November 2016.

[76] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series", in in: M. A. Arbib (eds.), the handbook of brain theory and neural networks, 1998, pp. 255-258.

[77] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture", The Journal of Physiology, January, vol. 160, no. 1, 1962, pp. 106-164.

[78] A. Waibel, T. Hanazawa, G. Hinton, K. Shikao and K. J. Lang, "Phoneme recognition using time-delay neural networks", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 3, 1989, 328-339.

[79] P. Y. Simard, D. Steinkraus and J. C. Platt, "Best practices for convolutional neural networks", in Proceedings of the Seventh International Conference on Document Analysis and Recognition, August 2003, pp. 958-963.

[80] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", 2015 IEEE Conference on Computer Vision and Pattern Recognition, June 2015, pp. pp. 1-9.

[81] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition, June 2016, pp. 770-776.

[82] G. Huang, Z. Liu, L. Van der Maaten, K. Q. Weinberger, "Densely connected convolutional networks", 2017 IEEE Conference on Computer Vision and Pattern Recognition, July 2017, pp. 2261-2269.

[83] M. Roopak, G. Yun Tian and J. Chambers, "Deep learning models for cyber security in IoT networks", 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 452-457.

[84] F. Martinelli, F. Marulli and F. Mercaldo, "Evaluating convolutional neural network for effective mobile malware detection", Procedia Computer Science, vol. 112, 2017, pp. 2372-2381.

[85] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning", 2017 International Conference on Information Networking (ICOIN), January 2017, pp. 712-717.

[86] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks", 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), July 2017, pp. 43-48.

[87] M. Lotfollahi, M. J. Siavoshani, R. S. Hossein-Zade and M. Saberian, "Deep packet a novel approach for encrypted traffic classification using deep learning", Soft Computing, 2019, pp. 1-14.

[88] S. A. Osia, A. S. Shamsabadi, A. Taheri, H. R. Rabiee and H. Haddadi, "Private and scalable personal data analytics using hybrid Edge-to-Cloud deep learning", Computer, vol. 51, no. 5, 2018, pp. 42-49.

[89] J. Kim, H. L. T. Thu and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection", 2016 International Conference on Platform Technology and Service (PlatCon), February 2016, pp. 1-5.

[90] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, vol. 5, no. 2, 1994, pp. 157-166.

[91] S. Hochreiter and J. Schmidhuber, "Long Short-Term memory", Neural Computation, vol. 9, no. 8, 1997, pp. 1735-1780.

[92] A. Graves, N. Jaitly and A-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM", 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, December 2013, pp. 277-278.

[93] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition", Sensors, vol. 16, no. 1, 2016, pp. 1-25.

[94] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embedings", ICML'16 Proceedings of the 33rd International Conference on Machine Learning, vol. 48, June 2016, pp. 526-534.

[95] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, December 2014, pp. 3104-3112.

[96] G. E. Hinton and R. S. Zemel, "Minimizing description length in an unsupervised neural network", April 1997.

[97] M. A. Ranzato, Y. L. Boureau and Cun Y. L., "Sparse feature learning for deep belief networks", in: J. C. Platts, D. Koller, Y. Singer and S. T. Roweis (eds.), Advances in Neural Information Processing Systems, vol. 20, 2008, pp. 1185-1192.

[98] V. L. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach", 2017 IEEE Wireless Communications and Networking Conference (WCNC), May 2017, 1-6.

[99] K. Sohn, H. Lee and X. Yan, "Learning structured output representation using deep conditional generative models", Advances in Neural Information Processing Systems, 2015, pp. 3483-3491.

[100] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes", International Conference on Learning Representations (ICLR), May 2014, pp. 1-14.

[101] M. Lopez-Martin, B. Carro, A. Sanchez-Esquevillas and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT", Sensors, vol. 17, no. 1967, August 2017, pp. 1-17.

[102] N. Le Roux and Y. Bengio, "Representational power of restricted Boltzmann machines and deep belief networks", Neural Computation, vol. 20, no. 6, June 2008, pp. 1631-1649.

