A Comprehensive Survey of Databases and Deep Learning Methods for Cybersecurity and Intrusion Detection Systems

D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, *Senior Member, IEEE*

***Abstract*—This survey presents a comprehensive overview of Machine Learning (ML) methods for cybersecurity intrusion detection systems, with a specific focus on recent approaches based on Deep Learning (DL). The review analyzes recent methods with respect to their intrusion detection mechanisms, performance results, and limitations as well as whether they use benchmark databases to ensure a fair evaluation. In addition, a detailed investigation of benchmark datasets for cybersecurity is presented. This paper is intended to provide a road map for readers who would like to understand the potential of DL methods for cybersecurity and intrusion detection systems, along with a detailed analysis of the benchmark datasets used in the literature to train DL models.**

***Index Terms*—Cybersecurity, IDS, Deep Learning.**

I. Introduction

CYBERSECURITY systems have been of great importance since the beginning of the computer network era. However, security attacks emerged even before that: in 1941, Alan Turing cracked the Enigma machine, which was designed to cipher messages [1]. Similar incidents have continued to occur through the present day; for example, the electrical system of Massachusetts Institute of Technology (MIT) was hacked in 1950, Yahoo accounts were stolen in 2014, and several worldwide organizations were affected by the WannaCry worm in 2017 [2], [3]. According to the August 2019 threat reports from McAfee Labs, the top ten attack vectors at present are malware, account hijacking, unknown, vulnerability, unauthorized access, targeted attack, code injection, Denial of Service (DoS), defacement, and theft [4]. Hence, not only known but also unknown attack vectors currently pose a significant cyber threat.

The recent increase in the volume of data generated and transmitted over the internet, the need to manage the security of such data [5]–[7], and the continuing changes/evolution in intrusion types are causing both the academic and industrial communities to show increasing interest in the deployment of cybersecurity systems [8], [9]. To shield computer networks from attacks, Intrusion Detection Systems (IDSs) are being deployed to support user authentication, ensure safe access, and prevent loss of privacy. An IDS first collects and processes data and then applies a detection mechanism to raise alarms, which are sent to a human network analyst for further screening.

D. Gümüşbaş and T. Yıldırım are with the Department of Electronics and Communication Engineering, Yıldız Technical University, Istanbul, Turkey (e-mail: f0415058@std.yildiz.edu.tr; tulay@yildiz.edu.tr).

A. Genovese and F. Scotti are with the Department of Computer Science, Università degli Studi di Milano, Italy (e-mail: angelo.genovese@unimi.it; fabio.scotti@unimi.it).

Different IDSs can employ diverse algorithms for detecting attacks. These algorithms can be classified into three categories [10]: *i)* rule-based algorithms, which use prior knowledge of attacks, such as the corresponding data distributions, to create a rule system and perform detection; *ii)* statistics-based algorithms, which detect anomalies by building a statistical distribution of intrusion patterns; and *iii)* Machine Learning (ML)-based approaches, in which learning algorithms are adopted to train classifiers that can distinguish among different types of attacks.

Rule-based methods, while simple and fast to execute, cannot compensate for incomplete or noisy data and are difficult to update. To overcome these problems, statistics-based approaches have been proposed to enable the processing of imprecise information; however, such methods entail a high computational cost and have a limited ability to handle large quantities of data. Recently, ML-based approaches have increasingly been studied due to their ability to use complex inference models that can be trained on large quantities of data to detect complex intrusion patterns [11].

Due to the increasing quantities of data transmitted over the internet, which are leading to the introduction of new networking paradigms (e.g., the Internet of Things – IoT, cloud computing, and fog/edge computing [12], [13]) and complex inference models (e.g., Deep Learning (DL) [14], [15]), throughout the remainder of this paper, we will focus on ML-based approaches to cybersecurity and IDSs.

A. Previous Surveys

This subsection introduces previous surveys published in the literature on cybersecurity, with a specific focus on ML-based methods. These surveys were chosen based on the criterion of being either the most cited or a pacesetter review on a specific topic. In this paper, in contrast to other surveys in the literature, we summarize existing surveys based on their strong points. The objective is to help readers find further material in accordance with their interests.

One of the most cited surveys in the literature is presented in [11]. This survey addresses the different ML methods used in IDSs; describes the structure of Internet Protocol (IP) traffic features, such as port-based, payload-based, and statistical features; and provides insight into feature categories such as packet-level and flow-level features. Although this review was performed using a limited number of papers published between 2004 and 2007 and some of the benchmark datasets described are now out of circulation, it presents valuable information on how to discover and extract novel IP-based intrusion patterns/features from network traffic. The review presented in [16], in addition to describing the various ML-based methods of network intrusion detection, focuses on the characteristics of the types of intrusion. Therefore, this review presents how available statistical features can be used and modified for distributed attack detection and the importance of the threshold used to process these types of features.

In contrast to [11], [16], the survey presented in [17] focuses on the use of ML and Data Mining (DM) concepts in IDSs. This review includes a clear explanation of ML and DM algorithms introduced in highly cited papers published before 2016 as well as their usage in IDSs. Notably, this review does not include the newest DL methods, such as Convolutional Neural Networks (CNN); the newest datasets, such as AWID2018 and CICIDS2017; or practical details such as attack frequency and sample size for the benchmark datasets. Nevertheless, this review does consider fuzzy logic, neural networks, genetic algorithms, and rule-based algorithms.

Similar to the survey presented in [17], the work reported in [18] provides a review of ML methods for IDSs, associating different types of attacks with the features that can be used to detect them. In particular, the associated features can provide insight into how similar features of different types of intrusion can support similar approaches to attack detection. For example, the duration and service features from the KDD99 dataset are the most highly contributing features for detecting both User-to-Root (U2R) and Remote-to-Local (R2L) attacks, often causing these two attack types to be misclassified as one another. Although this paper fails to investigate the newest DL algorithms and attack types and their most related features, it provides an extensive survey of feature selection methods.

The review published in [19] surveys ML-based intrusion detection methods alongside newer DL-based methods. Although this survey focuses on certain specific ML and DL methods, such as Deep Belief Networks (DBNs) and Recurrent Neural Networks (RNNs), as well as known benchmark datasets, it does not cover other DL algorithms, such as CNNs, or benchmark datasets such as CICIDS2017. The reviews presented in [14], [20], [21] also consider DL-based methods. However, they focus on only a subset of these methods, do not discuss benchmark datasets, or do not provide detailed descriptions of the accuracies achieved using DL methods.

In contrast to the previously mentioned surveys, the work presented in [22] focuses on the different types of attacks rather than algorithms for IDSs, without providing details on accuracy. Furthermore, this paper presents an attack taxonomy to provide detailed definitions of various attack types, including how and in which layers they occur. Attack tools are also explained in great detail for readers who wish to build IDSs for protection against specific attack types. Although this paper does not provide detailed information about new benchmark datasets or DL algorithms, a brief review on industrial IDSs, such as programmable logic controller systems, is presented. Similarly, the review published in [23] addresses only application-layer Distributed DoS (DDoS) attacks, describing how they are hidden behind low traffic and the features used to detect DDoS attacks occurring in the application layer. Furthermore, this review discusses defense mechanisms for protecting against these attacks, such as user puzzles; the limitations of attempts to detect these attacks; and attack generation scenarios.

Finally, there are several surveys that address specific aspects or applications of IDSs. For example, the work reported in [24] focuses on IDSs for IoT systems, describing their taxonomy and placement strategies. In a similar manner, the review presented in [25] discusses DM concepts with IoT applications. Another example is the survey in [26], which covers only unsupervised methods used in IDSs. Although this review is limited to unsupervised methods, it is a good reference for learning about a variety of feature selection methods. Additionally, datasets and EU standards (e.g., the General Data Protection Regulation – GDPR) for data collection and protection are addressed in this review. Other reviews considering specific aspects of this field include the work described in [27], which focuses on hardware techniques for IDS implementation; the paper presented in [28], which considers only immunity-based approaches; and the survey published in [29], which describes network security techniques for supervisory control and data acquisition systems.

B. Contributions

This work is intended to serve as an extensive survey of databases and methods based on ML and DL that have been introduced thus far in the literature on cybersecurity and intrusion detection. This survey focuses on papers published after 2013, with some exceptions being trendsetter algorithms or highly cited papers.

Compared to the other surveys on intrusion detection discussed in Section I-A, this survey makes three main contributions: *i)* it summarizes previous surveys with regard to their level of detail in describing methods for cybersecurity, with the purpose of encouraging further reading based on the readers' interests; *ii)* it focuses on a practical perspective when describing the relevant datasets, specifically addressing the number of features, the feature types, and attack distributions rather than describing general details, feature selection methods, and algorithms, which are analyzed in other surveys; and *iii)* it presents a comprehensive investigation of the newest DL methods for intrusion detection, analyzing their detection capability, performance, and limitations as well as the databases used. This review does not consider previous types of ML methods since they have been thoroughly addressed in other survey papers [11], [17], [18].

