A Survey on Data-driven Network Intrusion Detection

DYLAN CHOU, Carnegie Mellon University
MENG JIANG, University of Notre Dame

Data-driven network intrusion detection (NID) has a tendency towards minority attack classes compared to normal traffic. Many datasets are collected in simulated environments rather than real-world networks. These challenges undermine the performance of intrusion detection machine learning models by fitting machine learning models to unrepresentative "sandbox" datasets. This survey presents a taxonomy with eight main challenges and explores common datasets from 1999 to 2020. Trends are analyzed on the challenges in the past decade and future directions are proposed on expanding NID into cloud-based environments, devising scalable models for large network data, and creating labeled datasets collected in real-world networks.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Networks** → **Network security**; • **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; • **Computing methodologies** → **Machine learning;**

Additional Key Words and Phrases: Network intrusion detection, data mining, machine learning

ACM Reference format:
Dylan Chou and Meng Jiang. 2021. A Survey on Data-driven Network Intrusion Detection. *ACM Comput. Surv.* 54, 9, Article 182 (October 2021), 36 pages.
https://doi.org/10.1145/3472753

1 INTRODUCTION

Network intrusion detection (NID) monitors a network for malicious activity or policy violations [124, 129]. During the past two decades, data-driven methods have been developed and deployed for NID systems [42, 162], most of which are machine learning models such as Naïve Bayes [135], Random Forests [48, 192], Adaboost [74], and Deep Neural Networks [78, 158]. A review paper in 2009 summarized the NID systems that were supported by *anomaly detection* algorithms [61, 96]. In this survey, we present a broader view of *data-driven* NID, which includes related work from the past 10 years, and present a taxonomy of challenges and methods in data-driven NID research.

1.1 Background

Since the advent of computer networks, e-commerce, and web services, there has been a greater need for cyber-security and countermeasures toward network attacks. There was an interest in

This research was supported by NSF Grant IIS-1849816.
Authors' addresses: D. Chou, Carnegie Mellon University, Pittsburgh, PA, 15213; email: dvchou@andrew.cmu.edu; M. Jiang, University of Notre Dame, Notre Dame, Indiana, 46556; email: mjiang2@nd.edu.

0360-0300/2021/10-ART182 $15.00
https://doi.org/10.1145/3472753

intrusion detection in 1994, where intrusion detection was known to be a *retrofit* way to provide a sense of security when identifying unauthorized use, misuse, or abuse of computer systems [124]. The concept of intrusion detection later became contextualized in cyber-security systems. The term "intrusion detection systems" describes the extraction of information from one or multiple computers in a network that identifies attacks from external sources, but also misuse of resources in the network from internal sources [25].

Intrusion detection systems can be broadly categorized as either being host-based intrusion detection or network intrusion detection. Host-based intrusion detection monitors host-specific actions such as system files being accessed or applications in use. A network intrusion detection system is focused on monitoring data flow between computers and "sniffs" for anomalous network traffic between different computers in the network [126].

There are two general behaviors in a network: normal and anomalous. Normal network behavior follows a specific criteria in terms of the traffic volume, applications on the network, and types of data exchanged. Network anomalies fall into two general categories of network failures such as network congestion or file servers being down and network security attacks such as DDoS and other attacks that are conducted by a malicious agent [171].

Network intrusion detection systems aim to distinguish the norm from security-related anomalies and detect attacks on computer networks. Network intrusion detection methods can be anomaly-based that identify malicious activity that departs from normal-defined behavior on a network or signature-based that identifies known attacks based on pattern matching. Because signature-based detection relies on seen patterns, it is not as effective in detecting novel attacks, or zero-day attacks, so anomaly detection is often used to detect novel attacks.

1.2 Past Surveys

Among the network intrusion detection surveys gleaned from the past decade, as presented in Table 1, many have constructed taxonomies along with problem-solution frameworks for cloud-computing platforms. Jeong et al. [79] addressed the anomaly teletraffic intrusion detection systems in Hadoop-based platforms where there is a heavy focus on the methodology of statistical, machine learning, and knowledge-based models. Different attributes of big data—storage volume, velocity, variety, intrusion detection system, and cost—are associated with problems and technical solutions specific to Hadoop-based platforms. A new platform was proposed for anomaly teletraffic intrusion detection systems on Hadoop. Modi et al. [119] followed a high-level introduction of intrusion detection to cloud-based systems—a common solution to these intrusions being firewalls—and identified differences between signature and anomaly-based detection. Keegan et al. [88] inspected network intrusion detection datasets, approaches, cloud environments, algorithms, and advantages and disadvantages among the literature.

Other authors primarily heeded the network intrusion detection datasets rather than its methods. Ring et al. [147] examined packet-based, flow-based data along with host log files. Data recording environments were compared from the literature and a multitude of datasets, including some data repositories found on the Internet, were discussed along with their drawbacks. Ring presented a comprehensive overview of 34 datasets, their drawbacks, and how they may be related if one dataset was built off of another. Davis and Clark [39] studied intrusion detection features derived from network traffic along with data preprocessing methods including clustering, filtering packets by high anomaly score or extracting subsets during traffic payload analysis, tracing TCP sessions, statistical features per connection, and create separate dataset.

Some papers were method-specific, as Resende and Drummond [143] provided a comprehensive review of random forest-based network intrusion detection. Resende and Drummond presented both a high-level overview of random trees and its components: decision trees. Datasets and

Table 1. Summary of Past Surveys: Our Survey Presents Past Surveys That Define Overviews of Frameworks and Methodologies as Precedents for Our Paper, Which Looks Deeper into Research Trends over Time and Possible Solutions to Research Challenges on the Rise

	Title	Focus	Topics
Davis and Clark. 2011. [39]	**Data preprocessing** for anomaly based network intrusion detection: A review	Data preprocessing	Relevant features construction using targeted content parsing and deeper network packet inspection
Jeong et al. 2012. [79]	Anomaly teletraffic intrusion detection systems on **Hadoop-based Platforms**: A survey of problems and solutions	Framework	Hadoop and big data platforms for speed, storage volume, and cost-efficiency
Poston. 2012. [139]	A brief taxonomy of intrusion detection **strategies**	Strategies	Taxonomy of traditional network intrusion detection
Modi et al. 2013. [119]	A survey of intrusion detection techniques in **Cloud**	Framework	Incorporating IDS on host system and virtual machines
Keegan et al. 2016. [88]	A survey of **cloud-based** network intrusion detection analysis	Framework	Integrating machine learning algorithms and MapReduce to cloud computing environments
Resende and Drummond. 2018. [143]	A survey of **random forest**-based methods for intrusion detection systems	Machine learning strategy	Application of random forest methods over time
Ring et al. 2019. [147]	A survey of network-based intrusion detection **data sets**	Data collection	Categorization of 34 public datasets

common evaluation metrics were reviewed and the authors concluded that, in future work, random forests will be used more on unbalanced data and on dynamic data due to its ability to adapt to incremental learning problems.

General overviews of network intrusion detection definitions and infrastructures along with taxonomies to classify different types of intrusion detection systems were also made. Poston's taxonomy [139] covered high-level definitions of the types of intrusion detection and the types of analysis that can be done on host-based and network-based intrusion detection. However, the taxonomy is fairly general and the paper does not address future directions to intrusion detection research. Other papers looked to a specific result after observing the challenges in each paper and comparing their machine learning methods, as Buczak and Guven [23] organized their review based on a machine learning method, presented the papers that use that method, the data it used, the cyber approach (misuse or anomaly), and the number of times the paper was cited. Mitchell and Chen [118] broke down the classification of intrusion detection by system, collection process, techniques, models, analysis, and response. Many visuals are dedicated to their four defined types of intrusion detection: anomaly-based, signature-based, specification-based, and reputation-based. Most and least studied IDS techniques are analyzed and future direction of research in repurposing existing work on wireless intrusion detection applications, multitrust with intrusion detection, specification-based detection for cyber-physical systems, and others. Ahmed et al. [4] analyzed four main categories of anomaly-based detection: clustering, classification, statistical, and information theory. Each category was evaluated based on the computational complexity among approaches of that type, the most significant network attacks, and what the output is in each technique.

1.3 Our Contributions

There have been survey papers as broad as scanning over all network anomaly detection methods and as specific as cloud-based intrusion systems. Past surveys focused on the foundational knowledge of network intrusion detection frameworks such as TCP connection features or virtual machine layers in hypervisor/host systems. Surveys have looked into overviews of datasets, or

comparisons between specific machine learning methods, all while reflecting on past literature. Many authors present previous work with charts comparing different papers and discussing challenges with cloud computing, growing data, and other open issues. Challenges have been addressed in many of these surveys, but there is a lack of solutions presented under future direction. Mitchell and Chen [118] examined the most and least studied areas in wireless network intrusion detection to propose future research areas. Past surveys focus less on the most researched areas in data-driven NIDS over a long span of time, so we summarize trending data-driven NIDS research topics in a stacked bar chart over the past decade (see Section 5). In turn, we draw conclusions on plausible future directions in areas that researchers have not thoroughly investigated recently or research areas that they have abandoned.

1.4 Overview

In Section 2, the history of data processing, cloud computing, the lack of specific network attack types, and general big data processing techniques are examined. Section 3 covers common datasets from DARPA 1998 [94] to as recent as LITNET 2020 [132] along with their statistics in terms of how network attacks are distributed and how unbalanced the datasets are based on entropy. Section 4 addresses the high-level organization of the taxonomy and the details of each challenge and corresponding solutions/methods. Section 5 discusses the trends based on the articles collected that form the taxonomy and areas to look further into. Section 6 presents conclusions from the literature survey and taxonomy of data-driven network intrusion detection and reinforces future directions that researchers can look into.

2 DATA PROCESSING

The key purpose of anomaly detection systems is to separate anomalies from normal behavior. In computer networks, a network anomaly refers to circumstances where network operations deviate from normal network behavior [171]. Anomaly-based network intrusion detection methods are important to identify novel intrusion attacks.

Data reduction is done to remove large amounts of data to improve efficiency and reduce computational overhead. In network traffic, packets are exchanged and TCP connections are open for the exchanges to be carried out. Because so many packets are sent and received in a typical network, extracting only the first few packets of a TCP connection was done by Chen et al. [27] to mitigate effects of large packet data. Beyond extraction of data during its collection, other authors selected specific network features based on their importance. Tan et al. [166] aimed to address the challenge of the heavy computation associated with anomaly intrusion detection systems using **linear discrimination analysis (LDA)** and distance difference maps to select the most significant features. LDA finds an optimal projection matrix to project higher dimensional features to lower dimensions. This feature reduction method was done on payload-based anomaly intrusion detection. In 2013, Zhang and Wang [190] applied a simpler feature selection method that underwent a sequential search and sifted through the features in the feature domain, where a feature was added if the accuracy from the Bayesian network detection model lowered after removing a feature.

Most recently, attention has been directed towards new technologies in the cloud and newer optimizations with computation aside from parallelism. Cloud computing services allow for processing of large datasets, and a popular engine for big data processing is Apache Spark. Gupta and Kulariya [65] presented a framework where correlation-based and chi-square feature selection were applied to obtain the most important feature set and Logistic regression, **Support Vector Machines (SVMs)**, Random forest, Gradient Boosted Decision trees, and Naive Bayes were used for network intrusion classification from the MLlib library in Apache Spark. In 2019, Hajimirzaei

Table 2. Vertical Comparisons of Common Datasets

Dataset	Duration	Traffic Type	Method	#IPs	#Instances
KDDCup1999 [40]	N/A	Synthetic	Tcpdump	N/A	4,898,430
NSL-KDD 2009 [53]	7 weeks	Synthetic	N/A	N/A	125,973/22,544
UNSW NB15 IDS [131]	15–16 hours	Synthetic	Tcpdump/IXIA PerfectStorm	45	2,540,044
UGR'16 [51]	96 days	Real	Netflow	600M	16.9M
CIDDS'17 [130]	4 weeks	Emulated	Netflow	26	32M
CICDS'17 [54]	5 days	B profile sys.	User behavior	21	2,830,743
CSE-CIC-IDS2018 [55]	17 days	B/M profile system	CICFlowMeter	500	4,525,399
LITNET-2020 [132]	10 months	Real	Flow traces	7,394,481	39,603,674
MAWILab [95]	15 min/d	Real	Sample point collection	N/A	N/A

and Navimipour [67] used a combination of a **multilayer perceptron (MLP)** network, an **artificial bee colony (ABC)** algorithm, and a fuzzy clustering algorithm to detect network intrusions. The MLP network weights are initialized and the ABC algorithm, an optimization procedure simulating the foraging behavior in bees, updates the initial weights of the MLP until the weights of the model entail acceptable fitness. Here, fitness is defined as the error of the model on unseen traffic data. The environment was simulated in CloudSim and exemplifies an application of a novel integration of machine learning techniques into the cloud, different from Gupta and Kulariya's work that did not employ cloud-computing machine learning.

3 COMMON PUBLIC DATASETS AND REPRODUCIBILITY

Table 2 compares common datasets which are open-source and improvements of past datasets such as being one of the first publicly available **intrusion detection system (IDS)** datasets or being updated with newer attacks that are commonly used. Dataset inclusion was based off those that were most common among highly cited data-driven NID papers. Other datasets were not included, as some were not publicly available or did not make notable improvements on past datasets.

Table 3 presents the papers that used each dataset and the reproducibility of the methods that were applied to the datasets. These papers were included on the basis of the most cited papers over the past decade that used these datasets. The overarching criteria was that these papers had to be relevant to network intrusion detection.