[103] A. K. Noulas and B. J. A Kröse, "Deep belief networks for dimensionality reduction", Proceedings of the 20th Belgian-Dutch Conference on Artificial Intelligence, October, 2008, pp. 185-191.
[104] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines", Proceedings of the 25th International Conference on Machine Learning (ICML'08), July 2008, pp. 536-543.
[105] G. E. Hinton, "Learning multiple layers of representation", Trends in Cognitive Sciences, vol. 11, no. 10, October 2007, pp. 428-434.
[106] M. Welling, M. Rosen-Zvi and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval", in Advances in Neural Information Processing Systems, vol. 17, 2005, pp. 1481-1488.
[107] T. Kuremoto, M. Obayashi, K. Kobayashi, T. Hirata and S. Mabu, "Forecast chaotic time series data by DBNs", 2014 7th International Congress on Image and Signal Processing (CISP), October 2014, pp. 1130-1135.
[108] Y. Dauphin and Y. Bengio, "Stochastic ratio matching of RBMs for sparse high-dimensional inputs", Advances in Neural Information Processing Systems, vol. 26, 2013, pp. 1340-1348.
[109] Md. Z. Alom, V. Bontupalli and T. M. Taha, "Intrusion detection using deep belief networks", 2015 National Aerospace and Electronic Conference (NAECON), 2015, pp. 339-344.
[110] Y. Li, R. Ma and R. Jiao, "A hybrid malicious code detection method based on deep learning", International Journal of Security and Its Applications, vol. 9, no. 5, May 2015, pp. 205-216.
[111] M. Hossin, and N. Sulaiman, "A review on evaluation metrics for data classification evaluations", International Journal of Data Mining & Knowledge Management Process, Vol.5, No.2, 2015.
[112] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," Pattern recognition, vol. 30, no. 7, pp. 1145-1159, 1997.
[113] I. Ahmad, A. B. Abdullah, A. S. Alghamdi, "Application of artificial neural network in detection of probing attacks", 2009 IEEE Symposium on Industrial Electronics Applications, vol. 2, October 2009, pp. 557-562.
[114] J. J. Costa-Gondim, R. de Oliveira-Albuquerque, A. C. Alves-Nascimento, L. J. García-Villalba and T-H Kim, "A methodological approach for assessing amplified reflection distributed denial of service on the internet of things", Sensors, vol. 16, no. 11, November 2016.
[115] S. B. Wankhede, "Study of Network-Based DoS Attacks, Nanoelectronics", Circuits and Communication Systems, Lecture Notes in Electrical Engineering book serie, vol. 511, 2018, pp. 611-616.
[116] K. Hussain, SYN Flood Attack Detection based on Bayes Estimator (SFADBE) For MANET, 2019 International Conference on Computer and Information Sciences (ICCIS), 2019.
[117] L. Fernandez Maimó, Á. L. P. Gómez, F. J. G. Clemente, M. G. Pérez, and G. M. Pérez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks", IEEE Access, vol. 6, 2018, pp. 7700–7712.
[118] KDD cup 1999. [Online] Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed on: Feb. 2, 2021.
[119] NSL-KDD dataset for network-based intrusion detection systems. [Online] Available: https://www.unb.ca/cic/datasets/nsl.html. Accessed on: Feb. 2, 2021.
[120] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani. "A detailed analysis of the KDD CUP 99 data set", 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA), July 2009, pp. 1-6.
[121] A. Shiravi, H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", Computer & Security, vol. 31, no. 3, May 2012, pp. 357-374.
[122] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pp. 108-116.
[123] R. Vijayanand, D. Devaraj and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection", Computers & Security, vol. 77, August 2018, pp. 304-314.
[124] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection", 2013 IEEE Wireless Communications and Networking Conference (WCNC), April 2013, pp. 4487-4492.
[125] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: A preliminary analysis of ADFA-LD", 2013 6th International Congress on Image and Signal Processing (CISP), December 2013, pp. 1711-1716.