The remainder of this paper is organized as follows. Section II presents a review of cybersecurity datasets, including the data collection steps, feature and attack types, benchmark datasets, and reliability criteria. Section III reviews and analyzes DL-based intrusion detection methods, considering DBNs, Autoencoders (AEs), CNNs, Long Short-Term Memory (LSTM) networks, and Generative Adversarial Networks (GANs). Section IV provides a discussion of and insights into the limitations and current research trends regarding public datasets and IDSs. Finally, Section V concludes this work. Table I summarizes the acronyms and notations used in this paper.

TABLE I
LIST OF ACRONYMS AND NOTATIONS USED IN THIS PAPER

Notation	Description
ML	Machine Learning
DL	Deep Learning
DM	Data Mining
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
GDPR	General Data Protection Regulation
PCAP	Packet CAPture
SSH	Secure Shell
FTP	File Transfer Protocol
SQL	Structured Query Language
SYN	TCP packet used to request a connection
DoS	Denial of Service
DDoS	Distributed Denial of Service
U2R	User-to-Root
R2L	Remote-to-Local
XSS	Cross-Site Scripting
k-NN	k-Nearest Neighbor
ANN	Artificial Neural Network
SVM	Support Vector Machine
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
AE	Autoencoder
CNN	Convolutional Neural Network
RNN	Recurrent Reural Network
LSTM	Long Short-Term Memory
GAN	Generative Adversarial Networks
PCA	Principal Component Analysis

II. CYBERSECURITY DATASETS

This section presents a review of cybersecurity datasets, outlining the data collection steps, feature and attack types, available benchmark databases, and reliability criteria.

A. Data Collection

This section presents the methods of data collection used in cybersecurity applications. Specifically, data collection can be performed in two different ways. The first is based on processing system calls (system logs) from host-based operating systems. The second is based on packet headers and payloads extracted from network traffic packages and from applications using the Transmission Control Protocol (TCP)/IP communication stack [30].

The two main methodologies used to collect network traffic in the second way are full Packet CAPture (PCAP) and the NetFlow protocol:

- *PCAP* enables the collection of the most detailed data from a network because it involves the extraction of whole network packets (including packet headers) for all information being transmitted. In particular, the data collected from such packets include the packet size, protocol types, headers of flows, flags, source and destination IP addresses, and source and destination port numbers [31]. However, the information contained in the payload of a packet may be deleted or anonymized due to privacy issues. In fact, a packet payload may contain sensitive data such as private information, instant messaging conversations, or a history of visited websites. In most cases, a trade-off must be established between anonymizing payloads to protect user privacy and using all collected data to achieve accurate attack detection. This trade-off is especially important to consider in the case of nonflooding attack types such as R2L and U2R attacks, which are performed using packet payloads.
- *NetFlow* enables the collection of summary information or certain predefined attributes related to the flow of packets in a network. Examples of the features that can be extracted include the number of packets in given a time period or the size of data transmitted over the network. Although data collection via NetFlow is more memory efficient than data collection via PCAP, only summary data are considered, and it is not possible to extract new types of features to address new needs.

TABLE II
PROGRAMS USED TO CAPTURE AND PREPROCESS NETWORK TRAFFIC

Method	Step	Program	Ref.
PCAP	Capture	libPCAP	[32]
		winPCAP	[33]
		SNORT	[34]
	Preprocessing	Wireshark	[35]
		tshark	[36]
		tcpdump	[37]
		networkminer	[38]
		rapidminer	[39]
		scapy	[40]
NetFlow	Capture/Preprocessing	Cisco NetFlow	[41]
		nfdump	[42]

The most commonly used programs for performing PCAP are libPCAP, winPCAP and SNORT. In addition, several programs allow the preprocessing of PCAP files to extract different types of features. For example, such preprocessing programs include Wireshark, tshark, tcpdump, networkminer, rapidminer and scapy. The most commonly used programs for capturing and preprocessing NetFlow data are Cisco NetFlow and nfdump. Table II summarizes the various programs used to capture and preprocess network traffic using the PCAP and NetFlow methodologies.

B. Feature Types

This section examines the types of features extracted from the available datasets. Although new features are added when novel attack patterns are discovered, there are several reoccurring feature types in the literature.

First, a distinction can be drawn between host-based and network-based data based on the procedure used to collect the data, as described in Section II-A. In most cases, host-based data are composed of system/operation logs, which consist of attributes such as system calls. Feature extraction from system calls is generally performed using methods based on natural language processing, such as n-grams [43].

On the other hand, network-based data are obtained by collecting network traffic data. However, network traffic is composed of many individual packets/frames, and feature extraction must be performed for each traffic session, known as flow-level traffic data, to reduce the dimensionality of the data and detect intrusions. Such feature extraction is conducted based on three different types of features: basic, traffic-based, and content-based features.

- *Basic features* are extracted from TCP/IP connections and can be classified as header-based, flow-based, connection-based, or packet-based features. Header-based features are related to the packet header and include the source and destination IP addresses, the TCP and User Datagram

Protocol (UDP) source and destination ports, the IP protocol, the service, and the IP header length. Flow-based features include attributes computed through analysis of the flow. In particular, a flow is defined as a set of packets having a common set of properties (flow keys), which may include IP addresses, port numbers, or meta-information [44]. Examples of flow-based features are statistical aggregations (e.g., average, maximum, minimum) on the size, time of arrival, and number of inbound/outbound packets in a given time period, the duration of that period, and the type of packets. Connection-based features are related to a particular connection, which is defined as a stream of packets between two specific IP addresses. Such features include the interval between packets, the timestamp, and the time to live. Finally, packet-based features are related to the transmitted data and include the payload and mean number of bytes of a packet. The main advantage of basic features is that they are general and can be used to detect several kinds of attacks [45], [46].

- *Traffic-based features* are associated with either a specific time interval (e.g., 2 seconds) or a specific number of connections (e.g., 100 connections). These features can be extracted by considering either the same host or the same service. In the first case, the extracted features include statistical sums of connections with the same destination host, whereas in the second case, the extracted features comprise statistical sums of connections to the same service for a fixed amount of time or number of connections [45]. One drawback of traffic-based features is that some attack types span time intervals longer than 2 seconds or a number of connections greater than 100. Examples of such attack types include low-frequency attack types such as U2R, R2L, and low-rate DoS attacks, in which the frequency of the transmitted information is similar to that of legitimate traffic, in contrast to high-frequency attack types, which exhibit a higher frequency than normal traffic. Although some newly proposed connection-based features span time intervals longer than 2 seconds, these features are not fully adequate for identifying such attack patterns [45].
- *Content-based features* are extracted from information embedded in different data portions of packets and include the number of requests, the request type, and the number of failed login attempts. Content-based features are especially useful for detecting low-frequency attack types, which do not exhibit sequential patterns as high-frequency attacks do. In fact, while traffic-based features can be used to detect high-frequency attacks, low-frequency attacks are difficult to detect using only basic and traffic-based features, and in most cases, content-based features are also required [45].

C. Attack Types

This section outlines the various attack types considered in IDSs. In particular, we present the following attack types, since they are the ones considered in the most frequently used benchmark datasets [48]:

- *Denial of Service (DoS)* [45] attacks are based on temporarily blocking the normal use of network utilities by flooding the network with traffic. Examples of DoS attacks include botnet, Slowloris, smurf, and SYN flood attacks.
- *Distributed DoS (DDoS)* [46] attacks are based on flooding the server and making it unable to respond by overloading it with service requests. Unlike in DoS attacks, the flooding is performed via many sources. Examples of DDoS attacks include local area network denial (LAND), ping-of-death, RUDY, and teardrop attacks.
- *User-to-Root (U2R)* [45] attacks involve behaving as a normal user with the aim of detecting system vulnerabilities and gaining root access. Examples of U2R attacks include buffer overflow, rootkit, Perl, and loadmodule attacks.
- *Remote-to-Local (R2L)* [45] attacks attempt to use a remote system to gain unauthorized access to and damage the target system. R2L attacks may be combined with U2R attacks, making these types of attacks difficult to differentiate. Examples of R2L attacks include Secure Shell (SSH) brute force, warezmaster, multihop, imap, and spy attacks.
- *Probe* [45] attacks are based on searching for vulnerabilities throughout the whole network by sending scan packets and gaining information about the system. Examples of probe attacks include Satan, IP sweep, and port sweep attacks.
- *Password* [18] attacks attempt to gain unauthorized access to the system by using guessing techniques to steal passwords. Examples of password attacks include brute force FTP-Patator and brute force SSH-Patator attacks.
- *Injection* [47] attacks use scripts that inject commands/queries with the purpose of gaining unauthorized access and stealing information. Examples of injection attacks include SQL injection and Cross-Site Scripting (XSS).

Table III lists the attack types considered in the most frequently used benchmark datasets, along with their definitions.