3.1 Dataset Description

3.1.1 KDD Cup 1999. The KDD Cup 1999 was a version of the 1998 DARPA Intrusion Detection Evaluation Program that was collected by MIT Lincoln Labs in their packet traces and is one of the most widely used datasets for network intrusion detection [40]. Lincoln Labs acquired roughly nine weeks of raw tcp dump data from a **local area network (LAN)**, that simulates a similar environment as an air force LAN. The attacks fall into the four main categories: denial-of-service such as a syn-flood, unauthorized access to a remote machine (R2L), unauthorized access to a local superuser (U2R), and probing such as port scanning [40]. Although the KDD Cup 1999 dataset is considered relatively large in that it contains 41 features and over 4.8 million rows of data, it runs into the issue of duplicates between training and testing data [160]. The data is missing some important features such as IP addresses although there are basic TCP attributes provided such as the source and destination bytes. Although the KDD Cup 1999 dataset does capture a good number of attacks, the data was collected on a synthetic network. In general, the data collected is outdated, because it was made nearly two decades ago and has bias due to synthetic generation [41].

Table 3. Datasets and Papers That Used the Datasets for Evaluation

Dataset	Research works that used the dataset for evaluation
KDDCup1999	• [26, 27, 31, 34, 50, 57, 65, 72, 91, 92, 103, 112, 114, 148, 152, 158, 173, 176, 179, 181, 183, 187–189] ★ [7, 33, 44, 49, 64, 70, 89, 90, 98, 100, 101, 117, 120, 122, 141, 159, 164, 172, 182, 185]
NSL-KDD 2009	• [36, 45, 59, 73, 80, 81, 102, 105, 133, 134, 137, 150, 158, 170, 176, 181, 202] ★ [60, 75, 93, 125, 142, 149, 163, 174, 184, 190, 197, 198] ▷ [69]
UNSW NB15 IDS	• [16, 20, 73, 161] ★ [81, 89, 90, 122, 154, 174, 182, 184, 191] ▷ [69]
UGR'16	• [109] ▷ [110]
CIDDS-001	• [2, 144, 148]
CICIDS'17	• [6, 11, 29, 45, 157] ★ [63, 140, 191] ▷ [46, 69, 180, 194, 195, 201]
CSE-CIC-IDS2018	• [91]
LITNET-2020	• [38]
MAWILab	▷ [201]

"•" for those that are general methods or frameworks.
"★" for those that pseudo code and implementation details are available.
"▷" for those that have documented code associated with the paper.

3.1.2 NSL-KDD 2009. The NSL-KDD 2009 dataset was made to resolve issues of possible biases in duplicate data between training and testing datasets from the KDD Cup 1999 [53]. The Canadian Institute for Cybersecurity and University of New Brunswick were involved in collecting the dataset. However, NSL-KDD removed some redundant, more frequent records in the training set that were from the KDD Cup 1999 dataset, which can still be important. In turn, this may lead to further biases given that the data from the raw TCP dump should still be kept. An underlying issue with the NSL-KDD dataset is that it still contains data from a network dating back as early as 1998's DARPA dataset.

3.1.3 UNSW NB15 IDS. The UNSW NB15 Intrusion Detection System dataset contains source files in the formats of pcap, BRO, Argus, and CSV along with reports by Dr. Nour Moustafa [131]. The dataset was created with an IXIA traffic generator that had TCP connections to a total of three servers. Two of these servers were connected to a router that had a TCP dump and three clients, where the TCP dump resulted in pcap files. The third server was connected to a router with three clients as well. The two routers that the first two servers and the third server were connected to were separated by a firewall. An issue with the UNSW NB15 dataset is again with the realness in its data, because the dataset was created from a traffic generator.

3.1.4 UGR'16. The UGR'16 dataset was collected from several netflow v9 collectors in the network of a Spanish ISP by researchers from University of Granada in Spain [51]. The data is split into a calibration and training set, where long-term evolution and periodicity in data is a major advantage over previous datasets. However, a major issue is that most of the network traffic is labeled

as "background," which may either be anomalous or benign. Also, there is a mix of synthetically generated network attacks along with real-world network traffic, which is not of the same quality if none of the traffic was simulated. The dataset was labeled based on the logs from their honeypot system in their setup.

3.1.5 CIDDS-001. The CIDDS-001 dataset was collected in 2017 by four researchers [146], two PhD students, and two professors, who are affiliated with the Coburg University of Applied Sciences in Germany [130]. The data was part of the project WISENT, funded by the Bavarian Ministry for Economic affairs. The intention of the dataset was to be used as an evaluation dataset for anomaly-based intrusion detection systems. The dataset is labelled and flow-based, where a small business environment was emulated on OpenStack. For the infrastructure, on the Internet, there are three attackers and an external server that has a firewall separating it from a server, where there are three layers: developer, office, and management. There are four servers that are in the OpenStack environment containing the three subnet layers. Generation of DoS, Brute Force, and Port Scanning occurred in the network. The first label attribute is traffic class: normal, attacker, victim, suspicious, and unknown. The second label attribute is attack type and the third being an attack ID. Because the external server emulates a real network environment, the CIDDS-001 dataset is primarily used for benchmarking. There are only three types of attacks, which unveils a lack of diversity in the data [145].

3.1.6 CICIDS'17. The CICIDS dataset was collected under the Canadian Institute of Cybersecurity as well and University of New Brunswick [54]. The generation of network traffic came from a proposed B-profile system where abstract behaviors were derived for 25 users based on HTTP, HTTPS, FTP, SSH, and email protocols. With regards to the victim and attacker network information, there was a firewall against the IPs 205.174.165.80 and 172.16.0.1 and a DNS server at 192.168.10.3. The attackers network comprises two IPs: Kali: 205.174.165.73, Win: 205.174.165.69. The victim network is composed of 2 Web servers 16 Public, 6 Ubuntu servers, 5 Windows servers, and a MAC server. The data collection occurred over the course of five days where Monday was benign activity, Tuesday was brute force, Wednesday was DoS, and Thursday was web attacks where the afternoon saw Botnet, Port Scan, and a DDoS LOIT.

3.1.7 CSE-CIC-IDS2018. The CSE-CIC-IDS2018 dataset is a collaborative project between the **Communications Security Establishment (CSE)** and the **Canadian Institute of Cybersecurity (CIC)** [55]. A notion of profiles is adopted to generate data systematically. First is the B profile that captures behavior in users using machine and statistical learning techniques. M-profiles are human users or automated agents who may examine network scenarios. With the environment supported in AWS, the network topology includes an attack network of 50 machines, 5 departments holding 100 machines each and a server with 30 machines.

3.1.8 LITNET-2020. LITNET is a new annotated benchmark dataset where data was collected by four professors, a PhD student, and two students at the **Kaunas University of Technology (KTU)** [132]. The infrastructure of the network is composed of nodes with communication lines connecting them. The LITNET topology consists of senders and receivers, netflow senders (Cisco routers), and a netflow server. The netflow exporters were in four cities in Lithuania, Vilnius Gediminas Technical University, and two KTU university nodes. The dataset contains real network attacks in Lithuanian-wide network with servers in four geographic locations within the country.

3.1.9 MAWILab. MAWILab is a database containing the dataset from the MAWI archive that records network traffic data between two endpoints [95]: one in Japan and another in the U.S. MAWILab's dataset has been contributed to since 2010 [52] and records 15 minutes of network

traces each day. Labels of network traffic are generated from anomaly classifiers based on port numbers, TCP flags, and ICMP codes along with a taxonomy of traffic anomalies based on packets headers and connection patterns [116]. The graph on MAWILab's website divides the type of traffic over the course of 13 years based on byte and packet ratios. HTTP traffic used to be very common from 2007 to 2017, but sharply decreased at the end of 2017. Port Scanning is uncommon, where "multiple points" was the second most dominant traffic type from 2007 to 2017. A spike in **denial of service (DoS)** data was collected between 2011 and 2012. Currently, the most common type of traffic is multi points, then http, then IPV6 tunneling and alpha flow by byte and packet ratio. The number of anomalies from 2007 to 2020 ranged roughly between 100 to 200 at any time. Outliers are as low as 50 anomalies and as high as 500 anomalies daily. Since the network traffic has been between over the same link and two endpoints since 2007, MAWILab's network may not be as similar to most other networks used now. In addition, the labels fall into four broad categories: anomalous, suspicious, notice, and benign. The labels are dependent on the anomaly classifiers, so there may be misclassified traffic.

3.2 Reproducibility on Datasets

The code repository by He and others [69] implemented a LSTM neural network, **Multimodal Deep Auto Encoder (MDAE)**, **Autoencoder (AE)**, Gaussian **Restricted Boltzmann Machine (RBM)**, and **Multimodal-sequential approach with deep hierarchical progressive network (MS-DHPN)** in Python that were trained on the network intrusion datasets NSL-KDD 2009, UNSW NB15 IDS, and CICIDS 2017. The repository contains separate files to load data and run the different models to reproduce results. However, the train and test dataset paths are defined specific to the author's environment along with the naming schemes that they chose. The paper does not specify how their renamed CSV file paths were derived from the original CSVS in each of the datasets. The code is present, but not reproducible without consultation from the authors.

The repository by Roberto and others [110] implemented **logistic regression (LR), random forest (RF)**, and SVM in Python on the UGR 2016 dataset. The AUC, precision, recall, F1-score, and support metrics are reproducible for these models, where the model, number of iterations, number of cross-validation folds, and configuration file path are specified. However, only the SVM model results were included in the code repository. There was scaling of the data prior to training and testing the models when running the main Python file, but their implementations of the partial least squares regressions described in the paper were not provided in the code. The code is partially reproducible for some of the models' performances, but not all.

The code repository made public by Faker and others [46] implemented all their preprocessing and machine learning models in IPython on the CICIDS 2017 network intrusion dataset. However, no details in the README file points to Python package versions, resulting in the user inferring what versions were used at the time the code was run by the authors. The tensorflow package should not be version 2 for the code to run, which restricts the versions of Python to 3.7 or earlier. The package `distkeras` contains missing methods in its `SequentialWorker` class such as `add_history` and `training_history`, so a follow-up on package versions and fixes with the authors is needed.

The repository common to Zhang and others' 2019 papers on hierarchical clustering [195] and parallel cross convolutional neural network [194] implemented an LSTM-CNN model in Python on the CICIDS 2017 intrusion detection dataset. The network flow data comes from *Baidu Netdisc Storage*, which requires the registration of a Baidu cloud drive account. The data

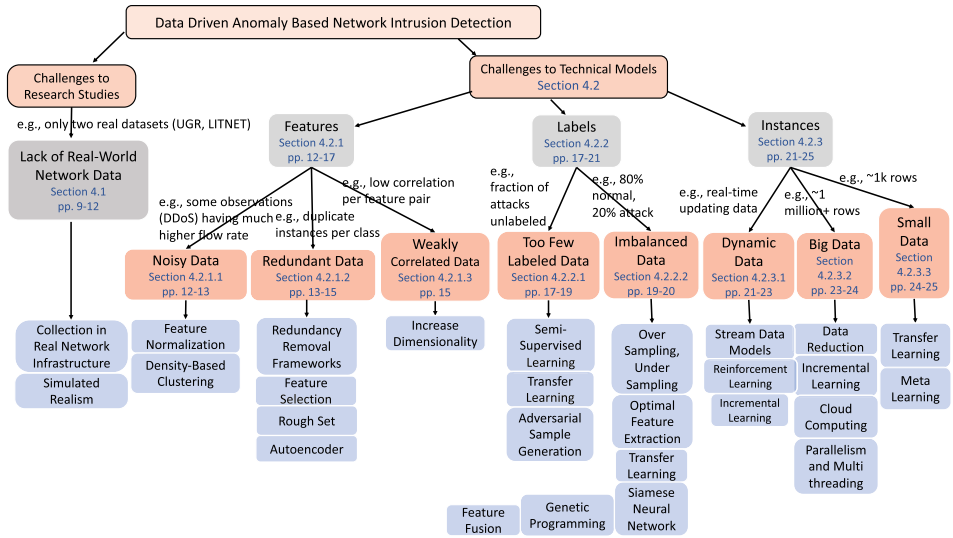


Fig. 1. Hierarchical chart of categorized high-level challenges and recent methods to resolve them. The section and page numbers are included in the boxes so readers can easily skip around different sections.

download process can be time-consuming. Also, there may be version issues, depending on the Python version, but `python3.6 -m venv env` can create a Python 3.6 virtual environment that supports tensorflow before version 2: `pip install tensorflow==1.5`. The data used in the paper is online and models are reproducible.

The code repository released by Zhong and others [201] contains a demo that consolidates the code for parsing the network flow packets, extracting features, training the deep belief network autoencoder and LSTM, and plotting the anomaly scores for the indices of the network packets. The model was tested on the MAWILab and CICIDS 2017 intrusion datasets. The code is able to completely recreate the results from the paper.

The code repository provided by Xu and others [180] is not reproducible, as the *code and datasets link* in the paper leads to a site with an insecure connection and a privacy error. Further consultation with the authors would need to take place.

4 A TAXONOMY: CHALLENGES AND METHODS

Figure 1 presents a hierarchical chart of categorized high-level challenges and recent methods to resolve them. This section will discuss the challenges and introduce the methods in details. The papers in the taxonomy were included because they are relatively highly cited by other researchers and can highlight a moment in the progression of a research area, such as big data or dynamic data.