[126] A. Sperotto, R. Sadre, F. van Vliet and A. Pras, "A labeled data set for flow-based intrusion detection", in: G. Nunzi, C. Scoglio and X. Li (eds.) IP Operations and Management, IPOM 2009, Lecture Notes in Computer Science, vol. 5843, October 2009, pp. 39-50.
[127] EPA-HTTP -a day of HTTP logs from a busy WWW server available at http: //ita.ee.lbl.gov/html/contrib/EPA-HTTP.html
[128] The CAIDA UCSD DDoS Attack 2007 Dataset. [Online] Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml. Accessed on: Feb. 2, 2021.
[129] Google play store. [Online] Available: https://play.google.com/store/apps. Accessed on: Feb. 2, 2021.
[130] Contagio. [Online] Available: http://contagiodump.blogspot.com/. Accessed on: Feb. 2, 2021.
[131] Y. Zhou and X. Jiang, Android malware genome project. [Online] Available: www.malgenomeproject.org/. Accessed on: Feb. 2, 2021.
[132] Z. Yuan, Y. Lu and Y. Xue, "DroidDetector: Android malware characterization and detection using deep learning", Tsinghua Science and Technology, vol. 21, no. 1, February 2016, pp. 114-123.
[133] VirusShare. [Online] Available: https://virusshare.com/. Accessed on: Feb. 2, 2021.
[134] K. Maxwell, "Maltrieve". [Online] Available: https://github.com/krmaxwell/maltrieve. Accessed on: Feb. 2, 2021.
[135] Virus Total. [Online] Available: https://www.virustotal.com/gui/home. Accessed on: Feb. 2, 2021.
[136] D. Arp, M. Spreitzenbarth, H. Gascon and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket", in Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium (NDSS'14), February 2014, pp. 1-15.
[137] Microsoft malware classification challenge, Bit Data Innovators Gathering (BIG 2015), May 2015. [Online] Available: https://www.kaggle.com/c/malware-classification. Accessed on: Feb. 2, 2021.
[138] Netlux. [Online] Available: http://www.netluxantivirus.com/. Accessed on: Feb. 2, 2021.
[139] Offensivecomputing, [Online] Available: https://rubygems.org/gems/offensivecomputing/versions/0.1.1. Accessed on: Feb. 2, 2021.
[140] E. B. Karbab, M. Debbabi, A. Derhab and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning", Digital Investigation, vol. 24, March 2018, pp. 548-559.
[141] J.P. Anderson, "Computer security threat monitoring and surveillance", Technical Report, James P. Anderson Company, 1980.
[142] D. E. Denning, "An intrusion detection model", IEEE Transactions on Software Engineering, vol. SE-13, no. 2, February 1987, pp. 222-232.
[143] P. Mishra, V. Varadharajan, U. Tupakula and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection", IEEE Communications Surveys & Tutorials, vol. 21 , no. 1, 2019, pp. 686-728.
[144] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme", Soft Computing in Industrial Applications, Advances in Intelligent and Soft Computing, vol. 96, 2011, pp. 293-303.
[145] N. Gao, L. Gao, Q. Gao and H. Wang, "An intrusion detection model based on deep belief networks", Second International Conference on Advanced Cloud and Big Data, November 2014, pp. 247-252.
[146] Weka 3: Machine Learning Software in Java. [Online] Available: https://www.cs.waikato.ac.nz/ml/weka/. Accessed on: Feb. 2, 2021.
[147] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "Deep learning approach for network intrusion detection system", IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, February 2018, pp. 41-50.
[148] TensorFlow. [Online] Available: https://www.tensorflow.org/. Accessed on: Feb. 2, 2021.
[149] Q. Niyaz, W. Sun, A. Y Javaid, and M. Alam, "A deep learning approach for network intrusion detection system", BICT'15 Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BIONETICS), December 2015, pp. 21-26.
[150] R. Raina, A. Battle, H. Lee, B. Packer and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data", ICML'07 Proceedings of the 24th International Conference on Machine Learning, June 2007, pp. 759-766.