Although the definitions provided in Table III can be used to distinguish the different attacks, three additional factors must be considered when designing an IDS. First, an attack of one type may be the beginning of another attack of a different type. In this case, the characteristics of the true attack will be a combination of the characteristics of both attacks. Second, some attack characteristics may evolve over time. For instance, DDoS attacks are mostly understood to be high-frequency attacks that flood the bandwidth of a network; however, DDoS attacks in the application layer are low-frequency attacks that flood the server instead of flooding the network. Third, some attack types may show similar patterns. For example, both DoS and probe attacks, in most cases, exhibit sequential patterns and involve a large number of connections to the same host, whereas R2L and U2R attacks are both embedded in packets. Therefore, although DoS and probe attacks are easy to differentiate from R2L and U2R attacks, it may not be as easy to differentiate DoS attacks from probe attacks or

TABLE III
ATTACK TYPES REPRESENTED IN THE MOST FREQUENTLY USED CYBERSECURITY BENCHMARK DATASETS

Attack name	Examples	Description
Denial of Service (DoS) [45]	Botnet, Slowloris, smurf, SYN flood	Temporarily blocks the normal use of network utilities by flooding the network with traffic.
Distributed DoS (DDoS) [46]	LAND, ping of death, RUDY, teardrop	Floods the server and makes it nonresponsive to users by overloading it with service requests. Unlike in DoS attacks, the flooding originates from many sources.
User-to-Root (U2R) [45]	Buffer overflow, rootkit, Perl, loadmodule	Behaves as a normal user with the aim of detecting system vulnerabilities and gaining root access.
Remote-to-Local (R2L) [45]	SSH brute force, warezmaster, multihop, imap, spy	Gains local access via a remote system and damages the system. May be combined with U2R attacks, thus making these attacks difficult to differentiate.
Probe [45]	Satan, IP sweep, port sweep	Searches for vulnerabilities throughout the whole network via IP addresses by sending scan packets and gaining information about the system.
Password [18]	Brute force FTP-Patator, brute force SSH-Patator	Gains access to the system after stealing passwords by guessing.
Injection [47]	SQL injection, Cross-Site Scripting (XSS)	Uses a script to inject commands/queries to gain unauthorized access and steal information.

U2R attacks from R2L attacks due to their similar embedding patterns.

To increase the effectiveness of differentiating among attack types, several studies have investigated which types of features are effective for detecting particular attack types. For example, the authors of [18] report that on the basis of the features contained in the KDDCUP99 dataset, even though DoS attacks can be differentiated using basic and traffic-based features, considering some sparse features, such as flags, destination IP addresses, percentages of connections to the same service, and percentages of connections to the same port, can result in more effective detection. Similarly, duration, service, destination host same service rate, and flag features are vital for detecting probe (scanning) attacks. The most important features for detecting U2R attacks are the number of failed logins, number of shells, number of roots, duration, and service. For R2L attacks, the most important features are the duration, service, service bytes, destination bytes, number of failed logins, count, destination host count, and destination host service count. As seen above, the features used to detect attacks of the probe, U2R, and R2L types show a high degree of similarity, which explains why these three attack types are often misclassified among each other.

D. Benchmark Datasets

This section introduces and analyzes benchmark datasets for intrusion detection, considering both the extent to which they reflect novel attack types due to the evolving nature of intrusion patterns over time and their shortcomings. For the benchmark datasets considered in this section, Table IV lists the most frequently used datasets, while Table V summarizes the distribution of the samples in each dataset across the different attack types considered.

1) AWID2018: Also known as CSE-CIC-IDS2018, this dataset includes databases for training and testing collected using two different capture procedures. The data collected using the first procedure consist of full-packet network traffic with system logs, while the data collected using the second procedure consist of reduced packet traffic. The dataset includes two different labels for attacks: a main attack label and a subattack label. This dataset has the advantages of including the newest attack types, such as password attacks based on the SSH/FTP brute force approach, injection attacks based on SQL injection, and flooding attacks based on DoS. However, the data exhibit some limitations, such as noisy, misleading features and uncategorized samples. The dataset consists of 155 features extracted using Wireshark [49].

2) CICIDS2017: This dataset was created from realistic traffic data at the Canadian Institute for Cybersecurity of the University of New Brunswick (UNB) in 2017 and includes a full-packet dataset with 152 features and raw PCAP files [50]. The dataset considers attacks and subattacks such as injection attacks based on SQL injection and XSS, password attacks based on brute force FTP-Patator and brute force SSH-Patator, and flooding attacks based on DoS, Goldeneye DDoS, HULK DDoS, slow HTTP DDoS, Slowloris DDoS, and Heartbleed. Although the criteria for a reliable dataset proposed by [54] are satisfied, one feature among the attributes is duplicated.

3) KDD99/KDDCup99: Also known as KDDCup99, the KDD99 dataset was created using DARPA 1998 PCAP files and includes full-packet data, divided into subsets for training and testing [51].

This dataset considers DoS-based subattacks such as back, LAND, ping of death, teardrop, Neptune, and smurf attacks; U2R subattacks such as buffer overflow, loadmodule, Perl, and rootkit attacks; R2L subattacks such as ftp-write, guess-password, imap, multihop, PHF, spy, warezclient, and warezmaster attacks; and probe-based subattacks such as port sweep, IP sweep, NMAP, and Satan attacks. As of 2019, this dataset remains the most widely used benchmark dataset in the field of network intrusion detection. However, this dataset suffers from several limitations, including duplicated samples, different probability distributions between the training and test data, unbalanced classes, and a lack of coverage of the newest attack types.

4) NSL-KDD: This dataset was created by erasing all duplicate records from the KDD99 dataset and using sampling techniques to balance the number of data samples in each class [45]. This dataset includes separate databases for training and testing, where the test database consists of fourteen subattack types that are not present in the training database. NSL-KDD is not subject to most of the limitations of the KDD99 dataset; however, this dataset still lacks newer attack types.

5) Kyoto: This dataset was created from honeypots at Kyoto University and consists of traffic data collected daily between 2006 and 2015 [52]. The dataset includes 24 features, fourteen of which are in common with the KDD99 dataset, and labels indicating normal data, known attacks, and unknown attacks. The dataset is missing data from some days and months during the time of its collection, and the average

TABLE IV
OVERVIEW OF THE MOST FREQUENTLY USED CYBERSECURITY BENCHMARK DATASETS

Ref.	Name	Year	Num. of features	Num. of samples	Attack types	Separate train-test sets
[49]	AWID2018	2018	155*	210900113 (full) 2326218 (reduced)	Flooding, impersonation, injection	Yes
[50]	CICIDS2017	2017	152***	2830743	DoS/DDoS, port scan, FTP-Patator, SSH-Patator, bot, web attacks, infiltration, Heartbleed	No
[51]	KDD99	1999	42*	4900000 (full) 494021 (subset) 311029 (testing)	DoS, probe, U2R, R2L	Yes
[45]	NSL-KDD	2009	42*	125973 (training) 22544 (testing) 25192 (training) 11850 (testing)	DoS, probe, U2R, R2L	Yes
[52]	Kyoto	2006-2015	24*	Various	Known, unknown	No
[53]	UNSW-NB15	2015	49**	2540047 (full) 175341 (training) 82332 (testing)	Fuzzers, worms, shellcode, analysis, backdoors, DoS, exploits, generic, reconnaissance	Yes

Notes: * = including 1 feature as a label; ** = including 2 features as labels; *** = at the time of this survey.

TABLE V
DISTRIBUTIONS OF ATTACK TYPES IN THE MOST FREQUENTLY USED BENCHMARK DATASETS

Name	**AWID2018**									
Attack	*Normal*	*Flooding*	*Impersonation*	*Injection*						
N. samples	205074514	1409392	2361892	2054315						
(Perc.)	(97.24%)	(0.67%)	(1.12%)	(0.97%)						
Name	**CICIDS2017**									
Attack	*Benign*	*DoS*	*DDoS*	*Port scan*	*FTP-P.*	*SSH-P.*	*Bot*	*Web att.*	*Infiltr.*	*Heartb.*
N. samples	2273097	252661	128027	158930	7938	5897	1966	2180	36	11
(Perc.)	(80.3004%)	(8.9257%)	(4.5228%)	(5.6144%)	(0.2804%)	(0.2083%)	(0.0695%)	(0.077%)	(0.0012%)	(0.0003%)
Name	**KDD99**									
Attack	*Normal*	*DoS*	*Probe*	*U2R*	*R2L*					
N. samples	972781	3683370	41102	52	1126					
(Perc.)	(20.71%)	(78.4%)	(0.8897%)	(0.0001%)	(0.0002%)					
Name	**NSL-KDD**									
Attack	*Normal*	*DoS*	*Probe*	*U2R*	*R2L*					
N. samples	77054	53385	14077	252	3649					
(Perc.)	(51.9%)	(35.9%)	(9.5%)	(0.2%)	(2.5%)					
Name	**Kyoto**									
Attack	*Normal*	*Known attacks*	*Unknown attacks*							
N. samples	1186780	11218206	563							
(Perc.)	(9.5706%)	(90.429%)	(0.0004%)							
Name	**UNSW-NB15**									
Attack	*Fuzzers*	*Worms*	*Shellcode*	*Analysis*	*Backdoors*	*DoS*	*Exploits*	*Generic*	*Rec.*	
N. samples	24246	174	1511	2677	2329	16353	44525	215481	13987	
(Perc.)	(7.572%)	(0.054%)	(0.47%)	(0.83%)	(0.72%)	(5.112%)	(13.8%)	(67.092%)	(4.35%)	

Notes: N. samples = Number of samples; Perc. = Percentage; FTP-P. = FTP-Patator; SSH-P. = SSH-Patator; Web att. = Web attacks; Infiltr. = Infiltration; Heartb. = Heartbleed; Rec. = Reconnaissance. The largest remainder method was used when computing the percentages to ensure a total of 100%.

number of samples per month is approximately twelve million. Since the traffic was captured from honeypots, which are designed to protect against less advanced attackers, most of the monitored attacks did not originate from advanced attackers. Therefore, the dataset may not be representative of realistic attacks.