4.1 Lack of Real-world Network Data

Challenge. When network traffic data was initially being collected, even as early as the KDD Cup dataset from 1999, attacks were outdated and not compatible to attacks done in the real-world. Because using real-world networks to collect network traffic was costly, researchers looked to simulating realistic networks with synthetic data generation or a simulated virtual network as an alternative. Initially, honeypots were used as a means of simulating a virtual network environment

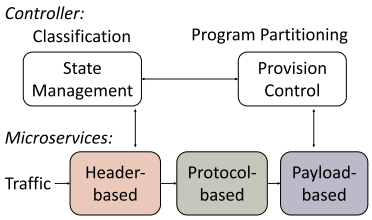
to attract attackers and gather traffic data. Honeypots are security resources that are meant to be misused by malicious attackers, where such attacks would be recorded in databases. They consist of a decoy, or an information source, and a security program that provides attack monitoring and detection [47]. These mechanisms can be used to collect network intrusion data with simulated realism that run in a virtual machine [43, 87], containing possibly more than one honeypot to resemble a distributed honeypot system to simulate a distributed network more accurately [108]. The main drawback with honeypots is their limited vision on network attacks. A honeypot is only aware of an attack if it is attacked, but would not be able to detect attacks directed at other systems. The use of TCP dumps in IXIA traffic generation also plays a large role in simulating realistic network intrusion data by synthetically generating data, which Moustafa and Slay [123] have done to create the UNSW-NB15 dataset. They generated data with an IXIA traffic generator, then collected pcap files extracted from a tcpdump. This synthetic data generation was improved upon in 2017 by Haider et al. with the generation of network traffic via IXIA Perfect Storm and collection of the host's network logs during the simulation. This was better than the UNSW-NB15 dataset, because UNSW-NB15 lacked the information of normal and synthetic data that came from the operating system's log files. Haider et al. also verified the realism of their dataset through the Sugeno fuzzy inference engine [66]. Architectures of main approaches to the creation of real-world network data are illustrated in Figure 2.

Collection in public network infrastructure. In the past couple of years, researchers have looked to collect network traffic data in a cloud environment due to the growing usage of cloud computing platforms such as Amazon Web Services and Google Cloud. A mixture of virtualization and cloud intrusion detection using hypervisors have been implemented as well, which can resolve the issue of small datasets by aggrandizing network traffic data. In 2018, Hongda et al. [99] combined network virtualization with software-defined networks to handle attack traffic. Their **virtual network intrusion detection system (vNIDS)** employed static program analysis to determine the detection states to share. The prototype of vNIDS was done in CloudLab for flexibility with processing capacity and placement location. In the past year, Aldribi et al. [9] acknowledged the new challenging terrain that cloud computing provides for attackers. In turn, they implemented a new hypervisor-based cloud network intrusion detection system using multivariate statistical change analytics to detect anomalies. Alongside further research into generating network traffic data in the cloud, the realism in past datasets was called into question because of their outdated attacks and synthetic traffic generation. A proposed solution involved generating real-world network through gathering network traffic from a university network such as the Lithuanian Research and Education Network [38] or a real virtual network of a tier-3 ISP done in the UGR'16 dataset [109].

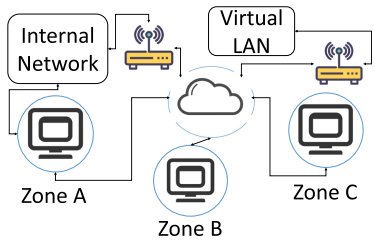
4.1.1 Ability to Transfer.

Challenge. There are no guarantees on how well network anomaly detection systems can perform in networks that consumers or corporations actively use. In fact, Allix and others [10] state that it is impossible of obtain a dataset that reflects current real-world networks due to the rarity and secrecy of network intrusions. And it would not be practical to train a network anomaly detection system in a real-network setting due to the consequences of undetected intrusions on the security of the network. However, researchers can transfer what an intrusion detection system has learned in a simulated environment to an in-use network if the environment mimics traffic patterns of networks currently in use.

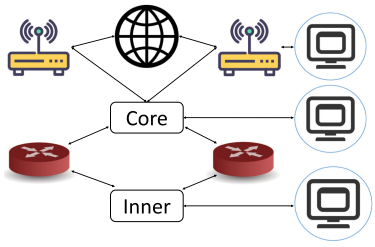
IDS Methods. There are a few recent paradigms for simulating a real in-use network environment, which are intended to be more efficient or scalable than past architectures: controller-microservices virtualization or container-based traffic collection. A network simulation approach



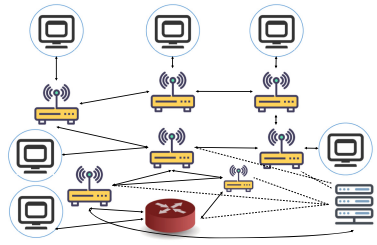
(a) vNIDS [99]: Network traffic arrives into the detection system and is passed through three types of microservices as shown in the figure. Then data is passed into the vNIDS controller, which contains state management that is responsible for detection state classification and provision control responsible for partitioning detection logic programs into header-and payload-based DLPs).



(b) ISOT-CID [9]: The three computer icons represent three hypervisor nodes (A, B, C) that hold 10 virtual machine instances. The yellow icons represent routers, and the cloud is the ISOT cloud network. Internal depicts the internal network that zone A's hypervisor is connected to and VLAN is connected, to zone B's hypervisor.



(c) UGR'16 [109]: The topology of the network begins with the internet represented through the globe icon, which have two routers, two yellow icons, connected to it. The attacker and victims' networks are depicted via computer icons. The two routers are called BR1 and BR2, which stand for border routers. The second border router is connected to the attacker network (five machines). The core network has 5 victim machines used in data collection, which has two firewalls represented by the red icons. The inner network holds 15 victim machines where five machines are placed in each of three distinct existing networks.



(d) LITNET [38]: The yellow icons are routers. The top three connecting nodes are CITY2 (Klaipeda University), CITY3 (Siauliai University), and CITY4 (KTU Panevezys Faculty of Technologies and Business), from left to right. The middle three are CITY1 (Kaunas–Vytautas Magnus University and Kaunas Technological University), KTU University 2, and CAPACITY (Vilnius Gediminas Technical University), from left to right. The lower left router is KTU University 1. The red icon is a firewall and the lower-right icon depicts a netflow server. The four nodes KTU UNIVERSITY 1, CAPACITY, KTU UNIVERSITY 2, CITY1 along with the firewall are netflow exporters that catch new traffic.

Fig. 2. Paradigms of systems used towards real-world network data collection.

can vary based on whether the traffic is being delivered entirely from a commercial generator (e.g., IXIA PerfectStorm Tool) or at least partially derived from an enterprise tracing project. Enterprise tracing projects collect typical traffic in a enterprise network.

The controller-microservices virtualization architecture introduced in Hongda et al. [99] distributes network traffic to different detection logic program instances based on the network header information. The decomposition of the network intrusion detection system into three microservices allows for independent detection for different network traffic, reflecting a "divide and conquer" approach that can reduce resource consumption in the virtualization of NIDS. They generated background network traffic by delivering typical campus, internet, or enterprise traces

of network traffic and labeled attack traffic from penetration tests in an isolated environment. Using typical in-use background network traffic in a simulated environment while introducing malicious attacks during penetration testing can provide estimates of an intrusion detection model's performance in a real-world network.

Expanding from traffic collection in virtual machines, Clausen and others [35] examine containerized networks to address the lack of ground truth traffic origin labels due to traffic originating from multiple background processes and a static design in virtualizations. Virtual machines require the use of hypervisors, which share the host's operating system resources. Containerized networks do not rely on hypervisors and instead share resources simultaneously across containers. Because containers isolate specific running applications, labels on the ground truth of traffic origin can be gathered. Containers are also lightweight and easily clonable, making containerized networks scalable and dynamic to changes in network environments. Attack traffic is produced by a select group of machines in the containerized environment, while benign traffic is generated using commercial traffic generators such as *IXIA PerfectStorm Tool*. The authors also mimicked network congestion by writing scripts in NetEm to artificially create packet delays, loss, and corruption.

Optimal network generation can occur when benign traffic data is generated using typical traffic recorded in an in-use network most similar to the network that will implement the NIDS technology. Malicious traffic can be implemented by other machines in the environment that deploy current attacks such as Denial of Service. This traffic can be relayed in a virtual machine or containerized environment with Hongda and Clausen's architectures being two specific examples, respectively, and can be accompanied with artificial packet loss or delays to mimic real-world network congestion. Although an IDS trained in the artificial network may not perform well in a real network setting, the more similar a simulated training environment is to the network integrating the IDS, the better the IDS will perform.

4.2 Methods for Feature, Label, and Instance-based Challenges

4.2.1 Methods for Feature-based Challenges: Noise, Redundancy, and Weak Correlation.

4.2.1.1 Handling Noisy Data.

Challenge. Some traffic data in datasets may contain outliers that can come in the form of less-frequent traffic classes. To combat noisy data or data with outliers, feature normalization methods have been applied to scale features and allow them to have similar effects in the model so noise would not weigh differently than the rest of the data. In other instances, density-based feature selection was used to identify the most important features by finding overlaps between feature probability distributions as well as non-overlapping regions.

Feature normalization. Feature normalization methods can be applied to scale features and allow them to have similar effects in the model so noise will not be weighed differently than the rest of the data. Statistical methods have been used to facilitate network anomaly classification. In 2015, Delahoz et al. [93] studied a probabilistic Bayesian self-organizing map model to perform unsupervised learning. To overcome the challenge of noise in the network data, they normalized continuous variables to have a mean of 0 and variance of 1, a standard normal distribution. For categorical variables, they are encoded before normalized. Categorical encodings are 1 if a feature is "activated" and 0 if not. Although normalization to a standard normal distribution via $\frac{x-\bar{x}}{\sigma}$ is one method, rescaling logarithmically is another option. Hsu et al. [73] developed an online intrusion detection system based on an autoencoder, SVM, and Random Forest ensemble where noise was dealt with feature normalization, where they used the two normalization functions:

$$\bar{a} = \frac{log(a+1)}{(log(a+1))_{max}}, \quad (1)$$

$$\bar{a} = \frac{a}{a_{max}}. \quad (2)$$

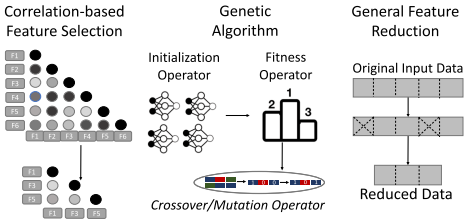
The functions were meant to rescale feature values to the proper range, where a is the original raw data value, a_{max} being the max value among all values under the same feature as a, and $(log(a+1))_{max}$ being the maximum $log(raw\ value\ +\ 1)$ for all logarithmic values under the same feature as a. Packets sent and received were two features that were extremely variable, because certain attacks (DDoS) entail much larger amounts of traffic in the network, so those feature values are normalized by its logarithm divided by its max value (first normalization equation). For features with lower variance, they are normalized by division of their max value (second equation). Specific to the sensitivity to noise innate in SVMs, Liu et al. [106] worked towards mitigating the sensitivity that SVMs have for noise samples by applying a fuzzy membership to measure the distance between a sample and the hyperplane, as in SVM. The larger the distance, the smaller the weight coefficient for the sample. Each sample will have a distinct effect on the optimized classification hyperplane so outliers and noise (values with larger distance) will not impact the classifier plane as much as they are assigned lower weights.

Density-based clustering. In other instances, density-based clustering is used to group together data from the same class and identify outliers that are unusually distant from the clusters observed. Because of the scattered nature of DoS attacks in **wireless sensor networks (WSNs)**, Shamshirband et al. [156] introduced an **imperialist competitive algorithm (ICA)** with density-based algorithms and fuzzy logic. Dense areas in data space are clusters and low-density areas (noise) surround them. Density-based clustering can detect shape clusters and handle noise. As network intrusion detection involves outlier detection, one may broaden the density-based approach to outlier detection. Tang and He [167] presented an effective density-based outlier detection method where a relative density-based outlier score is assigned to observations as a means of distinguishing major clusters in a dataset from outliers. Similarly, Gu et al. [63] applied a density-based initial cluster center selection algorithm to a Hadoop-based hybrid feature selection method for the mitigation of outlier effects.

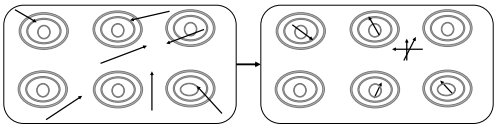
4.2.1.2 Handling Redundant Data.

Challenge. Some features in a network intrusion feature set may not contribute significantly to the predictive power of a model, so they may be removed based on *feature importance*. To handle redundant data, frameworks have been made to remove redundancies. Significant methods to handle redundant features in data are illustrated in Figure 3.

Feature removal frameworks. The presence of data redundancies is a prevalent issue among network intrusion datasets, so researchers have developed frameworks where specific data removal techniques are recommended. Initial feature removal methods were integrated into computational intelligence approaches over the course of the 2000s and into the 2010s. In 2013, Ganapathy et al. [58] wrote a review detailing a gradual feature removal method and modified mutual information method that selects features to maximize information for outputs (maximize relevance between inputs and outputs), **conditional random field (CRF)** as a layered method (each layer representing an attack type), and genetic feature selection where a set of trees are generated and the best set of features are extracted. Recent research appears to be reflective on integrating feature removal methods into a more streamlined model creation process. Bamakan et al. [18] proposed an effective intrusion detection framework where feature selection is embedded in their objective



(a) Correlation-based feature selection [92]: The features are lined up on the horizontal and vertical axes of the correlation map. The method chooses features that are highly correlated with a class, but not correlated with each other. Genetic algorithm [58]: Ganapathy et al. identified a trending feature selection method using genetic algorithm that uses a fitness function and a decision tree where features are removed and model fitness so the optimal feature set is obtained.



(b) Swarm optimization [34]: Chung and Wahid improved normal swarm optimization by conducted a local weighted search to avoid premature "optimal" solutions. Particles are shown as arrows in the figure and are updated by evolutionary operators. Depending on its fitness, its location is updated until the final feature set is optimal–the distribution of resulting particles after optimization shown below the first rounded rectangle.

Fig. 3. Methods for dealing with redundant features: Besides the above two methods, Autoencoder [8] and Fuzzy Rough Set [153] have also been used for reducing redundant features.

function combined with **time-varying chaos particle swarm optimization (TVCPSO)**. They streamlined their weighted objective function approach in a flow chart where, with each iteration, the fitness of the particles is updated in particle swarm optimization and chaotic search is done to find the global optima. Carrion et al. [110] addressed the lack of the evaluation in network intrusion detection methods by providing a structured methodology that involved more rigorous feature selection or removal techniques. Including steps on how feature selection or removal took place to arrive at a final accuracy, as they stated, can allow for easier replication and more reliable evaluation in network intrusion detection literature.