[151] X. Tao, D. Kong, Y. Wei and Y. Wang, "A big network traffic data fusion approach based on fisher and deep auto-encoder", Information, vol. 7, no. 2, March 2016, pp. 1-10.
[152] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink and J. Schmidhuber, "LSTM: A search space odyssey", IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 10, October 2017, pp. 2222-2232.
[153] T. A. Tang, L. Mhamdi, D. McLemon, S. A. Raza-Zaidi and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking", 2016 International Conference on Wireless Network and Mobile Communications (WINCOM), October 2016, pp. 258-263.
[154] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan and M. Sheikhan, "Flow-based anomaly detection using neural network optimized with GSA algorithm", 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, July 2013, pp. 76-81.
[155] P. Winter, E. Hermann and M. Zellinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines", 2011 4th IFIP International Conference on New Technologies, Mobility and Security, February 2011, pp. 1-5.
[156] C. Kolias, G. Kambourakis, A. Stavrou and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset", IEEE Communications Surveys & Tutorials, vol. 18, no. 1, January 2015, pp. 184-208.
[157] Q. Feng, Y. Zhang, C. Li, Z. Dou and J. Wang, "Anomaly detection of spectrum in wireless communication via deep auto-encoders", The Journal of Supercomputing, vol. 73, no. 7, July 2017, pp. 3161-3178.
[158] SourceFire, Inc., "Snort: An open source network intrusion detection and prevention system". [Online] Available: http://www.snort.org. Accessed on: Feb. 2, 2021.
[159] V. Richariya, U. P. Singh, and R. Mishra, Distributed approach of intrusion detection system: Survey, Int. J. Adv. Comput. Res., vol. 2, no. 6, pp. 358-363, 2012.
[160] Keras: The Python Deep Learning library. [Online] Available: https://keras.io/. Accessed on: Feb. 2, 2021.
[161] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like environment for machine learning", in Proc. BigLearn, NIPS Workshop, 2011, pp. 1-6.
[162] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods", Comput. Secur., vol. 45, 2014, pp. 100-123.
[163] Nexusguard, "DDoS Threat Report", 2018. [Online] Available: https://www.nexusguard.com/threat-report-q2-2018. Accessed on: Feb. 2, 2021.
[164] X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS attack via deep learning", 2017 IEEE International Conference on Smart Computing (SMARTCOMP), May 2017, pp. 1-8.
[165] T. Mitchell, Machine Learning, Chapters 3, 4, 6 and 7, McGraw-Hill Science/Engineering/Math, March 1997.
[166] Stuggart Neural Network Simulator, Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. [Online] Available: http://www.ra.cs.uni-tuebingen.de/SNNS/welcome.html. Accessed on: Feb. 2, 2021.
[167] M. Steve, "Preparing for the next DDoS attack", Network Security, 2013, vol. 5, pp. 5-6.
[168] S. Yadav and S. Subramanian, "Detection of application layer DDoS attack by feature learning using stacked AutoEncoder", 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), March 2016, pp. 361-366.
[169] A. A. Diro, N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things", Future Generation Computer Systems, vol. 82, May 2018, pp. 761-768.
[170] M. Zaharia, M. Chowdhury, M. J. Franlin, S. Shenker and I. Stoica, "Spark: Cluster computing with working sets", HotCloud'10 Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, June 2010, pp. 1-10.
[171] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using Autoencoder neural networks in WSN for IoT", 2018 International Conference on Communications (ICC), May 2018, pp. 1-6.
[172] S. Reece, S. Roberts, C. Claxton, D. Nicholson, "Multi-sensor fault recovery in the presence of known and unknown fault types", 2009 12th International Conference on Information Fusion, July 2009, pp. 1695-1703.
[173] W. Hardy, L. Chen, S. Hou, Y. Ye and X. Li, "DL4MD: A deep learning framework for intelligent malware detection", in Proceedings of the International Conference Data Mining (ICDM), December 2016, pp. 61-67.