6) UNSW-NB15: This dataset was synthetically created at the Cyber Range Lab of the Australian Centre for Cybersecurity and includes full, training, and test datasets as well as raw PCAP files. The dataset includes 49 features and two label attributes: the first label describes the attack, and the second label is binary. The dataset considers attacks such as fuzzers, backdoors, shellcode, DoS attacks, worms, generic attacks, reconnaissance attacks, exploits, and analysis attacks [53]. One of the limitations of this dataset is the existence of several missing samples.

7) DARPA: This dataset was created at the MIT Lincoln Laboratory in 1998 and includes full, training, and test sets of raw PCAP files [55]. The newer versions of the DARPA dataset, DARPA 1999 and DARPA 2000, are based on the 1998 version. This dataset is one of the most commonly used intrusion detection datasets; however, it is commonly considered to be outdated and to contain irregularities [56].

8) ISCX IDS 2012: Also known as UNB or UNB ISCX 2012, this dataset was created at UNB in 2012 and includes full-packet network data [57]. The dataset includes normal traffic data and attack data for attack types such as infiltration, DoS, DDoS, and brute force SSH attacks. Although this dataset includes some of the newest attack types, it is criticized as being unrealistic for not containing sufficient internet background noise, as it consists of pure network traffic rather than data received by any real device [58].

9) CIC DoS: This dataset was created at the Canadian Institute for Cybersecurity of UNB in 2017 [59]. It considers the application layer and incorporates data that describe high-volume (traditional) DoS attacks, data corresponding to low-volume DoS attacks, and normal data from the ISCX IDS 2012 dataset.

10) Gure-KddCup: This dataset was created using the PCAP data from the DARPA 1998 dataset [60]. It includes features similar to those of the KDD99 dataset, with the addition of payload information and other new features, such as IP addresses and port numbers, to make U2R and R2L attacks more visible/distinguishable [61].

11) CDX: The Cyber Defence Exercises (CDX) dataset [62] was collected from the United States Military Academy network in 2009 and consists of PCAP data extracted from system logs, divided into intrusion traffic and normal traffic [56].

12) ASNM-CDX: This dataset was created from the CDX network traffic data in 2009. The dataset includes 5772 samples, each with 875+1+1 features. It includes distributed features often used in detecting low-frequency attacks, such as the number of packets and the total bytes in/out from four seconds to fifty-four seconds. In some cases, the features have been converted with the fast Fourier transform to increase their discriminative ability. This dataset has two attack label attributes: the first label discriminates between legitimate and malicious traffic, and the second label indicates whether the attack is based on buffer overflow. However, this dataset lacks traffic diversity since it consists only of buffer overflow attacks [63].

13) LBNL: This dataset was created at the Lawrence Berkeley National Laboratory (LBNL) between 2004 and 2005. Although the dataset includes packet headers, the payloads are anonymized due to privacy issues, which limits its informativeness [64].

14) ISOT: This dataset was created in 2010 by combining Storm, Waledac, and Zeus botnet attack data from the French Chapter of the Honeynet Project and normal traffic data from the Traffic Lab at Ericsson Research and LBNL [65].

15) MAWI: This dataset was collected by the MAWI Working Group in Japan and includes continuously updated traffic data from 2001 to 2019. A graph-based methodology has been used to label the raw data as either abnormal or normal [66]. One of the limitations of this dataset is duplicated packets.

16) CTU-13: This dataset is a combination of botnet traffic data, normal data, and background data collected at Czech Technical University in Prague (CTU) in 2011. Although the data consist of a variety of botnet scenarios and extended truncated versions of PCAP files with complete TCP, UDP and Internet Control Message Protocol (ICMP) headers, the dataset is specifically designed only for botnet detection. Therefore, it is considered unrealistic to mix these data with normal and background traffic [67].

17) UMass: This dataset was collected between 2004 and 2018 and contains traffic data such as Tor traffic data, Gateway Link 3 Trace data, web requests, and response data. However, most of the data were collected under similar network traffic conditions and lack a broad variety of attacks [68].

18) Twente: This dataset was created from honeypots at the University of Twente in 2009 and consists of more than fourteen million flows and more than seven million alerts. In this dataset, some samples are left unlabeled, and informative data from the packet headers and payloads are anonymized [69]. This dataset has the limitation that traffic originating from honeypots does not represent realistic attacks since honeypots are designed to protect against less advanced attackers.

19) CAIDA: The CAIDA dataset consists of a variety of different databases that are specific to particular events, such as network telescope and DDoS databases [58], [70]. Although there are a few up-to-date databases, such as CAIDA DDoS, most do not accurately represent the different possible types of attacks. For instance, the DoS attack databases consist only of spoofed-source DoS attacks and exclude other versions of DoS attacks.

20) DEFCON: The DEFCON datasets are created for intrusion modeling competitions held every year. Although these datasets are continuously created, they focus only on intrusions and attacks and lack normal background traffic [58]. Therefore, they are not frequently used for network intrusion detection.

21) Others: In addition to the most commonly used benchmark datasets, a variety of publicly available raw traffic datasets exist. These datasets include Metrosec, UNIBS 2009, TUIDS, the University of Napoli traffic dataset, payload datasets such as the CSIC 2010 HTTP Dataset, the UNM system call dataset, and an enormous variety of network traffic from the Capture the Flag Competitions (CTF) and CDX. Moreover, several host-based datasets also exist, including the ADFA Linux Dataset (ADFA-LD), the ADFA Windows Dataset (ADFA-WD) and the ADFA Windows Dataset Stealth Attacks Addendum (ADFA-WD:SAA) [71].

III. DL-based Intrusion Detection Methods

Traditional ML-based methods for cybersecurity include approaches based on the k-Nearest Neighbor (k-NN) algorithm, k-means clustering, Artificial Neural Networks (ANNs), fuzzy logic, Bayesian networks, hidden Markov models, self-organizing maps, decision trees, evolutionary classifiers, Support Vector Machines (SVMs), and rule-based systems [17], [18], [22], [26]. In this survey, we focus on the more recent DL-based approaches, which have not been covered in detail in previous surveys.

To provide up-to-date descriptions of the recent methods developed for cybersecurity, this section describes DL-based methods for intrusion detection. For each algorithm, we consider evaluation criteria such as a fast run/convergence time, a high detection ability with a low false positive rate, adaptability to novel intrusions, computational efficiency, and scalability [16]. In the remainder of this section, we consider DL methods based on DBNs, AEs, CNNs, LSTM networks, and GANs [15]. A summary of the presented DL methods in the IDS context is presented in Table VI.

A. Deep Belief Networks (DBNs)

DBNs are a type of ANN obtained by stacking together several Restricted Boltzmann Machines (RBMs [77]), which act as the layers of the DBN, and introducing connections between the layers but not within each layer. The RBMs used to construct a DBN consist of two main layers, one visible and one hidden, constituted by a variable number of neurons. Additionally, within each RBM, the neurons of different layers are fully connected, whereas the connections within the same layer are restricted [72]. Fig. 1 shows an example of a DBN.

Because of their layered structure, DBNs have the advantage that fast learning procedures can be used, which can be applied in a greedy fashion, layer by layer, in an unsupervised way [78]. As a consequence of this advantage, methods based

TABLE VI
SUMMARY OF DL-BASED METHODS FOR INTRUSION DETECTION

Method	Description	Pros	Cons
Deep Belief Networks (DBNs) [72]	Stacks of Restricted Boltzmann Machines (RBMs) with connections between the layers but not within each layer.	Fast and unsupervised layer-by-layer learning in a greedy fashion. Unsupervised dimensionality reduction.	Training uses an approximation of the gradient.
Autoencoders (AEs) [73]	Encoder-decoder structure that maps input data to a hidden space and then reconstructs them.	Can be trained in an end-to-end manner using learning algorithms based on gradient descent. Unsupervised dimensionality reduction.	Requires an additional ML model to perform classification.
Convolutional Neural Networks (CNNs) [74]	Sequences of convolutional layers trained via gradient descent.	Performs classification while automatically learning data representations. Learns discriminant spatial patterns invariant to translation and shifting.	Computationally expensive to train. Not naturally suited to processing data in time-series form.
Long Short-Term Memory (LSTM) [75]	Neurons arranged in a temporal sequence, able to maintain memory for arbitrary intervals of time.	Can natively process time-series data.	The research community is increasingly focusing on CNNs rather than LSTM networks.
Generative Adv. Networks (GANs) [76]	Combination of a generator, which generates data starting from a random distribution, and a discriminator, which distinguishes real data from synthetic data.	Learns data distributions in an unsupervised manner.	Often requires visual inspection of the results.