Feature selection. Feature selection can rule out redundant features and select a subset of the features in the data without significantly degrading the performance of the model [76]. Early 2010's saw an interest in filtering-based feature selection methods as Koc et al. [92] applied the **hidden naïve Bayes (HNB)** model to data with highly correlated features. Accompanying their HNB model was a filter-based feature selection model that is both correlation and consistency-based and relies only on the statistical properties in the data. Correlation feature-selection picks features that are biased towards highly correlated classes. The consistency-based filter has an inconsistency criterion that specifies when to stop reducing the dimensionality of the data. After filter-based methods, there was interest in using forward selection for feature ranking via Random Forest by Aljarrah et al. [7]. But rather than finding the most optimal feature set, recently, Elmasry et al. [45] claimed that feature selection can be time-consuming due to its exhaustive search, and that evolutionary computation techniques may be applied to find near-optimal solutions in a shorter amount of time.

Automatic feature extraction. In the realm of automatic feature extraction, rough set theory and autoencoders are two important automation methods. Rough set ranks extracted features from network intrusion data and generalizes an information system by replacing the original attribute values with some discrete ranges [14], and autoencoders are considered to be nonlinear generalizations of principal component analysis that use an adaptive, multilayer "encoder"

network to reduce data dimensionality [71]. The early 2010's saw research interest in rough set theory for feature selection. Because **simplified swarm optimization (SSO)** may find premature solutions, Chung and Wahid [34] went about improving the performance of it by conducting a local weighted search after SSO to produce more satisfactory solutions. They applied k-means clustering to continuous network data values and rough set theory to minimally sized subsets of the feature. The goodness in selected features is evaluated using the fitness function given input data D, $|C|$ being the number of features, $|R|$ being the length of a feature subset where R is a feature subset, and γ_R as the classification quality of feature set R:

$$\alpha \times \gamma_R(D) + \beta \times \frac{|C| - |R|}{|C|}. \tag{3}$$

Data is changing rapidly and with the increasing presence of irrelevant features, Liu et al. [105] introduced a Gaussian mixture model to extract structural features in a network and identify anomalous and normal patterns where redundant features were removed and important features were optimally selected using fuzzy rough set theory. Alongside irrelevant features and the age of big data, the speed in which a model's objective function converges slows down. Both fuzzy rough set methods and autoencoders have been devised to tackle the large volume of data. With uncertainty surrounding whether network traffic is normal or anomalous, Selvakumar et al. [153] presented a fuzzy rough set attribute selection method where the fuzzy-oriented rough degree on dependency of $\gamma'_P(D)$ to subset P is defined as $\gamma'_P(D)$, where a subset of features is evaluated on its relevance to the data. To handle growing data, as well as irrelevant data, Alqatf et al. [8] proposed the use of an autoencoder for feature learning and dimensionality reduction to extract the most important features and filter out those that are redundant. Then they pass the reduced data into an SVM model for network traffic classification.

4.2.1.3 Handling Weakly Correlated Data.

Challenge. The lack of strong correlation between features in data may make the construction of a model more challenging. Correlation can be artificially made through increasing the dimensionality of the data by data fusion or the introduction of new features.

Increase dimensionality. Given one-dimensional feature data, Li et al. [102] augmented the data to two dimensions and performed data segmentation where split data was later fused back together for network intrusion classification. They split feature data into four separate parts based on features that are correlated with one another. The one-dimensional feature space is converted to grayscale, then the data output from the four data components are merged and passed to the output layer of the multi-fusion CNN.

4.2.1.4 Discussion on Model Robustness.

Challenge. A model is robust if the accuracy in its predictions is not impacted by changes in the input data such as from distribution shifts or outliers [186]. For intrusion detection, changes in network flow data can also come from an "adversary" that may "obfuscate" attack payloads to mimic its benign counterparts. To mitigate loss in intrusion detection accuracy due to noise or adversaries, different robust methods have been developed.

Robust Methods. Gornitz and others [62] reframed network anomaly detection as an active learning task and tested the robustness of a one-class SVM. They first viewed a network payload as a vector $\mathbf{x}$ containing data of the network packet and mapping it to a vector space using string s and embedding function ϕ. For each string s, $\phi(\mathbf{x})$ returns 1 if s is in the payload $\mathbf{x}$ and 0 otherwise. Using this vector space representation, the **support vector domain description (SVDD)**

was designed so normal data could be separated from anomalous data, where anomalies could easily be distinguished as outliers. They guided a security expert to low-confidence regions of the feature or vector space to administer active learning, where more attention was directed towards network data that had less accurate predictions. Their SVM was not designed to be robust against adversaries or noise in the network data, because model robustness was not factored into the construction of the SVM. It was during the empirical evaluation of their approach when they analyzed the effects of an adversary on their model's performance.

Recent work focused on designing methods robust to distributional changes or the presence of outliers in network data. Papers either tackle method-specific limitations in robustness such as the sensitivity of SVMs to noise or general limitations that result in high false positive rates or undetected false negative outliers. Bamakan and others [17] use ramp loss to remedy support vector's sensitivity to outliers. The ramp loss replaces hinge loss, which is a non-convex loss function that "depresses" the pressure of these outliers to make support vector models more robust and reliable. Armed with a new loss function, the authors apply the "concave-convex" procedure to minimize ramp loss by selecting the optimal s for the ramp loss $R_s(\mathbf{z})$ given the input vector $\mathbf{z}$. Then for each pair of $\forall i, j \in \{1, \ldots, p\}$ (i, j) for p output labels, the training dataset is partitioned into a positive, negative, and zero class. Variables $\delta^1,\ k = 1$ are initialized and an iterative update to δ^i for the ith iteration is applied to construct a decision function of the form $g(\mathbf{x}) = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq \epsilon_0 \\ -1 & \text{if } f(\mathbf{x}) \leq -\epsilon_0 \\ 0 & \text{otherwise} \end{cases}$. A network packet observation $\mathbf{x}$ has its label predicted based on the constructed decision functions. The ramp loss results in more sparsity, or more zero quantities, in the support vector model, because misclassified training examples do not simply become support vectors.

To tackle the general problem of being unable to capture an intrusion signal occurring on a network, Ahsan and others [5] applied kernel density estimation to Hotelling's T^2 control chart to design robust estimators. The Hotelling's T^2 chart can track the mean of a process where we observe independent and random vectors $\mathbf{x}_i$. The procedure is designed so, first, a matrix $\mathbf{X}_{normal}$ is constructed, which is the representation of the benign training data in the network. Then fast **minimum covariance determinant (MCD)** is then run where Manhalanobis distances are computed per $i \in \{1, \ldots, n\}$ and the distances are sorted to form some permutation of the original set $\{1, \ldots, n\}$. $\mathbf{T}$ and $\mathbf{S}$ are the mean and covariance of the new permutation set. The T^2 statistic in the Hotelling control chart using the mean vector and covariance matrix is computed for normal network connection data: $T^2_{Fast-MCD,\,i} = (\mathbf{x}_i - \mathbf{T})^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{T})$. The distribution of the T^2 statistic is computed with **kernel density estimation (KDE)** using the Gaussian kernel. Then the control limit is calculated using the CDF of the empirical density of the $T^2_{FMCD,\,i}$ statistic: $CL_{KDE} = \widehat{\widetilde{F}}^{-1}_h(\tilde{t})(1-\alpha)$. The design of the robust method is centered around the computation of robust estimators for the Hotelling T^2 control chart, which can result in more accurate control limits, or horizontal lines denoting the boundaries between normal and anomalous data, in Hotelling T^2 charts.

4.2.1.5 Discussion on Tolerance to Adversarial Attacks.

Challenge. An "adversary" can be a data generator or a network security expert that can mask network payloads to appear benign when they are in fact malicious. They have the intention of deceiving an intrusion detection system and leaving attacks on a computer network undetected. It is challenging to determine how an adversary will behave and ways to mitigate false positives or false negatives when detecting network intrusions, but methods using data generation have been devised to help models handle adversarial situations.

Against Adversarial Attacks. Marino and others [113] implemented an adversarial approach that sought to enable a machine learning model to correctly classify misclassified examples given they

are modified by adversarial sample generation. Rather than deceiving their classifiers that have a linear model and a multi-layer perceptron model, the authors intend to understand why their network data is being misclassified. They find an adversarial example $\widehat{\mathbf{x}}$ that is classified as $\widehat{\mathbf{y}}$ while minimizing the distance between $\widehat{\mathbf{x}}$ and the original data $\mathbf{x}_0$: $min_{\widehat{\mathbf{x}}}\ H(\widehat{\mathbf{y}}, p(\mathbf{y}, \widehat{\mathbf{x}}, w))\alpha I_{(\widehat{\mathbf{x}},\widehat{\mathbf{y}})} + (\widehat{\mathbf{x}} - \mathbf{x}_0)^T Q(\widehat{\mathbf{x}} - \mathbf{x}_0)$ with the constraint that $\mathbf{x}_{min} \preceq \widehat{\mathbf{x}} \preceq \mathbf{x}_{max}$, where Q is a positive semidefinite matrix that can be adjusted using specific weights. Note that $\widehat{\mathbf{y}}$ is not the same as the true label of the network data observation $\mathbf{x}_0$. $\mathbf{x}_0$, $\mathbf{y}$ is a sample from the dataset. $H(\widehat{\mathbf{y}}, p(\mathbf{y}, \widehat{\mathbf{x}}, w))$ is the cross-entropy between the estimated label of the adversarial example, $p(\mathbf{y}|\widehat{\mathbf{x}}, w)$, and the target label $\widehat{\mathbf{y}}$. α weighs the contribution of the cross-entropy in the objective function and $I_{(\widehat{\mathbf{x}},\widehat{\mathbf{y}})}$ is the indicator function that returns 0 if $\widehat{\mathbf{x}}$ is classified is $\widehat{\mathbf{y}}$ and 1 otherwise. The quadratic program optimization above is run and minimum changes are applied to correctly classify only misclassified samples $\mathbf{x}_0$, where we intend to construct $\widehat{\mathbf{x}}$ such that $\widehat{\mathbf{y}}$ is the predicted label. Explainability in misclassifications can be derived from adversarial sample generation, as the average of the deviations $\mathbf{x}_0 - \widehat{\mathbf{x}}$ can be used to explain how far off the misclassified samples are from samples that would be classified correctly.

Instead of using one adversarial attack generation method, Pawlicki and others constructed a pipeline for artificial neural networks that uses four different approaches, each minimizing the distance between the produced adversarial and real samples. The input network traffic dataset is partitioned into four subsets A, B, C, D. Portion A is used to train the intrusion detection system. Portion B is split between testing the detection system and training the adversary detector by performing four adversarial attacks—Carlini and Wagner attack, fast gradient sign method, basic iterative method, projected gradient descent—on 1397 "Attack" samples that are labeled as "adversarial." The remaining data labeled as "non-adversarial" are added to the "adversarial" examples and form the adversarial detector training dataset. C, D are used to test the adversarial detector.

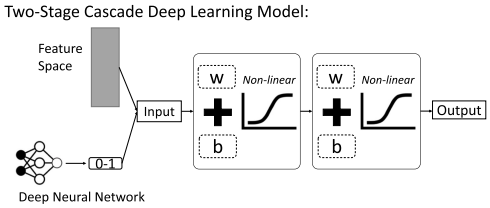
4.2.2 Methods for Label-based Challenges: Lack of Labels and Label Imbalance.

4.2.2.1 Handling Too Few Labels.

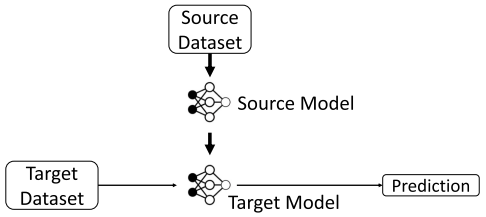
Challenge. Data may have a lack of labels, particularly when network traffic is ambiguous or unlabeled. This poses another challenge between the stages of data preprocessing and model creation. Figure 4 illustrates transfer learning, adversarial sample generation, and deep learning paradigms used to resolve the issue of unlabeled data.

Unsupervised/Supervised Learning. Initially with a completely unsupervised learning approach, Casas et al. [26] used an unsupervised sub-space clustering method to detect network intrusions by aggregated traffic packets into multi-resolution traffic flows. With too few labeled data, researchers may look to semi-supervised learning: First perform unsupervised learning on unlabeled data to label it, then pass the labeled data to a supervised learning model. More recently, there has been more research on semi-supervised network intrusion detection methods. Khan et al. [90] proposed a semi-supervised model that initially classified unlabeled traffic as normal or anomalous with a probability score that was used as input for an unsupervised autoencoder to train on, then the data was passed into stacked pretrained neural layers with a soft-max layer for classification. Through a more randomized approach, Ravi and Shalinie [142] proposed a semi-supervised learning model that employed repeated random sampling and k-means to label data as different traffic types, then passed it through classifiers developed in related work.

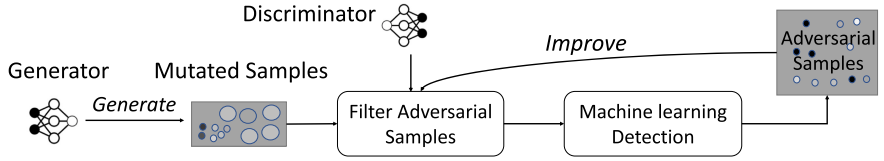
Transfer learning. Transfer learning can compensate for the lack of labeled data via transfer of knowledge from other labeled data sources [107]. Singla et al. [161] examined the viability in transfer learning for imbalanced datasets; namely, the UNSW-NB15 dataset was split into labeled sub-datasets. Each sub-dataset is split into a source dataset and a target dataset, where the classifier



(a) Semi-supervised learning [90]: In the initial stage, Khan et al. used a deep neural network that will predict whether a traffic observation is normal or anomalous using a probability score. This is used as an additional feature in the stacked autoencoder model represented by the two rounded rectangles containing the softmax layers.