[174] Y. Ye, L. Chen, S. Hou, W. Hardy and X. Li, "DeepAM: a heterogeneous deep learning framework for intelligent malware detection", Knowledge and Information Systems, February 2018, vol. 54, no. 2, pp. 265-285.
[175] Comodo anti-malware database. [Online] Available: www.malgenomeproject.org/. Accessed on: Feb. 2, 2021.
[176] R. Agrawal, J. W. Stokes, M. Marinescu, K. Selvaraj, "Neural Sequential Malware Detection with Parameters", 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2018, pp. 2656-2660.
[177] B. Kolosnjaji, A. Zarras, G. Webster and C. Eckert, "Deep learning for classification of malware system call sequences", in B. Kang and Q. Bai (eds.) Artificial Intelligence 2016: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 9992, December 2016, pp. 137-149.
[178] G. D. Webster, Z. D. Hanif, A. L. P Ludwig, T. K. Lengyel, A. Zarras and C. Eckert, "SKALD: A scalable architecture for feature extraction, multi-user analysis, and real-time information sharing", in Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016, Lecture Notes in Computer Science, vol. 9866, August 2016, pp. 231-249.
[179] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse and T. Yagi, "Malware detection with deep neural network using process behavior", in Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), June 2016, vol. 2, pp. 577-582.
[180] G. J. Tesauro, J. O. Kephart and G. B. Sorkin, "Neural networks for computer virus recognition", in IEEE Expert, August 1996, vol. 11, no. 4, pp. 5-6.
[181] W. C. Arnold and G. Tesauro, "Automatically generated win32 heuristic virus detection", in Proceedings of the 2000 International Virus Bulletin Conference, 2000, pp. 51-60.
[182] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals and L. Kaiser, "Multi-task sequence to sequence learning", ArXiv, March 2016, pp. 1-10.
[183] Y. Ki, E. Kim and H. K. Kim,"APIMDS (API-based malware detection system", 2016. [Online] Available: http://ocslab.hk security.net/apimds-dataset. Accessed on: Feb. 2, 2021.
[184] Y. Ding, S. Chen, J. Xu, "Application of Deep Belief Networks for opcode based malware detection", in Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), July 2016, pp. 3901-3908.
[185] H. HaddadPajouh, A. Dehghantanha, R. Khayami and K-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting", Future Generation Computer Systems, vol. 85, August 2018, pp. 88-96.
[186] V. Team, VirusTotal-Free Online Virus, Malware and URL Scanner. Available at: https://www.virustotal.com/gui/home/upload (last time revised on September 2020)
[187] Linux Packages Search. Available at: https://pkgs.org/ (last time revised on January 2020).
[188] Scikit-Learn, Machine learning in Python. Available at: https://scikit-learn.org/stable/ (last time revised on September 2020).
[189] Cisco 2018 Annual Cybersecurity Report. [Online] Available: https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf. Accessed on: Feb. 2, 2021.
[190] T. Berghoff, "Malware figures for Android rise rapidly", G DATA Blog, July 2018. [Online] Available: https://www.gdatasoftware.com/blog/2018/07/30937-malware-figures-for-android-rise-rapidly. Accessed on: Feb. 2, 2021.
[191] Y. Zhou, Z. Wang, W. Zhou and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets", in Proceedings of the 2012 Network and Distributed System Security (NDSS) Symposium (NDSS'12), February 2012, pp. 1-13.
[192] M. Grace, Y. Zhou, Q. Zhang, S. Zou and X. Jiang, "RiskRanker: Scalable and accurate zero-day Android malware detection", in Proceedings of the 10th International Conference on Mobile Systems, Applications and Services(MobiSys), 2012, pp. 281-294.
[193] V. Rastogi, Y. Chen and X. Jiang, "DroidChameleon: Evaluating Android anti-malware against transformation attacks", in Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (ASIA CCS), 2013, pp. 329-334.
[194] R. Fedler, J. Schutte, M. Kulicke, " On the effectiveness of malware protection on Android. An evaluation of Android antivirus Apps", Mobile Application Security, Fraunhofer AISEC, April 2013, pp. 1-36.