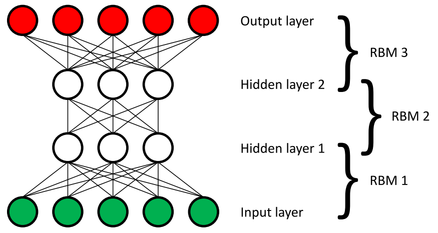

Fig. 1. Example of a Deep Belief Network (DBN). DBNs are obtained by stacking together several Restricted Boltzmann Machines (RBMs), which act as the layers of the DBN. In DBNs and RBMs, neurons of different layers are fully connected, while connections within the same layer are restricted.

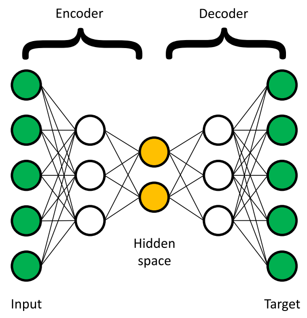

Fig. 2. Example of an Autoencoder (AE). The same data are used as both input and target: the encoder maps the input data to a hidden space in a nonlinear manner, and the decoder reconstructs the input data by mapping the encoded data back to the original input space.

on DBNs were among the first DL-based approaches studied for intrusion detection. In addition, the ability to train DBNs using fast and unsupervised learning algorithms makes them particularly suitable for performing a preliminary dimensionality reduction step, with the aim of extracting a compact and discriminant representation of the data, without the need for labels, even in the case of large intrusion detection databases. For example, in the method described in [79], proposed in 2011, a DBN was first applied to perform a feature reduction step, and an SVM was then used to classify the intrusions contained in the NSL-KDD dataset. Similarly, the approach proposed in [80] is based on a DBN, the parameters of which are first optimized via particle swarm optimization to map the input data to a space of reduced dimensionality. Then, a probabilistic neural network is trained to perform classification.

More recent methods have introduced ML architectures based on deeper DBNs, such as the approaches described in [81]–[83], which have achieved improved detection accuracy on the NSL-KDD and KDD99 datasets.

DBNs have the drawback that it is computationally unfeasible to train them end to end in a supervised way using gradient descent methods. Due to this drawback, in most cases, DBNs are trained using training algorithms based on contrastive divergence, which rely on an approximation of the gradient [84]. Recently, however, the growing availability of computing power and GPU-based training architectures (e.g., CUDA [85]) has made it possible to train DL models using end-to-end learning algorithms based on gradient descent, without the need to approximate the gradient [15]. Examples of recent DL models trained end to end include AEs, CNNs, LSTMs, and GANs.

B. Autoencoders (AEs)

AEs are a type of ANN used to learn and reconstruct a representation of input data. In AEs, the same data are used as both input and target, with the purpose of learning a model that can extract a compact and discriminant representation of the input data in an unsupervised manner. Such a representation can then be used as input to a classifier to perform detection. An AE consists of two components: an encoder and a decoder. The encoder maps the input data to a hidden space in a nonlinear manner, and the decoder reconstructs the input data by mapping the encoded data back to the original input space. The purpose of the decoder is to minimize the reconstruction error, defined as the difference between the input data and the reconstructed data [73]. Fig. 2 shows an example of an AE.

Because of their ability to extract a compact and highly discriminative representation with reduced dimensionality from input data, AEs are often used as a preprocessing step in intrusion detection. In most cases, the related approaches presented in the literature involve using AEs to preprocess the input data, followed by the application of an ML classifier. In contrast to DBNs, AEs can be trained in an end-to-end manner using learning algorithms based on gradient descent, without the need to approximate the gradient [15]. For example, in the method described in [86], an AE with seven layers is used to obtain a compact and discriminant representation of the input data. On the NSL-KDD dataset, this method achieves superior detection accuracy compared with other dimensionality reduction methods based on principal component analysis (PCA) and kernel PCA. Its main limitation is the lack of information about the shallow classifiers used to classify the data after the dimensionality reduction phase. Similar to that in [86], the method proposed in [87] involves applying an AE to the input data to extract features, which are then classified using shallow classifiers such as naive Bayes, k-NN, and SVM classifiers. This work considers the NSL-KDD dataset and reports higher accuracy than that achieved by previous methods, particularly when using the naive Bayes classifier.

Several methods in the literature combine an AE with a density estimation model to achieve greater detection accuracy. For example, the method proposed in [88] adopts a combined approach based on an AE and density estimation. This method achieves a high detection accuracy on the NSL-KDD dataset, especially for DoS and probe attacks. The method described in [89] extends the previous method by combining an AE with a Gaussian mixture model to perform intrusion detection. The model consists of an estimation network, which evaluates the densities of the samples in a low-dimensional space, and a compression network, which projects the data into a lower-dimensional space. A procedure based on joint parameter optimization is used to update the model parameters. On the KDD99 dataset, this method achieves a significant accuracy improvement compared to baseline methods using pretrained AEs.

A variant of AEs is represented by sparse AEs, which use a sparsity constraint to further reduce the dimensionality of the obtained representation [73]. Specifically, the method described in [90] uses a sparse AE combined with a softmax regression classifier to perform intrusion detection. The method achieves higher accuracy than previous models on the NSL-KDD dataset but considers only binary classification, differentiating between normal and anomalous traffic.

As the available computational power has increased, recent methods based on AEs have also considered Stacked AEs (SAEs) [91], which consist of several AEs trained separately and then "stacked" to obtain a deeper model and a more discriminant representation. The method proposed in [92] uses a model based on an SAE to preprocess raw traffic data in the CTU-13 dataset. Similarly, the method proposed in [93] uses an SAE to process traffic data captured from home wireless networks, representing several types of DDoS attacks. To improve the detection accuracy, the method introduced in [94] combines an SAE with a random forest classifier. The method has been tested on the KDD99 and NSL-KDD datasets by reducing the feature dimensionality of the input data and performing five-class classification. The method achieves high overall detection accuracy but exhibits low accuracy in detecting U2R and R2L attacks. Similarly, the method described in [95] achieves high detection accuracy on the KDD99 dataset based on a combination of four AEs. Instead of using a random forest classifier, the method presented in [96] combines SAEs with a radial basis function classifier to achieve high detection accuracy on the AWID2018 dataset.

Recently, one of the most commonly used types of AE models has been Variational AEs (VAEs) [97]. Their main novelty is that, whereas AEs use a deterministic discriminative model, VAEs use a probabilistic generative model to reconstruct the input data. As a consequence, VAEs are less prone to overfitting than AEs are. In the field of intrusion detection, the method described in [98] combines VAEs with a gradient-based fingerprinting detection model. In this method, gradient-based fingerprints are first extracted from NetFlow data taken from the UGR16 dataset [99], and VAEs are then applied for feature reduction. The work reported in [100] presents a more comprehensive evaluation conducted by combining VAEs with several classifiers, such as naive Bayes, SVM, decision tree, and random forest classifiers. This method achieves good results on the NSL-KDD and UNSW-NB15 datasets, especially when using the decision tree and random forest classifiers.

Despite the ability of AE-based methods to automatically obtain a compact and discriminant feature representation that can be adapted to the input data, AEs have the drawback of requiring an additional ML model to perform classification based on the obtained feature representation. To compensate for this drawback, the use of CNNs is increasingly being considered in recent DL-based methods developed for IDSs. This is because CNNs can be trained to process input data to automatically learn a compact and discriminant representation [15] while simultaneously classifying the obtained representation into the corresponding attack types [21].

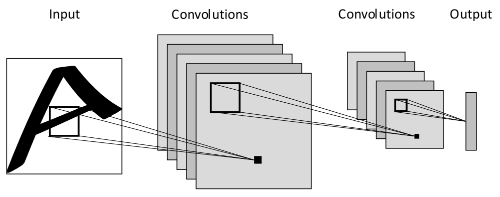

Fig. 3. Example of a convolutional neural network (CNN). The layers perform subsequent processing by convolving the data with banks of filters.

C. Convolutional Neural Networks (CNNs)

CNNs are a type of ANN in which the layers are structured to process input data in the form of multidimensional signals such as images or three-dimensional volumes. The different layers in a CNN perform subsequent processing by convolving the data with banks of filters, with parameters typically learned via gradient descent. CNNs have the main advantage of performing classification while automatically learning data representations, without the need for a handcrafted feature extraction step [74], [101]. Due to this advantage, CNNs have been successfully applied in several scenarios [15], [102]–[105]. Fig. 3 shows an example of a CNN.

In addition to their advantage of automatically learning data representations, CNNs have been especially successful in processing data with the aim of learning discriminant spatial patterns among the features that are invariant to translation and shifting [101]. In the context of IDSs, the advantages of CNNs have been useful for classifying attack types by considering the relationships among the features while requiring minimal preprocessing of the data [21].

Recently, however, the vast majority of CNN architectures have been structured to process data in the form of images [15]. Therefore, to use a CNN for intrusion detection, a preprocessing step must be performed to transform the features into a two-dimensional format that can be processed by the CNN. Several methods have been proposed for transforming features into a two-dimensional format. For example, the preprocessing method proposed in [106] converts feature attributes into binary vectors. The method converts symbolic attributes, such as flag, service, and protocol type attributes, into binary vectors using one-hot encoding [107]. Then, continuous attributes are converted by performing min-max normalization, discretizing the normalized values into ten intervals, and applying one-hot encoding. Finally, the obtained vectors are combined and reshaped to form a two-dimensional image. Similarly, the preprocessing approach presented in [108] converts malware binaries into grayscale images. A malware file is first read as a vector of 8-bit binary numbers, and each binary number is then converted into its equivalent decimal value. Finally, the resulting decimal vector is reshaped into a two-dimensional grayscale image.