(b) Transfer learning [161]: Singla et al. adhered to the transfer learning heuristic above where representing the source model is trained on a target dataset as the knowledge from training on the source dataset carries over, which produces a target model used for anomalous prediction.

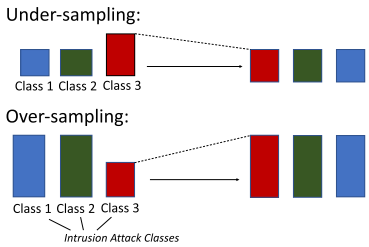


(c) Adversarial sample generation [32]: Among the methods to generate adversarial samples, Cheng et al. used a generative adversarial network (GAN) that used a generator to produce fake data fed along with real data into a discriminator that "Filters Adversarial Samples" and outputs predicted labels. The loss from the predictions are used to refine the adversarial samples that are again fed into the discriminator to distinguish real vs. fake data.

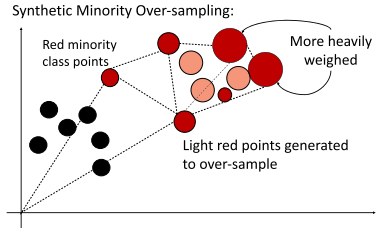
Fig. 4. Illustration of main methods towards handling too few labels in network data.

was pretrained on the source dataset, then retrained on the target dataset to combat the lack of labeled data. Beyond the synthetic dataset UNSW-NB15, a network type that has recently been explored was the consumer network, which does not have the firewall or switches to deter network intrusion attacks. Patel et al. [136] proposed normalized entropy from features in payload, packet and frame statistics to be funneled into a training and testing dataset and passed into a one-class SVM for classification of traffic in consumer networks. Through the collection of packet capture data sent from programs on devices in a consumer network, a consumer network dataset was constructed to combat the lack of consumer network datasets. Patel and other researchers exhibited that exhaustively labeled datasets are not necessary for accurate intrusion detection models.

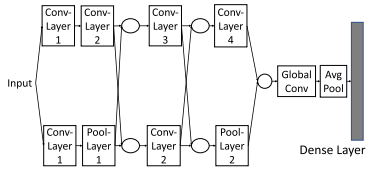
Adversarial sample generation. Adversarial sample generation is done to fool a machine learning model, especially a neural network, with adversarial samples, so correctly classified data can be mistakenly identified for another class [86]. Using a random forest, Apruzzese et al. [15] used network flows to identify normal and botnet activity. The adversary is assumed to have already compromised at least one machine in the network and deployed a bot to communicate with other machines through limited "Command and Control infrastructure." The attacker intends to trick the classifier by slightly increasing flow duration and exchanged bytes and packets. Instead of a base adversary that changed feature attributes in adversarial samples, Cheng [32] used a **generative adversarial network (GAN)** where a generator aims to refine the generation of fake data while a discriminator determines which network traffic flows are legitimate or anomalous, which was what Usama et al. did as well [172]. Similar to Apruzzese's adversarial sample generation, but in a



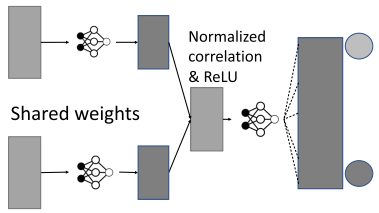
(a) Undersampling [117]: Mikhail et al. applied random undersampling, which is illustrated in randomly sampling less of class 3 to enable equally sized class datasets. Another idea to make the classes of equal size is oversampling.



(b) Synthetic Minority Over-sampling (SMOTE) [193]: Zhang et al. implemented a minority oversampling technique that weighs harder data examples more heavily, which will be synthesized more. The light red points are synthesized, which are from the large red minority class circles that are more heavily weighed.



(c) Parallel Convolutional Neural Network (CNN) and feature fusion [194]: Zhang et al. used feature fusion and a parallel CNN. The top branch of convolutional layers is responsible for pixel-level classification. The lower branch of convolutional and pooling layers mitigates redundant features from majority class samples via down-sampling. Then output feature maps are fused at the while circles at different stages. A global average pooling layer is used to further reduce redundant features.



(d) Siamese neural network [19]: Bedi et al. used a siamese neural network that accepts some input, illustrated as the two left gray parallelograms, and are accepted by two identical sub-networks containing the same weights. The networks extract feature representations that are passed into a fully convolutional network that results in a prediction of normal or anomalous traffic (the rightmost circles).

Fig. 5. Highlighted machine learning and sampling methods for imbalanced labels.

more controlled environment, Aiken and Scott-Hayward [6] developed an adversarial testing tool called *Hydra* that behaves as an emulator for a system that launches attacks in a software-defined network where a test manager sends traffic, evading the classifier by changing payload size and rate. In all such cases, adversarial sample generation not only offers more data to combat unlabeled data, but can develop defensive mechanisms for more robust NID systems.

4.2.2.2 *Handling Unbalanced Labels.*

Challenge. The data may also be imbalanced where network intrusion attacks are disproportionately smaller than that of normal network activity. As discussed in Section 3 on common public datasets, most network intrusion datasets face considerable imbalance between normal and anomalous traffic, but especially among attack types. Figure 5 highlights the random oversampling and undersampling techniques used to handle minority and majority classes in network intrusion datasets and main machine learning models implemented to handle unbalanced classes.

Over/under-sampling. Oversampling is meant to increase samples from the minority class and balance the distribution of data among attacks and normal activity in a network. Undersampling

removes samples from the majority class to allow minority and majority classes to become similar in size, disallowing misclassifications of underrepresented network attacks [200]. A collection of work has been written last year on the use of over or under sampling to balance network intrusion datasets. Mikhail et al. [117] resolved the issue of minority attack classes by training an ensemble classifier with undersampling data and training each sub-ensemble. Gao et al. [59] noticed that the KDD Cup dataset they used had a large amount of **user to root (U2R)** attacks, so they changed the proportion of classes in samples passed as input into the models. They used **classification and regression trees (CARTs)** where multiple trees were trained on adjusted samples by random undersampling—similar to Mikhail and others' work—where $\frac{1}{16}$ of normal traffic (a majority class) was sampled to solve imbalances. Although minority sampling can allow for more evenly proportioned classes for network intrusion detection, there is the potential for majority classes to be predicted with lower accuracy due to undersampling of majority classes or oversampling of minority classes. Zhang et al. [193] resolved this issue by combining weighted oversampling with a boosting method. The weighted oversampling technique updates weights associated with minority classes and the misclassified majority class observations are forced on the classifier to learn.

Optimal feature extraction. Ranking features based on their importance can be done to reduce a feature set to an optimal feature subset. Thaseen et al. [170] used a consistency-based feature selection method that determines whether the value and class label of two observations match. Zhang et al. [196] aggregated time intervals of network traffic into subgroups to find information from the five features: address count, packet count, port count, byte count, and the bytes per packet.

Siamese neural network. To combat the challenge of minority and majority classes in imbalanced datasets, Bedi et al. [19] employed a few-shot learning method called a Siamese Neural Network that was first introduced by Bromley et al. [22]. Siamese neural networks compute the similarity between two input samples to determine how similar or dissimilar they are, so pairs of samples belonging to the same class such as DoS-DoS, Normal-Normal, U2R-U2R were considered most similar and labeled with a 1, whereas distinct pairs were labeled with a 0. Traditional methods of oversampling and undersampling were bypassed with the use of siamese neural networks paired with sampling equal number of observations per network traffic class.

Feature fusion. Feature fusion can combine different data that will, together, result in balanced attention to features. Zhang et al. [194] implemented a parallel cross convolutional neural network that fused traffic flow features learned by two separate convolutional neural networks to make the network pay more attention to minority attack classes. After downsampling the two neural networks, the number of channels was doubled in the output feature map, then a pooling layer was applied to reduce the dimensionality of the data by combining outputs of clusters in one layer into one neuron in the next layer.

Genetic programming. Genetic programming uses an agent to learn an optimal or near-optimal solution to a problem [178] and can be used in conjunction with machine learning models to evolve the model until its fitness is optimized for network intrusion detection. Le et al. [97] found genetic programming to perform well on imbalanced datasets when using accuracy as the fitness function: the number of true positives and true negatives over all classified observations.

4.2.2.3 Discussion on Explainability.

Challenge. Model explainability is being able to understand the behavior of a machine learning model and why it presents specific solutions or predictions for given tasks [56]. Explainability in network intrusion detection models is important due to the side effects of a wrong prediction. If a model predicts a false negative, or presumes that a network packet is not anomalous or malicious

when it is responsible for intrusion of a computer network, then the entire network could be in jeopardy, leading to possible breaches in private information. And if one cannot explain why a model made this poor and costly decision, then individuals will be less accepting of novel intrusion detection systems powered by machine learning models. Thereby, research into more explainable intrusion detection models can heighten the chances of integrating the model into a corporate or public network.

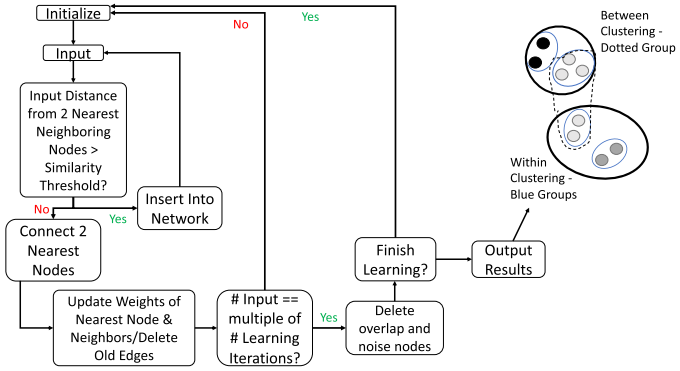
Explainable Methods. Explainable network intrusion detection models have been gaining traction such as in References [12, 13, 77, 104, 111, 127, 151, 165, 175, 203], but there are a few representative papers that portray the changes of explainable IDS methodologies. Adetunmbi and others [3] explored seeking model explainabillity using a rule system. Their rule system is based off of rough set theory, where rules are denoted by logical implications. Based on the values of a set of features, or attributes $A_1, \ldots, A_k$, the rules $(A_1 = a_1) \wedge \ldots \wedge (A_k = a_k)$ will decide the outcome of whether or not a network connection record is anomalous. Although rough sets can generate simple decision rules that are explainable to practitioners and indeed three times faster than a traditional k-nearest neighbors approach that they implemented, the approach did not perform well on **remote to user (R2L)** or U2R attacks in the KDD Cup 1999 dataset. To improve detection accuracy while maintaining model explainability, Amarasinghe and colleagues [12] present a post hoc framework of explanations for their **deep neural network (DNN)** predictions. They provide the relevance of input features for the prediction, the prediction confidence, and a textual summary of why the prediction was made such as "DoS attack WITH high confidence BECAUSE connections to the same host in the last 2 seconds is high." The textual summary and feature relevancy are similar to rough set's decision rules that are based off relevant features. Nguyen and others [127] combat the limitations of excessive amounts of labeled data needed to train supervised learning models such as DNNs. They examine an unsupervised method called a variational autoencoder and use gradients generated by the autoencoder to form "gradient-based fingerprinting" and explain anomaly predictions. A fingerprint is formed from the feature and corresponding gradient of it.

4.2.3 *Methods for Instance-based Challenges: Dynamics and Large or Too Small Volume.*

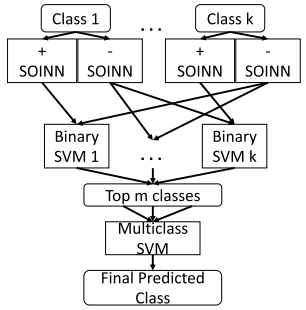
4.2.3.1 *Handling Dynamic Data.*

Challenge. Due to the changing landscape of new data being generated daily, adaptive models have been ever more important to dynamic data, especially as data has been growing exponentially for the past decade and now the digital world contains roughly 2.7 zetabytes [24]. Figure 6 summarizes the significant and novel dynamic network intrusion models developed recently.

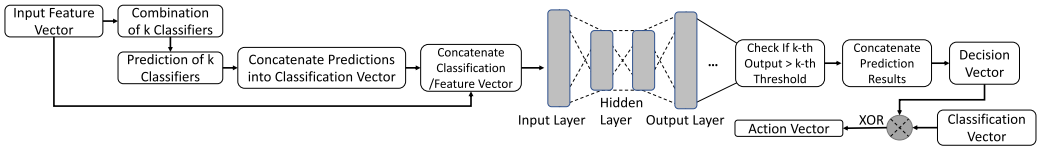
Stream-based models. Since dynamic data may come in the form of a stream, researchers have looked at specializing model for stream data. To resolve the issue of irrelevant data in dynamic streaming data, Thakran et al. [169] employed density and partition-based clustering methods along with weighted attributes to handle noisy data in streaming data, which was used for outlier detection. For better real-time responsiveness from intrusion detection models, HewaNadungodage et al. [70] accelerated outlier detection with parallelized processing power from a **graphics computing unit (GPU)**. Instead of improving upon the real-time speed in which outliers are detected, Noorbehbahani et al. [128] looked towards a more adaptive model that uses incremental learning, which still performs well with limited labels in streaming data. They implemented a **mixed self-organizing map incremental neural network (MSOINN)** and "within and between" clustering for offline and online learning. An initial cluster set from the network training data and initial classification model are generated during the offline phase. Clusters are updated



(a) Noorbehbahani et al. semi-supervised model [128]: Noorbehbahani et al. used a semi-supervised model where a mixed-data self-organizing incremental network is trained and continuously updated with new data. The unsupervised learning takes place with both within and between clustering illustrated above in the figure. The leftmost flowchart illustrates how new inputs are fit into the incremental network during offline learning. When the model is online, the old cluster sets are updated and the old incremental network is used to classify new data, depicted in the two rightmost rounded rectangles. Also note that the green branches indicate an answer of yes to question nodes. Answers of no are represented by red branches.