[195] C. A. Visaggio, G. Canfora, F. Mercaldo and P. Di Notte, "Metamorphic malware detection using code metrics", Information Security Journal: A Global Perspective, July 2014, vol. 23, no. 3, pp. 57-67.

[196] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware Android malware classification using weighted contextual API dependency graphs", in Proceedings of the 21st ACM Conference on Computer and Communications Security (ACM CCS'14), 2014, pp. 1105-1116.

[197] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for Android", in Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 2011, pp. 15-26.

[198] Y. Aafer, W. Du and H. Yin, "DroidAPIMiner: Mining API-Level features for robust malware detection in Android", in: T. Zia, A. Zomaya, V. Varadharajan, M. Mao (eds.) Security and Privacy in Communication Networks (SecureComm 2013), Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 127, 2013, pp. 86-103.

[199] K. A. Talha, D. I. Alper and C. Aydin, "APK auditor: Permission based Android malware detection system", Digital Investigation, vol. 13, June 2015, pp. 1-14.

[200] L. Apvrille and A. Apvrille, "Identifying unknown android malware with feature extractions and classification techniques", in 2015 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2015, pp. 182-189.

[201] M. Hubner H. Gascon D. Arp, M. Spreitzenbarth and K. Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In NDSS, 2014.

[202] X. Su, D. Zhang, W. Li, K. Zhao, "A deep learning approach to Android malware feature learning and detection", 2016 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), August 2016, pp. 244-251.

[203] Z. Yuan, Y. Lu, Z. Wang and Y. Xue, "Droid-Sec: Deep learning in Android malware detection", in Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 44, 2014, pp. 371-372.

[204] D. Zhu, H. Jin, Y. Yang, D. Wu and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data", in 2017 IEEE Symposium on Computers and Communications (ISCC), July 2017, pp. 438-443.

[205] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau and P. McDaniel, "FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps", ACM SIGPLAN Notices, vol. 49, no. 6, 2014, pp. 259-269.

[206] S. Rasthofer, S. Arzt and E. Bodden, "A Machine-learning approach for classifying and categorizing Android sources and sinks", in Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium (NDSS'14), February 2014, pp. 1-15.

[207] S. Hou, A. Saas, L. Chen and Y. Ye, "Deep4MalDroid: A deep learning framework for Android malware detection based on Linux kernel system call graphs", in 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), October 2016, pp. 104-111.

[208] N. McLaughlin, J. Martinez del Rincon, B. Kang, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé and G. J. Ahn, "Deep Android malware detection", in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (CODASPY'17), March 2017, pp. 301-308.

[209] Torch: A scientific computing framework for Luajit. Available at: http://torch.ch/ (last time revised on September 2020).

[210] V. Bushaev, "Understanding RMSProp -faster neural network learning". [Online] Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a. Accessed on: Feb. 2, 2021.

[211] W. Y. Lee, J. Saxe and R. Harang, "SeqDroid: Obfuscated Android Malware Detection Using Stacked Convolutional and Recurrent Neural Networks", Deep Learning Applications for Cyber Security, August 2019, pp. 197-210.

[212] T. Kim, B. Kang, M. Rho, S. Sezer and E. Gyu Im, "A multimodal deep learning method for Android malware detection using various features", IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, March 2019, pp. 773-788.

[213] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto and R. M. Salles, "Botnets: A survey", Computer Networks, vol. 57, no. 2, February 2013, 378-403.

[214] M. Oulehla, Z. K. Oplatková, D. Malanik, "Detection of mobile botnets using neural networks", in Proceedings of the IEEE Future Technologies Conference (FTC), December 2016, pp. 1324-1326.

[215] S. Garcia, "Malware Capture Facility Project", 2013. Available at: https://mcfp.weebly.com/ (last time revised on September 2020).

[216] C. D. McDermott, F. Majdani and A. V. Petrovski, "Botnet detection in the internet of things using deep learning approaches", in Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), July 2018, pp. 1-8.

[217] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem and M. Hossain, "A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks", Neural Computing and Applications, vol. 29, no. 11, June 2016, pp. 991-1004.

[218] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning", in Proceedings of the 9th Annual International Conference on Privacy, Security and Trust (PST), July 2011, pp. 174-180.

[219] M. Eslahi, R. Salleh and N. B. Anuar, "MoBots: A new generation of botnets on mobile devices and networks", IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), 2012, pp. 262-266.

[220] R. Richardson and M. North, "Ransomware: Evolution, Mitigation and Prevention", International Management Review, vol. 13, no. 1, 2017, pp. 10-21.

[221] P. Lestringant, F. Guihéry and P.-A. Fouque, "Automated identification of cryptographic primitives in binary code with data flow graph isomorphism," in Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, 2015, pp. 203-214.

[222] D. Xu, J. Ming and D. Wu, "Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping", in Proceedings of the 38th IEEE Symposium on Security and Privacy (SP), May 2017, pp. 921-937.

[223] G. D. Hill and J. A. Bellekens, "Deep learning based cryptographic primitive classification", ArXiv, September 2017, pp. 1-9.

[224] dpkt library. Available at: https://dpkt.readthedocs.io/en/latest/ (last time revised on December 2019).

[225] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization", Advances in Neural Information Processing Systems, vol. 4, 1992, 950-957.

[226] R. Vinayakumar, K. P. Soman and K. K. Senthil-Velan and S. Ganorkar, "Evaluating Shallow and Deep Networks for Ransomware Detection and Classification", in Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), September 2017, 259-265.