Recent preprocessing methods for CNN-based intrusion detection have extended grayscale image representations to consider multiple channels. For example, the new encoding method proposed in [109] is designed to give equal weight to each feature, producing a feature representation with 24 bits for each pixel, similar to an RGB color image.

To accelerate the transformation process, some methods involve performing feature selection before converting the data into an image-based format. For example, in the method described in [110], feature selection is applied via a genetic algorithm.

After the features are transformed into an image-based format, a CNN-based approach is applied to classify the obtained images and perform intrusion detection. Several such approaches have been proposed in the literature. Most of these CNN-based methods rely on a layer structure based on an existing architecture. In particular, several methods use architectures based on LeNet [74], ResNet [111], GoogLeNet [112], or VGG-16 [113]. The LeNet architecture is used in the methods described in [114]–[118], which achieve high detection accuracy, especially on the AWID2018 dataset [119]. Similarly, the ResNet and GoogLeNet architectures are used in the method proposed in [106], which has been tested on the NSL-KDD dataset after preprocessing. Although satisfactory results are achieved on this dataset, detection rates are not reported for each class. The VGG-16 architecture is used in the method presented in [108], which has been applied to the Malign Dataset and the Microsoft Malware Dataset. This method achieves almost the same accuracy as the winner of the Microsoft Malware Dataset Challenge [120].

In addition to methods based on existing networks, there are some CNN-based methods for which innovative ML architectures have been proposed. For example, the authors of [121] propose a novel CNN and present its application to the KDD99 dataset. The method achieves a high classification accuracy for five types of DoS attacks, exhibiting performance superior to that of naive Bayes and k-NN classifiers. An innovative architecture is also proposed in [122] based on a convolutional AE; however, this method has been tested only on a private dataset.

Methods based on CNNs have recently achieved high accuracy on several intrusion detection datasets due to their ability to simultaneously learn a compact representation of the input data and perform adaptive classification. However, CNNs are

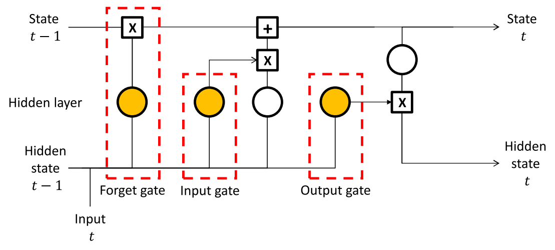

Fig. 4. Example of a Long Short-Term Memory (LSTM) network. The neurons are connected following a temporal sequence. The forget, input, and output gates control which information is preserved in the network and passed to the next time step.

primarily useful for learning discriminant spatial patterns from input data, whereas they are not naturally suited for processing data in the form of time series with the intent of learning discriminant temporal patterns. To overcome this disadvantage, several recent methods have considered the use of LSTM networks, which are specifically structured for learning temporal patterns by processing new data while maintaining a memory of previous samples [75].

D. Long Short-Term Memory (LSTM) Networks

An LSTM network is a type of ANN based on an RNN in which the neurons are connected following a temporal sequence. However, in contrast to traditional RNNs, LSTM networks have a deeper structure of hidden neurons with the ability to maintain a memory of previous inputs for arbitrary intervals of time [75]. Due to this node arrangement, RNNs and LSTM networks are often used to process data in the form of time series [123]. Fig. 4 shows an example of an LSTM network.

The ability of LSTM networks to process time-series data has proven useful in the IDS context since datasets for cybersecurity and intrusion detection are often structured as sequences of features evolving over time. Due to this advantage, several intrusion detection methods in the literature are based on LSTM networks [124]. Among these methods, the approach proposed in [125] applies a three-layer LSTM network. It achieves high detection accuracies on the KDD99, ADFA-LD, and UNM datasets. Similarly, the method proposed in [126] uses a cascade of three LSTM network modules, combined using a voting mechanism, to achieve an increased intrusion detection accuracy.

To exploit both the accuracy of LSTM networks in processing time series and the capability of CNNs to extract spatial patterns from images, recent methods have increasingly considered combinations of LSTM and CNN architectures for intrusion detection. For example, the method proposed in [127] uses an LSTM network combined with a CNN to perform multiclass detection of anomalies in the KDD99 dataset. Similarly, the approach described in [128] uses both a CNN and a hybrid LSTM-CNN model to perform detection. In some cases, hybrid LSTM-CNN models have been developed based on existing architectures, such as the method developed in [129], which relies on a CNN designed based on the LeNet model. This method has been tested on recent databases, including CICIDS2017 and CTU-13.

Despite the ability of LSTM networks to natively process time-series data, the introduction of novel and advanced CNN

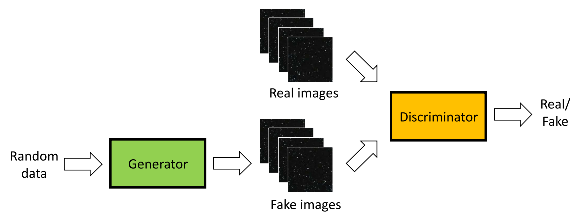

Fig. 5. Example of a Generative Adversarial Network (GAN), composed of a generator, which generates data starting from a random distribution, and a discriminator, which distinguishes real data from synthetic data. The generator and discriminator are trained in an alternating fashion and, in most recent architectures, have CNN-based structures.

architectures is shifting the attention of the research community towards the use of CNNs in a wider range of application scenarios, including the learning of temporal patterns [130]. In fact, current research trends are increasingly focusing on CNN architectures that are deeper (e.g., ResNet) [111] or lighter in weight (e.g., MobileNet [131]) and on computing platforms specifically designed to accelerate the training of such architectures [85]. Consequently, CNN-based methods tend to outperform models based on recurrent architectures, such as LSTM-based models, in most cases [132].

E. Generative Adversarial Networks (GANs)

GANs are DL models that can learn and mimic the distribution of input data to generate synthetic samples with a strong resemblance to the original data. Specifically, a GAN is structured as a combination of a generator, which generates data starting from a random distribution, and a discriminator, which distinguishes real data from synthetic data. The generator and discriminator are trained in an alternating fashion until equilibrium is reached [76], [133]. In most recent applications, the generator and discriminator of the GAN are structured as CNNs, with the consequence that recent GANs can generate synthetic image samples with a high degree of realism [134]. Fig. 5 shows an example of a GAN.

GANs have the main advantage of being able to learn the distribution of the input data in an unsupervised manner, that is, without requiring class labels. In the IDS context, this characteristic is useful for learning the characteristics of data distributions in specific situations (e.g., under normal conditions). Due to this advantage, many recent methods developed for IDSs use GANs trained on existing datasets to detect anomalies, where the training data include only data captured in specific situations. Among these methods, a CNN-based GAN is introduced in [135] to learn the characteristics of data captured under normal conditions. Then, the method is used to detect anomalies by computing the distance between freshly captured data and normal data. To achieve faster detection, an algorithm is proposed in [136] that improves the computational efficiency of the GAN described in [135] while achieving a similar accuracy on the KDD99 dataset.

Despite the ability of GANs to simulate input data distributions, the synthetic data generated from the learned distribution may be insufficiently realistic compared with the real data, and thus, manual (e.g., visual) inspection may be required to achieve good results. In the case of IDSs, such visual examinations of the feature vectors could be relatively difficult to perform compared with cases in which a GAN is used to generate images of known objects or people.

IV. Discussion

This section presents a discussion of the limitations, challenges, and research trends of the current databases and intrusion detection approaches for cybersecurity applications. In particular, we will focus on the issue of dataset reliability and on research directions regarding novel features for intrusion detection.

A. Dataset Reliability

The recent rise in the number of ML-based approaches, particularly those based on DL, has resulted in an increase in the accuracy of intrusion detection that can be achieved using state-of-the-art methodologies. However, the performance of DL-based methods strongly depends on the quantity and quality of the data available [15], with the consequence that the biases and limitations of the datasets used to train the models directly affect the reliability of the predictions.

Currently, although the majority of works focus on researching algorithms that can yield improved detection results, a few studies have been dedicated to evaluating the reliability of benchmark datasets. As a first step towards evaluating dataset reliability, the work proposed in [137] discusses the criteria for a reliable benchmark dataset, which concern the diversity of the traffic data, the diversity of the protocols, the volume of collected data, the diversity of the attacks considered, the inclusion of novel attack types, the inclusion of full payloads without anonymization, the presence or absence of informative features, the updatability, the consideration of realistic traffic, the extent of labeling, and the size of the feature set. Finally, any discussion of dataset reliability should consider the ability of a dataset to adapt to changes over time, for example, by mimicking statistically normal traffic in accordance with upcoming needs.

Similarly, the work described in [54] proposes eleven criteria for assessing the reliability of a dataset for intrusion detection: *i)* attack diversity, *ii)* anonymity, *iii)* available protocols, *iv)* complete capture (with payloads), *v)* complete interaction, *vi)* complete network configuration, *vii)* complete traffic, *viii)* feature set, *ix)* heterogeneity (all network traffic and system logs), *x)* correct labeling, and *xi)* metadata (full documentation of data collection). In addition to complying with these criteria, a reliable dataset should also provide a means of anonymizing the payload information to guarantee users' privacy [9].