(b) Constantinides et al. self-organizing incremental neural network [36]: The detection system is initialized with a dataset containing k attack classes. Each attack class category is modeled with two self-organizing incremental neural networks. The input vector per SVM is constructed from that SVM's positive n-SOINN and other negative n-SOINNs from the other classes.

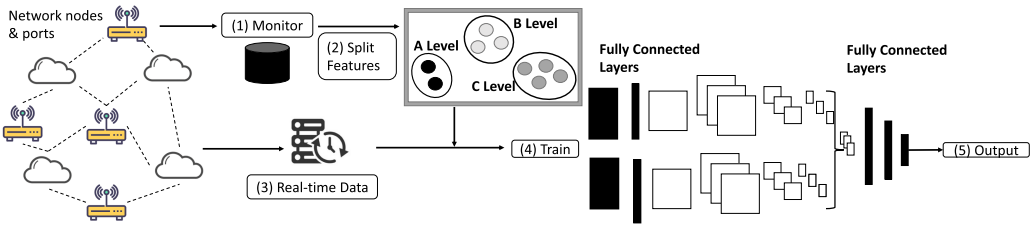


(c) Sethi et al. Agent Network [154]: Sethi et al. employed reinforcement learning by initially passing in an input feature vector that concatenates predictions of k classifiers with the input feature vector as input for the agent network, which is depicted in blue. Then decision vector is composed with the classification vector (either 1 (attack) or 0 (normal)) in an XOR to form the agent network's output vector (action vector).

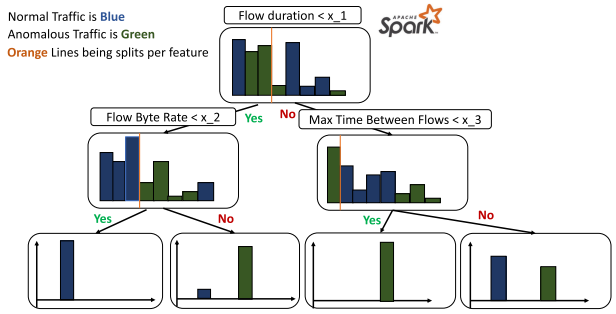
Fig. 6. Paradigms of dynamic network data models.

with the MSOINN clustering algorithm, and new observations are classified with the MSOINN model during the online phase of learning.

Reinforcement learning. Reinforcement learning is one type of machine learning that learns a mapping, or a policy, between the states of a system and the actions it can execute given a reward and punishment notion [115]. Through an adaptive approach, Bensefia and Ghoualmi [21] proposed the integration of an adaptive artificial neural network and a learning classifier system that uses a reactive learning base to learn new attack patterns. There has recently been research on cloud environments and applying reinforcement learning to changing data in the cloud by Sethi et al. [154], who applied reinforcement learning to the cloud where a host network communicates with an agent network through VPN. Log generation from the virtual machine was provided to an agent that applied a deep Q-network and compared the model's result with the actual result from the administrator network, calculating the reward and iterating until the reward was maximized.



(a) Chen et al. DDoS Multi-Channel Convolutional Neural Network Incremental (MC-CNN) Learning Model [29]: Chen et al. implemented a multi-channel incremental network that monitored network traffic and packets, inputting the data into a database (represented as a black cylinder). Features are split into traffic, packet, and host level, which are represented through the A, B, C levels. Then the real-time data from the network along with the partitioned features are passed into a multi-channel CNN where the top branch accepts traffic features and the lower branch takes in packet features. The top layer undergoes pooling to reduce parameter complexity in the fully connected layer. The top and bottom branches are combined and passed to the fully connected layers and a final prediction is outputted.



(b) Morfino and Rampone Apache Spark with decision tree [121]: Morfino et al. used Apache Spark's Machine Learning Library, MLlib, which stores filter/map operations in a directed acyclic graph and uses "Catalyst" to optimize an efficient execution plan. The decision tree splits a distribution by features until splits divide features into more homogeneous groups of normal and anomalous traffic.

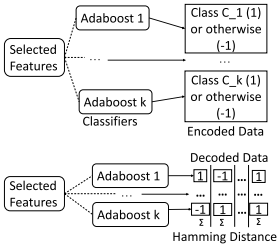
Fig. 7. Novel methods to handle big volume of data.

Incremental learning. With data in dynamic environments, it is necessary that pretrained models are updated with new data in an *incremental fashion without compromising classification performance on preceding data* [138]. Addressing botnet intrusion attacks, Feilong Chen et al. [28] argued that botnet detection starts with the set of server IP-addresses visited by past client machines, so an incremental least-squares SVM was implemented to be adaptive to feature and data evolution. Meng-Hui Chen et al. [30] made a population-based incremental learning method that learned from evolved data through past experiences and applied collaborative filtering to automate classification, adapting to key features in the data. A shift towards more scalable applications came with an online neural network accompanied by an SVM in Reference [36].

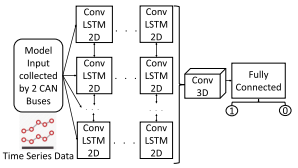
4.2.3.2 *Handling Big Data.*

Challenge. For big data, processing such large amounts of data is overwhelming, so optimization methods were devised to speed up preprocessing, such as reduction methods that remove redundant features and reduce the size of the data. Figure 7 depicts the paradigms from three pivotal methods handling large amounts of data using incremental learning, parallel processes, and Apache Spark for Cloud Computing.

Incremental learning. To handle such large amounts of data, incremental learning may be applied to process it in increments. Chen et al. [29] implemented an incremental training method that



(a) Adaboost and Error Correcting Output Code (ECOC) [1]: Abdelrahman et al. initialized a selected group of features that is distributed among k Adaboost classifiers and encoded in a binary string of length k. The bit positions are shown in the decoded data figure and each classifier is applied to every data observation to obtain a new binary string that labeled with the traffic class closest to it (with lowest Hamming distance, or least number of distinct bits).



(b) Transfer learning with LSTM network [168]: Tariq et al. remedied the problem with a small amount of time-series data for CAN bus network intrusions by collecting data on two CAN buses that is dispersed across multiple convolutional LSTM 2D networks. The timesteps in the time series are transformed into a two-dimensional multivariate time series that convolutional LSTMs were trained on. The outputs on the 2D data form a three-dimensional output, which is passed through a fully connected layer and final predictions of normal or anomalous is outputted.

Fig. 8. Novel small data transfer and meta learning.

repeatedly trained one convolutional layer then added another layer to a **convolutional neural network (CNN)** as new data came in until the target structure was achieved for the final CNN to optimize training time.

Cloud computing. With the advent of cloud computing platforms such as **Amazon Web Services (AWS)** and the Apache software foundation, using virtual services is not only available, but fast. The current research interests appear to lie in implementing machine learning with the Apache services. Manzoor and Morgan [112] used Apache Storm to accelerate intrusion detection and employ a *real-time* support vector machine-based intrusion detection system; Faker and Dogdu [46] used Apache Spark to implement deep feed-forward neural network, random forest and gradient boosting tree methods; most recently, Morfino and Rampone [121] used the MLlib library of Apache Spark to reduce training time for their highest performing model—a decision tree—so they can fit the model to over 2 million rows of data and tackle SYN-DOS attacks.

4.2.3.3 Handling Small Data.

Challenge. The concomitant challenge with a growing network intrusion data repository is the continued lack of data on more current, diverse network attack types. As seen from Section 3, datasets have been riddled with a lack of evenly represented attack classes. Some datasets are dominated by specific attacks, while other types of attacks are underrepresented. Some datasets are dominated by non-attack (benign) class and all the attack types are minority classes. To resolve the issue of small amounts of data, specifically a lack of attack types, meta- and transfer-learning techniques have been explored. Novel machine learning models implementing the two techniques are highlighted in Figure 8.

Meta learning. Meta-learning uses automated learning to improve the way in which a model learns from data. Typically data is split into learning and prediction sets. The support set is in the learning set, and training and testing sets are in the prediction set. In "few-shot" learning, prediction error on unlabeled data is intended to be reduced given only a meager support set. Panda et al. [134] conducted learning with multiple classifiers where *ensembles of balanced nested*

dichotomies for multi-class problems were employed to handle multi-class datasets and make intelligent decisions in identifying network intrusions. A similar ensemble-based method using bagging and Adaboost was proposed by Abdelrahman and Abraham [1]. They implemented the meta-learning technique of **Error correcting output code (ECOC)**, where, per attack class, a binary string of length k is made so each bit is a classifier output and the class with closest string of outputs are returned and used for classification. As a direct response to handling the limited number of malicious samples in network data, Xu et al. [180] devised a few-shot meta-learning method that used a deep neural network and a feature extraction network. The few-shot detection begins with a comparison between feature sets extracted from two data flows and a delta score indicating how different the two input data flows are. During the meta-training phase, samples from query and sample sets are compared and average delta scores are calculated. During meta-testing, samples from the test set and support set are compared and predicted labels for samples are the ones with the minimum average delta score in the support set.

Transfer learning. Just as with the lack of labeled data, transferring knowledge from other data sources through transfer learning can resolve issues of a lack of data, specifically on attack types. Because generating labels for data can be time-consuming, Zhao et al. [197] employed a heterogeneous feature-based transfer learning method to detect network anomalies that was compared to other feature-based approaches such as HeMap and **Correlation Alignment (CORAL)**. Rather than feature-based methods, mimic learning has been applied as a means of transfer learning by retraining a parent model—pretrained on private data—on public data to protect privately collected data and improve accuracy in the final model. Shafee et al. [155] transferred the knowledge from a privately trained model—a random forest that performed best during experimentation of the teacher model—to a public training setting, producing a shareable student model. More niche to robust vehicles, **Controller Area Networks (CANs)** were revealed to be easily exploited and that there was a lack of intrusion data on CANs. Thereby, Tariq et al. [168] recently collected CAN traffic data using two CAN buses and applied transfer learning to train a convolutional long-short term memory network on the new intrusion data.

5 RESEARCH TRENDS AND FUTURE DIRECTIONS

Figure 9 displays the trends of research interests from 2010 to 2020 on data-driven NID methods.

5.1 Research Trends

Upon examining literature from the past decade on NID, there was already a pre-existing interest in big data research since 2010. This interest can be attributed to the large amounts of data on the Internet since 2010, as mentioned in Section 2 of the article, which continued growing through the past decade. 2019 saw the largest number of articles on big data where researchers continued to study parallel processing techniques and incremental learning methods to handle processing large amounts of data. In 2010, there was also effort put into resolving the challenge of small data.

Although there were large amounts of data, data on different attack types were lacking, as exhibited in the datasets attack type breakdown and entropy analysis in Section 3 of the article. In general, the lack of network intrusion attack types, pertinent to the challenge of small data, comes from the typically short time frame that intrusions take place. Small data issues were first researched in the early 2010s, particularly with meta-learning.

With noisy data challenges, authors have done more extensive research into methods that weigh noisy observations over others in network intrusion datasets since 2017. Although there have not been many papers on handling noisy data, solutions to noisy data have been well established, such as rescaling features or using density-based feature selection.

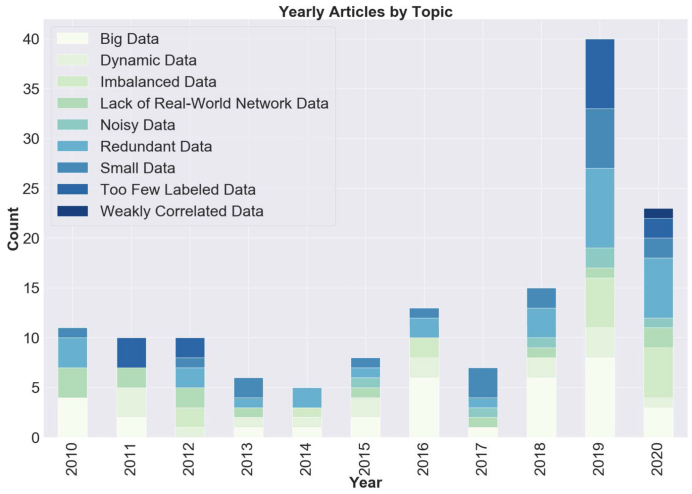


Fig. 9. Data-driven network intrusion papers by topic 2010–2020: Topics were chosen on the basis of the most common research areas covered in highly cited papers published from 2010 to 2020.

The majority of the research between 2010 and 2015 studied ways to work around big data and too few data. Since then, big data processing has remained a popular research topic in the field of network intrusion detection. However, due to the changing environment of the present-day databases, research addressing dynamic data issues has gone up. The lack of labeled data has seen more research proposing semi-supervised learning models. However, one area of research that has not seen much attention is real-world network data. 2010 and 2011 saw some honeypot emulation of networks for data collection, and it was only recently in 2020 when the LITNET dataset [132] was made and released as one of the first real-world network intrusion datasets.

5.2 Discussion on Future Directions

5.2.1 Real-world Data Collection. There was initially a step towards real-world network intrusion data by emulating a realistic network environment with honeypots that would attract attackers with synthetic (IXIA) data generation. However, simulated data may not be as valuable to fit and test a model on as data collected on a real-world network due to possibly incorrect network attack models and behaviors in sandbox network environments. The issue with the current research on applying models to network intrusion is that 46 of the papers in the taxonomy used the KDD Cup 1999 as an evaluation dataset for their models. Because it is synthetically generated, there is bias in the traffic patterns that real-world traffic would not have. A step towards more modern network attacks on a real-world network came with the LITNET dataset [132] collected in 2020 on a Lithuanian network covering nodes in four major Lithuanian cities as being one of the first long-term (10 months) and real-world network intrusion datasets produced and made available for researchers. Realism and availability are the two significant areas that current network intrusion datasets should be striving to have, which will be a future goal for researchers interested in creating new real-world datasets. Currency and realism in normal network traffic and attacks are problems confirmed through word vector analyses in the figure referenced earlier. In turn, network intrusion research requires further data collection of realistic attacks in real-world networks.