[227] S. Maniath, A. Ashok, P. Poornachandran, V. G. Sujadevi, A. U. Prem-Sankar and S. Jan, "Deep learning LSTM based ransomware detection", International Conference on Recent Developments in Control, Automation and Power Engineering (RDCAPE), October 2017, 442-446.

[228] R. Agrawal, J. W. Stokes, K. Selvaraj and M. Marinescu, "Attention in Recurrent Neural Networks for Ransomware Detection", in Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2019, 3222-3226.

[229] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", in Proceedings of the 3rd International Conference for Learning Representations (ICLR), May 2015, 1-15.

[230] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K. R. Choo and D. E. Newton, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer", Future Generation Computer Systems, vol. 90, January 2019, 94-104.

[231] Malwarebytes labs, "Look into locky ransomware-malwarebytes labs", March 2016. Available at: https://blog.malwarebytes.com/threat-analysis/2016/03/look-into-locky/. (last time revised on December 2019).

[232] Malwarebytes labs, "Cerber ransomware: New, but mature", March 2016. Available at: https://blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/ (last time revised on December 2019).

[233] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, vol. 86, no. 11, November 1998, pp.2278-2324.

[234] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Ng. "Reading digits in natural images with unsupervised feature learning", in NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning, December 2011.

[235] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy", Journal Foundations and Trends in Theoretical Computer Science, vol. 9, no. 3-4, August 2014, pp. 211-407.
[236] CIFAR-10 and CIFAR-100 datasets. [Online] Available: http://www.cs.toronto.edu/~kriz/cifar.html. Accessed on: Feb. 2, 2021.
[237] F. S. Samaria and A. C. Harter. "Parameterisation of a stochastic model for human face identification", in Proceedings of the Second IEEE Workshop on Applications of Computer Vision, December 1994, pp. 138-142.
[238] S. A. Osia, A. S. Shamsabadi, A. Taheri, H. R. Rabiee, N. D. Lane and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics", ArXiv, March 2017.
[239] S. Chopra, R. Hadsell and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, June 2005, pp. 539-546.
[240] R. Rothe, R. Timofte and L. Van Gool, "Dex: Deep EXpectation of apparent age from a single image," in Proceedings of the IEEE International Conference on Computer Vision Workshops, 2015, pp. 10-15.
[241] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments", University of Massachusetts, Amherst, Technical Report, October 2007, pp. 7-49.
[242] M. Malekzadeh, R. G. Clegg, A. Cavallaro and H. Haddadi, "Protecting sensory data against sensitive inferences", in Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems, ACM, June 2018, pp. 1-6.
[243] S. Servia-Rodriguez, L. Wang, J. R. Zhao, R. Mortier and H. Haddadi. "Personal model training under privacy constraints", in Proceedings of the 3rd ACM/IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI), April 2018, 1-11.
[244] J. R. Kwapisz, G. M. Weiss and S. A. Moore, "Activity recognition using cell phone accelerometers", ACM SigKDD Explorations Newsletter, vol. 12, no. 2, December 2010, pp. 74-82.
[245] D. Newman, "Bag of words data set". [Online] Available: https://archive.ics.uci.edu/ml/datasets/Bag+of+Words. Accessed on: Feb. 2, 2021.
[246] Wikipedia dataset. Available at: https://dumps.wikimedia.your.org/ (last time revised on September 2020).
[247] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud", in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, September 2018, pp. 2407-2416.
[248] L. Lyu , J. C. Bezdek, X. He and J. Jin , "Fog-Embedded Deep Learning for the Internet of Things", IEEE Transactions on Industrial Informatics, vol. 15, no. 7, April 2019, pp. 4206-4215.
[249] G. Roig, X. Boix, H. B. Shitrit and P. Fua, "Conditional random fields for multi-camera object detection", in Proceedings of the IEEE International Conference Computer Vision (ICCV), November 2011, pp. 563-570.
[250] C. Gentry, "Fully homomorphic encryption using ideal lattices", in Proceedings of the 41st ACM Symposium on Theory of Computing (STOC), June 2009, pp. 169-178.
[251] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig and J. Wernsing. "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy", in Proceedings of the 33rd International Conference Machine Learning, vol. 48 of JMLR Workshop and Conference Proceedings, 2016, pp. 201-210.
[252] S. Halevi and V. Shoup, "Helib: An Implementation of homomorphic encryption". [Online] Available: https://github.com/shaih/HElib/. Accessed on: Feb. 2, 2021.
[253] Y. Liu, Z. Xia, P. Yi, Y. Yao, T. Xie, W. Wang and T. Zhu, "GENPass: A general deep learning model for password guessing with PCFG rules and adversarial generation", in Proceedings of the IEEE International Conference on Communications (ICC), July 2018, pp. 1-6.
[254] S. Greydanus, "Learning the enigma with recurrent neural networks", ArXiv, August 2018, pp. 1-7.
[255] H. Maghrebi, T. Portigliatti and E. Prouff, "Breaking cryptographic implementations using deep learning techniques", in Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, 2016, pp. 3-26.
[256] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", CRYPTO, vol. 1666, 1999, pp. 388-397.
[257] R. Ning, C. Wang, C. S. Xin, J. Li and H. Wu, "Deepmag: Sniffing mobile apps in magnetic field through deep convolutional neural networks", in Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), March 2018, pp. 1-10.
[258] Z. Chen and B. Liu, "Lifelong machine learning," in Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 10, Morgan & Claypool, 2016, pp. 1-145.
[259] S.-W. Lee et al., "Dual-memory deep learning architectures for lifelong learning of everyday human behaviors," in Proc. Int. Joint Conf. Artif. Intell., New York, NY, USA, 2016, pp. 1669-1675.
[260] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in Proc. Nat. Conf. Artif. Intell. (AAAI), San Francisco, CA, USA, 2017, pp. 1553-1561.
[261] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," IEEE Commun. Surveys Tuts., vol. 19, no. 4, 2017, pp. 2392-2431.
[262] S. Tang and J. Han, "A pruning based method to learn both weights and connections for LSTM", 2015. [Online] Available: https://nlp.stanford.edu/courses/cs224n/2015/reports/2.pdf. Accessed on: Feb. 2, 2021.