Among the criteria considered in [54], [137], attack and traffic diversity play a major role, since a limited diversity or a high imbalance among attack types might increase the bias of detection approaches towards specific situations. To limit the problem of model bias and enable accurate evaluation of detection algorithms, it is therefore crucial to consider datasets that are as free as possible from internal biases while also being sufficiently representative of real-world data. However, dataset bias has been considered mainly for the benchmark datasets used in the field of computer vision [138], [139], whereas there is no analysis in the literature of the bias of benchmark datasets for intrusion detection. Therefore, an evaluation of dataset bias for IDSs may contribute to a fairer assessment of the various algorithms that have been proposed in the field of cybersecurity.

In addition to dataset bias, a few works in the literature address other issues related to public benchmark datasets, such as repeated data, missing values, incorrect labeling [137], or an optimistic number of false alarms due to considering specific situations in a nonrealistic way [140].

B. Novel Features

As the number of methodologies that are able to achieve high accuracy on known datasets increases, attack patterns tend to evolve to better cheat the existing IDSs. This evolution, which can arise in nonstationary environments, is known as concept shift and occurs as the definitions of attacks change over time [141].

For instance, the work presented in [142] shows that some low-frequency DDoS attacks that appear in newer datasets exhibit a higher degree of similarity to normal data traffic than do similar attacks in older datasets. As a consequence, in recent cases, some features are less effective in detecting such attacks than they are in detecting older attack patterns.

Therefore, it remains an open research issue to investigate whether the available features in known benchmark datasets are sufficient to achieve high detection rates even in the presence of changing attack patterns or whether it will be necessary to add new features to maintain a high level of detection accuracy.

V. Conclusion

In this review, we have analyzed Machine Learning (ML)-based approaches to cybersecurity and intrusion detection systems, with a specific focus on the most recent methods based on Deep Learning (DL), which represent the current state of the art for intrusion detection in network traffic. Specifically, we have considered methods based on deep belief networks, autoencoders, convolutional neural networks, long short-term memory networks, and generative adversarial networks. In contrast to previous surveys, this review considers studies that use common benchmark datasets to ensure a fair evaluation and comparison of the proposed algorithms.

To provide a reference for how recent cybersecurity methods use benchmark datasets for intrusion detection, in this survey, we have also reviewed the main datasets used for this purpose by highlighting their potential for training effective ML-based algorithms. In particular, we have considered the data collection procedures, the distributions of feature and attack types, and dataset reliability criteria.

By providing a survey of ML and DL approaches, along with descriptions of the benchmark datasets considered when developing recent methods, this review aims to provide a practical road map for researchers in academia and industry working in the field of ML and DL for cybersecurity applications.

References

[1] S. Muggleton, "Alan Turing and the development of artificial intelligence," *AI Commun.*, vol. 27, no. 1, pp. 3–10, 2014.

[2] "WannaCry ransomware attack," https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

[3] "Hacked consumers don't forgive companies who lose their data. bad news for yahoo," https://secludit.com/en/blog/consumer-hacking-confidence.

[4] McAfee, "Mcafee labs threats report," https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf, 2019.

[5] R. Bhadoria, "Security architecture for cloud computing," in *Cyber Security and Threats: Concepts, Methodologies, Tools, and Applications*, 2018, pp. 729–755.

[6] M. Swarnkar and R. Bhadoria, "Security aspects in utility computing," in *Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing*, 2016, pp. 262–275.

[7] S. Dorbala and R. Bhadoria, "Analysis for security attacks in cyber-physical systems," in *Cyber-Physical Systems: A Computational Perspective*, 2015, pp. 395–414.

[8] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Syst. J.*, vol. 9, no. 2, pp. 350–365, 2015.

[9] R. Sandhu and P. Samarati, "Authentication, access control and intrusion detection," in *CRC Handbook of Computer Science and Engineering*. CRC Press Inc., 1997, pp. 1929–1948.

[10] S. Han, M. Xie, H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: Techniques and challenges," *IEEE Syst. J.*, vol. 8, no. 4, pp. 1052–1062, 2014.

[11] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 2008.

[12] R. Donida Labati, A. Genovese, V. Piuri, F. Scotti, and S. Vishwakarma, "Computational intelligence in cloud computing," in *Recent Advances in Intelligent Engineering: Volume Dedicated to Imre J. Rudas' Seventieth Birthday*. Springer, 2020, pp. 111–127.

[13] Y. Cai, A. Genovese, V. Piuri, F. Scotti, and M. Siegel, "IoT-based architectures for sensing and local data processing in ambient intelligence: Research and industrial trends," in *Proc. of I2MTC*, 2019.

[14] 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 Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 2017.

[15] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, 2018.

[16] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 2014.

[17] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2016.

[18] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 2019.

[19] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of Deep Learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, 2017.

[20] E. Hodo, X. J. A. Bellekens, A. W. Hamilton, C. Tachtatzis, and R. C. Atkinson, "Shallow and Deep networks intrusion detection system: A taxonomy and survey," *ArXiv*, vol. abs/1701.02145, 2017.

[21] 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, pp. 35 365–35 381, 2018.

[22] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. C. Atkinson, and X. J. A. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *CoRR*, vol. abs/1806.03517, 2018.

[23] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 661–685, 2019.

[24] B. B. Zarpelo, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, no. C, pp. 25–37, 2017.

[25] C. Tsai, C. Lai, M. Chiang, and L. T. Yang, "Data mining for internet of things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 77–97, 2014.

[26] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, 2018.

[27] R. Abdulhammed, M. Faezipour, and K. M. Elleithy, "Network intrusion detection using hardware techniques: A review," in *Proc. of LISAT*, 2016, pp. 1–7.

[28] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection – a review," *Nat. Comput.*, vol. 6, no. 4, pp. 413–466, 2007.

[29] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer, "Security challenges in control network protocols: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 619–639, 2019.

[30] O. Savas and J. Deng, *Big Data Analytics in Cybersecurity*. Auerbach Publications, 2017.

[31] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2019.

[32] "LibPCAP," https://www.tcpdump.org.

[33] "WinPCAP," https://www.winpcap.org.

[34] "Snort," https://www.snort.org.

[35] "Wireshark," https://www.wireshark.org.

[36] "tshark," https://www.wireshark.org/docs/man-pages/tshark.html.

[37] "TCPDump," https://www.tcpdump.org.

[38] "Networkminer," https://www.netresec.com/?page=NetworkMiner.

[39] "Rapidminer," https://rapidminer.com.

[40] "Scapy," https://scapy.net.

[41] "Cisco Netflow," https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.

[42] "Nfdump," https://github.com/phaag/nfdump.

[43] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. Prentice Hall PTR, 2000.

[44] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 2014.

[45] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. of CISDA*, 2009.

[46] X. Jing, Z. Yan, and W. Pedrycz, "Security data collection and data analytics in the internet: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 586–618, 2019.

[47] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL Injection and XSS attacks," in *Proc. of PRDC)*, 2007, pp. 365–372.

[48] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," *Managing Cyber Threats*, vol. 5, pp. 19–78, 2005.

[49] University of the Aegean, "AWID2018 dataset," http://icsdweb.aegean.gr/awid/features.html, 2018.

[50] Canadian Institute for Cybersecurity, "Intrusion Detection Evaluation Dataset (CICIDS2017)," https://www.unb.ca/cic/datasets/ids-2017.html, 2017.

[51] University of California, Irvine (UCI), "KDD Cup 1999," http://www.kdd.org/kdd-cup/view/kdd-cup-1999, 1999.

[52] Kyoto University, "Traffic Data from Kyoto University's Honeypots," http://www.takakura.com/Kyoto_data, 2015.

[53] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in *Proc. of MilCIS*, 2015.

[54] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of ICISSP*, 2018.

[55] Massachussets Institute of Technology, "1998 DARPA Intrusion Detection Evaluation Dataset," https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset, 1998.

[56] B. Sangster, T. J. O'Connor, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, "Toward instrumenting network warfare competitions to generate labeled datasets," in *Proc. of CSET*, 2009.

[57] Canadian Institute for Cybersecurity, "Intrusion Detection Evaluation Dataset (ISCXIDS2012)," https://www.unb.ca/cic/datasets/ids.html, 2012.

[58] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[59] Canadian Institute for Cybersecurity, "DoS dataset (CIC DoS dataset 2017)," https://www.unb.ca/cic/datasets/dos-dataset.html, 2017.

[60] ALDAPA, "Gure-Kddcup dataset," http://www.sc.ehu.es/acwaldap/gureKddcup, 2008.

[61] I. Perona, I. Gurrutxaga, O. Arbelaitz, J. I. Martín, J. Muguerza, and J. M. Pérez, "Service-independent payload analysis to improve intrusion detection in network traffic," in *Proc. of AusDM*, 2008.

[62] National Security Agency, "Cyber Defense Exercise (CDX)," https://apps.nsa.gov/iaarchive/programs/cyber-defense-exercise/index.cfm, 2001.