5.2.2 Labeling Real-world Traffic. Although traffic flows may be labelled manually by network security experts, real-world network traffic flow can easily grow into the millions. The UGR dataset

from 2016 [51] was labeled using log files from the honeypot system used for data collection. Often experts may be the ones responsible for labeling traffic data, while other datasets such as LITNET in 2020 [132] are less clear on how labeling took place. Labelling training data has been a roadblock for anomaly-based intrusion detection since the late 2000s [37]. Labeling traffic too scrupulously may go against privacy policies, so detection models tend be updated whenever data becomes labeled and manual labeling still occurs with offline learning [177]. To handle newly labeled data being fed into intrusion detection models, there should be further development in adaptive models or incremental models such as an online incremental neural network with SVM by Constantinides et al. [36]. Future research in labeling network data lies in devising more adaptive detection models and developing paradigms and techniques for efficient traffic data labeling [84, 199].

5.2.3 Consumer Network Intrusion. Specific to collecting data in a real-world network, the collection of data on *consumer networks* such as those at home, which are not armed with the same security resources as enterprise networks, lack datasets. Recently, Patel et al. [136] handled the natural entropy with detecting anomalies in a home network by collecting basic traffic features such as packet size, source, and destination ports and analyzing feature entropy. Further data collection in consumer networks has yet to be seen but is a viable route for research in the future.

5.2.4 Extending Anomaly Detection to Cloud Environments. Aside from speedup in model convergence or reducing anomaly detection time with cloud computing, exploring network intrusion in cloud environments has yet to be exhaustively researched. A hypervisor-based cloud network intrusion detection system based on statistical analytics was devised by Aldribi et al. [9], but more sophisticated attack methods have yet to be implemented, as Aldribi and others have noted the overtly regular pattern in the traffic data that was collected. Another trait of cloud environments now is that there is constantly changing data. Because a tremendous amount of data is stored on the cloud, looking to develop machine learning for dynamic data in the cloud should be a future step in research. Sethi et al. [154] applied a deep Q-learning reinforcement model to the cloud that is adaptable to changing data. Although there has been some work towards incorporating machine learning on dynamic data in the cloud, this is still nascent in terms of research and has potential to be studied further. Applying edge computing to cloud computing environments housing large amounts of network data is a potential route of research in the upcoming years to speed up detection time by bringing data storage and computation closer to the location where it is needed [68].

5.2.5 Machine Learning Scalability and Performance Improvements. Parallelism in big data machine learning models could help researchers improve *anomaly-based* intrusion detection methods as currently [82, 83, 85], an emphasis is made instead on signature-based techniques using CUDA. NID data and traffic is rapidly changing, and a natural approach to handling dynamic data is processing data in increments using incremental learning. Recently, Constantinides et al. [36] focused on scalability with incremental machine learning models. To handle the growth of their incremental self-organizing neural network commensurate with the growth of new data, a parameter n is used so any node that is nearest in Euclidean distance to more than n input vectors (more than n "wins") passes a "win" to the node with more than n "wins." The aging parameter in the network also removes nodes that are not updated to maintain a manageable size. With the dearth of scalability research, in the future, researchers should continue to study methods that enable incremental machine learning models to be more scalable in light of tremendous data growth.

6 CONCLUSIONS

Network intrusion detection has existed for a little over two decades when network resources were misused. Although most data-driven network intrusion systems have not been integrated

with an anomaly-based intrusion detection system on a large scale due to high false positive rates, researchers continue to improve anomaly detection accuracy and performance in the literature because of anomaly detection's ability to detect novel network attacks. This article introduces a general taxonomy on data-driven network intrusion detection methods based on a challenge-method heuristic and examines common public datasets used by papers in the taxonomy. Our focus is on the research trends gathered from the survey on network intrusion detection methods in the past decade. We conclude that, given the research trends over time, areas requiring future research are in big network data, streaming and changing data, and real-world network data collection and availability. Based on Table 3, only two of the papers that we investigated have fully reproducible code, so prospective papers can fill this gap in the research. Many solutions have been implemented for the other challenges specified in the taxonomy, but there remains a dearth of real-world network data, especially data on consumer networks, that could limit the accuracy of model performance in simulated network environments using real in-use network traffic data. This survey provides a high-level overview of the background on network intrusion detection, common datasets, a taxonomy of important research areas, and future directions.

REFERENCES

[1] Shaza Merghani Abdelrahman and Ajith Abraham. 2014. Intrusion detection using error correcting output code based ensemble. In *International Conference on Hybrid Intelligent Systems*. 181–186.

[2] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh. 2019. Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE Sensors Lett.* 3, 1 (2019), 1–4.

[3] Adebayo O. Adetunmbi, Samuel O. Falaki, Olumide S. Adewale, and Boniface K. Alese. 2008. Network intrusion detection based on rough set and k-nearest neighbour. *Int. J. Comput. ICT Res.* 2, 1 (2008), 60–66.

[4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. 2016. A survey of network anomaly detection techniques. *J. Netw. Comput. Applic.* 60 (2016), 19–31.

[5] Muhammad Ahsan, Muhammad Mashuri, Muhammad Hisyam Lee, Heri Kuswanto, and Dedy Dwi Prastyo. 2020. Robust adaptive multivariate Hotelling's T2 control chart based on kernel density estimation for intrusion detection system. *Exp. Syst. Applic.* 145 (2020), 113105.

[6] J. Aiken and S. Scott-Hayward. 2019. Investigating adversarial attacks against network intrusion detection systems in SDNs. In *IEEE Conference on Network Function Virtualization and Software Defined Networks*. 1–7.

[7] O. Y. Al-Jarrah, A. Siddiqui, M. Elsalamouny, P. D. Yoo, S. Muhaidat, and K. Kim. 2014. Machine-learning-based feature selection techniques for large-scale network intrusion detection. In *IEEE International Conference on Distributed Computing Systems Workshops*. 177–181.

[8] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi. 2018. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access* 6 (2018), 52843–52856.

[9] Abdulaziz Aldribi, Issa Traoré, Belaid Moa, and Onyekachi Nwamuo. 2020. Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. *Comput. Secur.* 88 (2020), 101646.

[10] Kevin Allix, Tegawendé François D. Assise Bissyande, Jacques Klein, and Yves Le Traon. 2014. *Machine Learning-based Malware Detection for Android Applications: History Matters!* Technical Report. University of Luxembourg, SnT.

[11] H. S. Alsaadi, R. Hedjam, A. Touzene, and A. Abdessalem. 2020. Fast binary network intrusion detection based on matched filter optimization. In *IEEE International Conference on Informatics, IoT, and Enabling Technologies*. 195–199.

[12] Kasun Amarasinghe, Kevin Kenney, and Milos Manic. 2018. Toward explainable deep neural network based anomaly detection. In *International Conference on Human System Interaction*. 311–317.

[13] Kasun Amarasinghe and Milos Manic. 2019. Explaining what a neural network has learned: Toward transparent classification. In *IEEE International Conference on Fuzzy Systems*. 1–6.

[14] A. An, C. Chan, N. Shan, N. Cercone, and W. Ziarko. 1997. Applying knowledge discovery to predict water-supply consumption. *IEEE Expert* 12, 4 (1997), 72–78.

[15] G. Apruzzese and M. Colajanni. 2018. Evading botnet detectors based on flows and random forest with adversarial samples. In *IEEE International Symposium on Network Computing and Applications*. 1–8.

[16] M. Azizjon, A. Jumabek, and W. Kim. 2020. 1D CNN based network intrusion detection with normalization on imbalanced data. In *International Conference on Artificial Intelligence in Information and Communication*. 218–224.

[17] Seyed Mojtaba Hosseini Bamakan, Huadong Wang, and Yong Shi. 2017. Ramp loss K-support vector classification-regression: A robust and sparse multi-class approach to the intrusion detection problem. *Knowl.-based Syst.* 126 (2017), 113–126.

[18] Seyed Mojtaba Hosseini Bamakan, Huadong Wang, Tian Yingjie, and Yong Shi. 2016. An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization. *Neurocomputing* 199 (2016), 90–102.
[19] Punam Bedi, Neha Gupta, and Vinita Jindal. 2020. Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Comput. Sci.* 171 (2020), 780–789.
[20] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. 2018. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Comput. Sci.* 127 (2018), 1–6.
[21] Hassina Bensefia and Nacira Ghoualmi. 2011. A new approach for adaptive intrusion detection. In *International Conference on Computational Intelligence and Security*. 983–987.
[22] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In *International Conference on Advances in Neural Information Processing Systems*. 737–744.
[23] A. L. Buczak and E. Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* 18, 2 (2016), 1153–1176.
[24] Mohamad Bydon, Clemens M. Schirmer, Eric K. Oermann, Ryan S. Kitagawa, Nader Pouratian, Jason Davies, Ashwini Sharan, and Lola B. Chambless. 2020. Big data defined: A practical review for neurosurgeons. *World Neurosurg.* 133 (2020), e842–e849.
[25] J. B. D. Caberera, B. Ravichandran, and R. K. Mehra. 2000. Statistical traffic modeling for network intrusion detection. In *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. 466–473.
[26] Pedro Casas, Johan Mazel, and Philippe Owezarski. 2012. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Comput. Commun.* 35, 7 (2012), 772–783.
[27] Chia-Mei Chen, Ya-Lin Chen, and Hsiao-Chung Lin. 2010. An efficient network intrusion detection. *Comput. Commun.* 33, 4 (2010), 477–484.
[28] Feilong Chen, Supranamaya Ranjan, and Pang-Ning Tan. 2011. Detecting bots via incremental LS-SVM learning with dynamic feature adaptation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[29] Jinyin Chen, Yi-tao Yang, Ke-ke Hu, Hai-bin Zheng, and Zhen Wang. 2019. DAD-MCNN: DDoS attack detection via multi-channel CNN. In *International Conference on Machine Learning and Computing*. 484–488.
[30] Meng-Hui Chen, Pei-Chann Chang, and Jheng-Long Wu. 2016. A population-based incremental learning approach with artificial immune system for network intrusion detection. *Eng. Applic. Artif. Intell.* 51 (2016), 171–181.
[31] T. Chen, X. Pan, Y. Xuan, J. Ma, and J. Jiang. 2010. A naive feature selection method and its application in network intrusion detection. In *International Conference on Computational Intelligence and Security*. 416–420.
[32] A. Cheng. 2019. PAC-GAN: Packet generation of network traffic using generative adversarial networks. In *IEEE Annual Information Technology, Electronics and Mobile Communication Conference*. 0728–0734.
[33] Zouhair Chiba, Noureddine Abghour, Khalid Moussaid, Amina El Omri, and Mohamed Rida. 2018. A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection. *Comput. Secur.* 75 (2018), 36–58.
[34] Yuk Ying Chung and Noorhaniza Wahid. 2012. A hybrid network intrusion detection system using simplified swarm optimization. *Appl. Soft Comput.* 12, 9 (2012), 3014–3022.
[35] Henry Clausen, Robert Flood, and David Aspinall. 2020. Traffic generation using containerization for machine learning. *arXiv preprint arXiv:2011.06350* (2020).
[36] Christos Constantinides, Stavros Shiaeles, Bogdan Ghita, and Nicholas Kolokotronis. 2019. A novel online incremental learning intrusion prevention system. In *IFIP International Conference on New Technologies, Mobility and Security*. 1–6.
[37] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. 2008. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE Symposium on Security and Privacy*. 81–95.
[38] Robertas Damasevicius, Algimantas Venckauskas, Sarunas Grigaliunas, Jevgenijus Toldinas, Nerijus Morkevicius, Tautvydas Aleliunas, and Paulius Smuikys. 2020. LITNET-2020: An annotated real-world network flow dataset for network intrusion detection. *Electronics* 9, 5 (2020).
[39] Jonathan J. Davis and Andrew J. Clark. 2011. Data preprocessing for anomaly based network intrusion detection: A review. *Comput. Secur.* 30, 6 (2011), 353–375.
[40] Knowledge Discovery and Data Mining. 1999. KDD Cup 1999: Computer Network Intrusion Detection. Retrieved from https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data.
[41] Abhishek Divekar, Meet Parekh, Vaibhav Savla, Rudra Mishra, and Mahesh Shirole. 2018. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In *IEEE International Conference on Computing, Communication and Security*. 1–8.