Eva Rodríguez obtained her Ph. D in Computer Science in 2007 and her BSc in Telecommunication Engineer in 2001. She works at the Department of Computer Architecture of the Universitat Politècnica de Catalunya (UPC), since 2005, as Assistant Professor. From 2002 to 2005 worked as researcher in the Department of Technology of UPF (Universitat Pompeu Fabra). Her research focuses on security, privacy, multimedia information retrieval and object recognition. She is author of several papers, published in international journals and conferences, and she has been participating since 2003 in the ISO/MPEG standardisation group, contributing to different parts of the MPEG-21, MPEG-A and MPEG-M standards.

Beatriz Otero received her BSc. (1996) and M.Sc. (1999) degrees from Uni-versidad Central de Venezuela, and Ph.D. (2007) degree in Computer Architecture from the Universitat Politècnica de Cata-lunya (UPC). She is an Associate Professor at the Department of Computer Architecture of the UPC. In the last years Beatriz has carried out her research work in Modeling, Computing in Mathematics, Deep Learning and Parallel Computing. Beatriz has participated as a researcher in various research projects. She has published around 50 research articles in international conferences and in peer-reviewed journals, most of them in-dexed by JCR. Beatriz has also been a reviewer of more than 40 research articles at prestigious journals, and national and international conferences, as well.

Norma Gutiérrez is a student at the School of Telecommunications Engineering of the Universitat Politécnica de Catalunya (UPC). She is focused on the application of Deep Learning a real problems, especially those related to the cybersecurity and data privacy.

Ramon Canal received his M.S. (1998) and Ph.D. (2004) degrees from the Universitat Politècnica de Catalunya (UPC), in Barcelona, Catalonia, EU. He joined the faculty of the Computer Architecture Department of UPC in 2003. He finished his M.S. in the University of Bath (UK), worked at Sun Microsystems in 2000, and was a Fulbright visiting scholar at Harvard University in 2006/2007. His research focuses on power and thermal aware architectures, as well as reliability and security. He has an extensive list of publications and several invited talks. He has been program committee member in several editions of HPCA, ISCA, MICRO, HiPC, IPDPS, ICCD, ICPADS, CF. He has been co-general chair of HPCA 2016 and IOLTS 2012. He is a member of the IEEE.