[63] I. Homoliak, M. Barabas, P. Chmelar, M. Drozd, and P. Hanacek, "ASNM: Advanced security network metrics for attack vector description," in *Proc. of SAM*, 2013.

[64] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 29–38, 2006.

[65] 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 *Proc. of PST*, 2011, pp. 174–180.

[66] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. of CoNEXT*, 2010.

[67] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, 2014.

[68] University of Massachusetts Amherst - Laboratory for Advanced Software Systems, "UMassTraceRepository," http://traces.cs.umass.edu/index.php/Network/Network, 2018.

[69] A. Sperotto, R. Sadre, F. van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *IP Operations and Management*, ser. Lect. Notes in Comput. Sc. Springer, 2009, pp. 39–50.

[70] Center for Applied Internet Data Analysis, "Data Collection, Curation and Sharing," https://www.caida.org/data/, 2018.

[71] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. of WCNC)*, 2013, pp. 4487–4492.

[72] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proc. of AISTATS*, 2009, pp. 448–455.

[73] I. Goodfellow, Y. Bengio, and A. Courville, "Autoencoders," in *Deep Learning*. MIT Press, 2016.

[74] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[75] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[76] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NIPS*, 2014, p. 2672–2680.

[77] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, p. 1771–1800, 2002.

[78] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, 2006.

[79] M. A. Salama, H. Eid, R. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," *Adv. Intell. Soft Comput.*, vol. 96, pp. 295–302, 2011.

[80] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using Deep Belief Network and probabilistic neural network," in *Proc. of CSE*, vol. 1, 2017, pp. 639–642.

[81] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on Deep Belief Networks," in *Proc. of CBD*, 2014, pp. 247–252.

[82] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using Deep Belief Networks," in *Proc. of NAECON*, 2015, pp. 339–344.

[83] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on Deep Learning," in *Proc. of ICMLA*, 2016, pp. 195–200.

[84] E. R. Merino, F. M. Castrillejo, J. D. Pin, and D. B. Prats, "Weighted contrastive divergence," *CoRR*, vol. abs/1801.02567, 2018.

[85] NVIDIA, "CUDA," https://developer.nvidia.com/cuda-zone, 2020.

[86] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using Auto-Encoder bottleneck features," in *Proc. of IKT*, 2015, pp. 1–5.

[87] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. of IJCNN*, 2017, pp. 3854–3861.

[88] V. L. Cao, M. Nicolau, and J. McDermott, "A Hybrid Autoencoder and density estimation model for anomaly detection," in *Proc. of PPSN*, 2016, pp. 717–726.

[89] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. ki Cho, and H. Chen, "Deep Autoencoding Gaussian Mixture Model for unsupervised anomaly detection," in *Proc. of ICLR*, 2018.

[90] A. Y. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning approach for network intrusion detection system," in *Proc. of BICT*, 2015.

[91] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, p. 3371–3408, 2010.

[92] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated Convolutional Autoencoders," *Secur. Commun. Netw.*, vol. 2017, pp. 1–10, 2017.

[93] Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning based DDoS detection system in software-defined networking (sdn)," *EAI Endorsed Trans. on Security and Safety*, vol. 4, no. 12, 2017.

[94] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.

[95] F. Farahnakian and J. Heikkonen, "A Deep Auto-Encoder based approach for intrusion detection system," in *Proc. of ICACT*, 2018.

[96] L. R. Parker, P. D. Yoo, T. A. Asyhari, L. Chermak, Y. Jhi, and K. Taha, "DEMISe: Interpretable Deep extraction and mutual information selection techniques for IoT intrusion detection," in *Proc. of ARES*, 2019, pp. 98:1–98:10.

[97] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of ICLR*, 2014.

[98] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "GEE: A gradient-based explainable Variational Autoencoder for network anomaly detection," in *Proc. of CNS*, 2019, pp. 91–99.

[99] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs," *Computers & Security*, vol. 73, pp. 411–424, 2018.

[100] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning latent distribution for distinguishing network traffic in intrusion detection system," in *Proc. of ICC*, 2019, pp. 1–6.

[101] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, 2017.

[102] A. Genovese, V. Piuri, K. N. Plataniotis, and F. Scotti, "PalmNet: Gabor-PCA Convolutional Networks for touchless palmprint recognition," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, 2019.

[103] R. Donida Labati, A. Genovese, E. Muñoz, V. Piuri, and F. Scotti, "A novel pore extraction method for heterogeneous fingerprint images using Convolutional Neural Networks," *Pattern Recognit. Lett.*, vol. 113, no. 1, pp. 58–66, 2018.

[104] A. Genovese, V. Piuri, F. Scotti, and S. Vishwakarma, "Touchless palmprint and finger texture recognition: A Deep Learning fusion approach," in *Proc. of CIVEMSA*, 2019.

[105] R. Donida Labati, A. Genovese, E. Muñoz, V. Piuri, and F. Scotti, "Applications of computational intelligence in industrial and environmental scenarios," in *Learning Systems: from Theory to Practice*. Springer, 2018, vol. 756, pp. 29–46.

[106] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using Convolutional Neural Networks for representation learning," in *Neural Information Processing*. Springer, 2017, pp. 858–866.

[107] "One-hot encoding," https://www.sciencedirect.com/topics/computer-science/one-hot-encoding, 2020.

[108] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with Deep Convolutional Neural Networks," in *Proc. of NTMS*, 2018, pp. 1–5.

[109] T. Kim, S. C. Suh, H. Kim, J. Kim, and J. Kim, "An encoding technique for CNN-based network anomaly detection," in *Proc. of Big Data*, 2018, pp. 2960–2965.

[110] R. Blanco, P. Malagón, J. J. Cilla, and J. M. Moya, "Multiclass network attack classifier using CNN tuned with genetic algorithms," in *Proc. of PATMOS*, 2018, pp. 177–182.

[111] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of CVPR*, 2016, pp. 770–778.

[112] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of CVPR*, 2015.

[113] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of ICLR*, 2015.

[114] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using Convolutional Neural Networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[115] U. Çekmez, Z. Erdem, A. G. Yavuz, O. K. Sahingoz, and A. Buldu, "Network anomaly detection with Deep Learning," in *Proc. of SIU*, 2018, pp. 1–4.

[116] M. Ito and H. Iyatomi, "Web application firewall using character-level Convolutional Neural Network," in *Proc. of CSPA*, 2018, pp. 103–106.

[117] S. Z. Lin, Y. Shi, and Z. Xue, "Character-level intrusion detection based on Convolutional Neural Networks," in *Proc. of IJCNN*, 2018, pp. 1–8.

[118] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and Convolutional Neural Networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[119] G. Feng, B. Li, M. Yang, and Z. Yan, "V-CNN: Data visualizing based Convolutional Neural Network," in *Proc. of ICSPCC*, 2018, pp. 1–6.

[120] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *CoRR*, vol. abs/1802.10135, 2018.

[121] S.-N. Nguyen, V.-Q. Nguyen, J. Choi, and K. Kim, "Design and implementation of intrusion detection system using Convolutional Neural Network for DoS detection," in *Proc. of ICMLSC*, 2018.

[122] S. Park, M. Kim, and S. Lee, "Anomaly detection for HTTP using Convolutional Autoencoders," *IEEE Access*, vol. 6, 2018.

[123] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, M. Steinbrecher, F. Klawonn, and C. Moewes, *Computational Intelligence: A Methodological Introduction*, 2nd ed. Springer, 2016.

[124] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proc. of MLCS*, 2018, pp. 1–8.

[125] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," *ArXiv*, vol. abs/1611.01726, 2016.

[126] F. Jiang, Y. Fu, B. B. Gupta, F. Lou, S. Rho, F. Meng, and Z. Tian, "Deep Learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2018.

[127] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying Convolutional Neural Network for network intrusion detection," in *Proc. of ICACCI*, 2017, pp. 1222–1228.

[128] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using Deep Neural Networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[129] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on Deep Hierarchical Network and original flow data," *IEEE Access*, vol. 7, pp. 37 004–37 016, 2019.

[130] M. Elbayad, L. Besacier, and J. Verbeek, "Pervasive attention: 2D Convolutional Neural Networks for sequence-to-sequence prediction," *CoRR*, vol. abs/1808.03867, 2018.

[131] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[132] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[133] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 10:1–10:43, 2019.

[134] A. Genovese, V. Piuri, and F. Scotti, "Towards explainable face aging with Generative Adversarial Networks," in *Proc. of ICIP*, 2019.

[135] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with Generative Adversarial Networks to guide marker discovery," *CoRR*, vol. abs/1703.05921, 2017.

[136] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-based anomaly detection," *ArXiv*, vol. abs/1802.06222, 2018.

[137] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proc. of ICISS)*, 2016, pp. 1–6.

[138] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Proc. of CVPR*, 2011, pp. 1521–1528.

[139] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars, "A deeper look at dataset bias," in *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 37–55.

[140] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 262–294, 2000.

[141] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, pp. 521–530, 2012.

[142] R. F. Fouladi, T. Seifpoor, and E. Anarim, "Frequency characteristics of DoS and DDoS attacks," in *Proc. of SIU*, 2013, pp. 1–4.