[42] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, and Pang-Ning Tan. 2002. Data mining for network intrusion detection. In *NSF Workshop on Next Generation Data Mining*. 21–30.
[43] L. Dongxia and Z. Yongbo. 2012. An intrusion detection system based on honeypot technology. In *International Conference on Computer Science and Electronics Engineering*. 451–454.
[44] Adel Sabry Eesa, Zeynep Orman, and Adnan Mohsin Abdulazeez Brifcani. 2015. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Exp. Syst. Applic.* 42, 5 (2015), 2670–2679.
[45] Wisam Elmasry, Akhan Akbulut, and Abdul Halim Zaim. 2020. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Comput. Netw.* 168 (2020), 107042.
[46] Osama Faker and Erdogan Dogdu. 2019. Intrusion detection using big data and deep learning techniques. In *ACM Southeast Conference*. 86–93.
[47] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá. 2018. Enabling an anatomic view to investigate honeypot systems: A survey. *IEEE Syst. J.* 12, 4 (2018), 3906–3919.
[48] Nabila Farnaaz and M. A. Jabbar. 2016. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* 89, 1 (2016), 213–217.
[49] Wenying Feng, Qinglei Zhang, Gongzhu Hu, and Jimmy Xiangji Huang. 2014. Mining network data for intrusion detection through combining SVMs with ant colony networks. *Fut. Gen. Comput. Syst.* 37 (2014), 127–140.
[50] Feng Xie, Hongyu Yang, Yong Peng, and Haihui Gao. 2012. Data fusion detection model based on SVM and evidence theory. In *IEEE International Conference on Communication Technology*. 814–818.
[51] Gabriel Macia Fernandez, Jose Camacho, Roberto Magan-Carri, Pedro Garcia-Teodoro, and Roberto Theron. 2016. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers and Security* 73 (2018), 411–424.
[52] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *ACM CoNEXT'10*.
[53] Canadian Institute for Cybersecurity. 2009. NSL-KDD Dataset. Retrieved from https://www.unb.ca/cic/datasets/nsl.html.
[54] Canadian Institute for Cybersecurity. 2017. Intrusion Detection Evaluation Dataset (CICIDS2017). Retrieved from https://www.unb.ca/cic/datasets/ids-2017.html.
[55] Canadian Institute for Cybersecurity. 2018. UNB CSE-CIC-IDS2018 on AWS. Retrieved from https://www.unb.ca/cic/datasets/ids-2018.html.
[56] Krishna Gade, Sahin Geyik, Krishnaram Kenthapadi, Varun Mithal, and Ankur Taly. 2020. Explainable AI in industry: Practical challenges and lessons learned. In *the Web Conference*. 303–304.
[57] S. M. Gaffer, M. E. Yahia, and K. Ragab. 2012. Genetic fuzzy system for intrusion detection: Analysis of improving of multiclass classification accuracy using KDDCup-99 imbalance dataset. In *International Conference on Hybrid Intelligent Systems*. 318–323.
[58] Sannasi Ganapathy, Kanagasabai Kulothungan, Sannasy Muthurajkumar, Muthusamy Vijayalakshmi, Palanichamy Yogesh, and Arputharaj Kannan. 2013. Intelligent feature selection and classification techniques for intrusion detection in networks: A survey. *EURASIP J. Wirel. Commun. Netw.* 2013, 1 (2013), 271.
[59] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu. 2019. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access* 7 (2019), 82512–82521.
[60] Y. Gao, Y. Liu, Y. Jin, J. Chen, and H. Wu. 2018. A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system. *IEEE Access* 6 (2018), 50927–50938.
[61] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* 28, 1–2 (2009), 18–28.
[62] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2009. Active learning for network intrusion detection. In *ACM Workshop on Security and Artificial Intelligence*. 47–54.
[63] Y. Gu, K. Li, Z. Guo, and Y. Wang. 2019. Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm. *IEEE Access* 7 (2019), 64351–64365.
[64] Y. Guo, B. Wang, X. Zhao, X. Xie, L. Lin, and Q. Zhou. 2010. Feature selection based on Rough set and modified genetic algorithm for intrusion detection. In *International Conference on Computer Science Education*. 1441–1446.
[65] Govind P. Gupta and Manish Kulariya. 2016. A framework for fast and efficient cyber security network intrusion detection using Apache Spark. *Procedia Comput. Sci.* 93 (2016), 824–831.
[66] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *J. Netw. Comput. Applic.* 87 (2017), 185–192.
[67] Bahram Hajimirzaei and Nima Jafari Navimipour. 2019. Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. *ICT Express* 5, 1 (2019), 56–59.
[68] Eric Hamilton. 2019. What is edge computing: The network edge explained. *Cloudw. Retr.* 3 (2019), 18–20.

[69] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren. 2019. A novel multimodal-sequential approach based on multi-view features for network intrusion detection. *IEEE Access* 7 (2019), 183207–183221.
[70] Chandima HewaNadungodage, Yuni Xia, and John Jaehwan Lee. 2016. GPU-accelerated outlier detection for continuous data streams. In *IEEE International Parallel and Distributed Processing Symposium*. 1133–1142.
[71] G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[72] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. 2011. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Exp. Syst. Applic.* 38, 1 (2011), 306–313.
[73] Y. Hsu, Z. He, Y. Tarutani, and M. Matsuoka. 2019. Toward an online network intrusion detection system based on ensemble learning. In *IEEE International Conference on Cloud Computing*. 174–178.
[74] Weiming Hu, Wei Hu, and Steve Maybank. 2008. Adaboost-based algorithm for network intrusion detection. *IEEE Trans. Syst., Man, Cybern.* 38, 2 (2008), 577–583.
[75] Shin-Ying Huang, Fang Yu, Rua-Huan Tsaih, and Yennun Huang. 2015. Network-traffic anomaly detection with incremental majority learning. In *International Joint Conference on Neural Networks*. 1–8.
[76] M. Ichino and J. Sklansky. 1984. Optimum feature selection by zero-one integer programming. *IEEE Trans. Syst., Man, Cybern.* SMC-14, 5 (1984), 737–746.
[77] Sheikh Rabiul Islam, William Eberle, Sheikh K. Ghafoor, Ambareen Siraj, and Mike Rogers. 2019. Domain knowledge aided explainable artificial intelligence for intrusion detection and response. *arXiv preprint arXiv:1911.09853* (2019).
[78] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. 2016. A deep learning approach for network intrusion detection system. In *EAI International Conference on Bio-inspired Information and Communications Technologies*. 21–26.
[79] H. J. Jeong, W. Hyun, J. Lim, and I. You. 2012. Anomaly teletraffic intrusion detection systems on Hadoop-based platforms: A survey of some problems and solutions. In *15th International Conference on Network-based Information Systems*. 766–770.
[80] H. Jiang, Z. He, G. Ye, and H. Zhang. 2020. Network intrusion detection based on PSO-Xgboost model. *IEEE Access* 8 (2020), 58392–58401.
[81] K. Jiang, W. Wang, A. Wang, and H. Wu. 2020. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access* 8 (2020), 32464–32476.
[82] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2016. Spotting suspicious behaviors in multimodal data: A general metric and algorithms. *IEEE Trans. Knowl. Data Eng* 28, 8 (2016), 2187–2200.
[83] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Catching synchronized behaviors in large networks: A graph mining approach. *ACM Trans. Knowl. Discov. Data* 10, 4 (2016), 1–27.
[84] Meng Jiang, Peng Cui, and Christos Faloutsos. 2016. Suspicious behavior detection: Current trends and future directions. *IEEE Intell. Syst.* 31, 1 (2016), 31–39.
[85] Meng Jiang, Christos Faloutsos, and Jiawei Han. 2016. CatchTartan: Representing and summarizing dynamic multicontextual behaviors. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. 945–954.
[86] X. Kang, B. Song, X. Du, and M. Guizani. 2020. Adversarial attacks for image segmentation on multiple lightweight models. *IEEE Access* 8 (2020), 31359–31370.
[87] Gurdip Kaur, Meenu Khurana, and Monika Sethi. 2011. Intrusion detection system using honeypots and swarm intelligence. In *International Conference on Advances in Computing and Artificial Intelligence*. 34–38.
[88] Nathan Keegan, Soo-Yeon Ji, Aastha Chaudhary, Claude Concolato, Byunggu Yu, and Dong Hyun Jeong. 2016 . A survey of cloud-based network intrusion detection analysis. *Hum.-centr. Comput. Inf. Sci.* 6 (2016), 19.
[89] Chaouki Khammassi and Saoussen Krichen. 2017. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput. Secur.* 70 (2017), 255–277.
[90] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain. 2019. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access* 7 (2019), 30373–30385.
[91] Jiyeon Kim, Jiwon Kim, Hyunjung Kim, Minsun Shim, and Eunjung Choi. 2020. CNN-based network intrusion detection against denial-of-service attacks. *Electronics* 9, 6 (2020).
[92] Levent Koc, Thomas A. Mazzuchi, and Shahram Sarkani. 2012. A network intrusion detection system based on a Hidden Naive Bayes multiclass classifier. *Exp. Syst. Applic.* 39, 18 (2012), 13492–13500.
[93] Eduardo De la Hoz, Emiro De La Hoz, Andrés Ortiz, Julio Ortega, and Beatriz Prieto. 2015. PCA filtering and probabilistic SOM for network intrusion detection. *Neurocomputing* 164 (2015), 71–81.
[94] MIT Lincoln Laboratory. 1998. 1998 Darpa intrusion detection evaluation dataset. Retrieved from https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset.
[95] Fukudu Labs. 2020. Mawilab. Retrieved from http://www.fukuda-lab.org/mawilab/documentation.html#labels.

[96] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *SIAM International Conference on Data Mining*. 25–36.
[97] T. A. Le, T. H. Chu, Q. U. Nguyen, and X. H. Nguyen. 2014. Malware detection using genetic programming. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 1–6.
[98] John Zhong Lei and Ali A. Ghorbani. 2012. Improved competitive learning neural networks for network intrusion and fraud detection. *Neurocomputing* 75, 1 (2012), 135–145.
[99] Hongda Li, Hongxin Hu, Guofei Gu, Gail-Joon Ahn, and Fuqiang Zhang. 2018. VNIDS: Towards elastic security with safe and efficient virtualization of network intrusion detection systems. In *ACM SIGSAC Conference on Computer and Communications Security*. 17–34.
[100] Peipei Li, Xindong Wu, Xuegang Hu, and Hao Wang. 2015. Learning concept-drifting data streams with random ensemble decision trees. *Neurocomputing* 166 (2015), 68–83.
[101] Y. Li, Z. Li, and R. Wang. 2011. Intrusion detection algorithm based on semi-supervised learning. In *International Conference of Information Technology, Computer Engineering and Management Sciences*. 153–156.
[102] Yanmiao Li, Yingying Xu, Zhi Liu, Haixia Hou, Yushuo Zheng, Yang Xin, Yuefeng Zhao, and Lizhen Cui. 2020. Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. *Measurement* 154 (2020), 107450.
[103] Jinping Liu, Jiezhou He, Wuxia Zhang, Tianyu Ma, Zhaohui Tang, Jean Paul Niyoyita, and Weihua Gui. 2019. ANID-SEoKELM: Adaptive network intrusion detection based on selective ensemble of kernel ELMs with random features. *Knowl.-based Syst.* 177 (2019), 104–116.
[104] Jinxin Liu, Burak Kantarci, and Carlisle Adams. 2020. Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In *ACM Workshop on Wireless Security and Machine Learning*.
[105] Jinping Liu, Wuxia Zhang, Zhaohui Tang, Yongfang Xie, Tianyu Ma, Jingjing Zhang, Guoyong Zhang, and Jean Paul Niyoyita. 2020. Adaptive intrusion detection via GA-GOGMM-based pattern learning with fuzzy rough set-based attribute selection. *Exp. Syst. Applic.* 139 (2020), 112845.
[106] Wei Liu, LinLin Ci, and LiPing Liu. 2020. A new method of fuzzy support vector machine algorithm for intrusion detection. *Appl. Sci.* 10 (2020).
[107] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. 2015. Transfer learning using computational intelligence: A survey. *Knowl.-based Syst.* 80 (2015), 14–23.
[108] Ma Yue, Lian Hong, and X. F. Zhang. 2010. Researches on the IPv6 network safeguard linked system. In *International Conference on Computer Science and Information Technology*. 387–390.
[109] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. 2018. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* 73 (2018).
[110] Roberto Magán-Carrión, Daniel Urda, Ignacio Díaz-Cano, and Bernabé Dorronsoro. 2020. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. *Appl. Sci.* 10, 5 (2020).
[111] Shraddha Mane and Dattaraj Rao. 2021. Explaining network intrusion detection system using explainable AI framework. *arXiv preprint arXiv:2103.07110* (2021).
[112] M. A. Manzoor and Y. Morgan. 2016. Real-time support vector machine based network intrusion detection system using Apache Storm. In *Annual Information Technology, Electronics and Mobile Communication Conference*. 1–5.
[113] Daniel L. Marino, Chathurika S. Wickramasinghe, and Milos Manic. 2018. An adversarial approach for explainable AI in intrusion detection systems. In *IECON Annual Conference of the IEEE Industrial Electronics Society*. 3237–3243.
[114] Nathan Martindale, Muhammad Ismail, and Douglas A. Talbert. 2020. Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data. *Information* 11, 6 (2020), 315.
[115] Maja Mataric. 1991. A comparative analysis of reinforcement learning methods. http://bitsavers.informatik.uni-stuttgart.de/pdf/mit/ai/aim/AIM-1322.pdf.
[116] Johan Mazel, Romain Fontugne, and Kensuke Fukuda. 2014. A taxonomy of anomalies in backbone network traffic. In *International Wireless Communications and Mobile Computing Conference*. 30–36.
[117] Joseph W. Mikhail, John M. Fossaceca, and Ronald Iammartino. 2019. A semi-boosted nested model with sensitivity-based weighted binarization for multi-domain network intrusion detection. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 3 (2019), 1–27.
[118] Robert Mitchell and Ing-Ray Chen. 2014. A survey of intrusion detection in wireless network applications. *Comput. Commun.* 42 (2014), 1–23.
[119] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. 2013. A survey of intrusion detection techniques in Cloud. *J. Netw. Comput. Applic.* 36, 1 (2013), 42–57.
[120] Sara Mohammadi, Hamid Mirvaziri, Mostafa Ghazizadeh-Ahsaee, and Hadis Karimipour. 2019. Cyber intrusion detection by combined feature selection algorithm. *J. Inf. Secur. Applic.* 44 (2019), 80–88.
[121] Valerio Morfino and Salvatore Rampone. 2020. Towards near-real-time intrusion detection for IoT devices using supervised learning and Apache Spark. *Electronics* 9, 3 (2020).

[122] N. Moustafa and J. Slay. 2015. The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems. In *International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. 25–31.
[123] N. Moustafa and J. Slay. 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems. In *Military Communications and Information Systems Conference*. 1–6.
[124] Biswanath Mukherjee, L. Todd Heberlein, and Karl N. Levitt. 1994. Network intrusion detection. *IEEE Netw.* 8, 3 (1994), 26–41.
[125] Saurabh Mukherjee and Neelam Sharma. 2012. Intrusion detection using naive Bayes classifier with feature reduction. *Procedia Technol.* 4 (2012), 119–128.
[126] R. Newman. 2009. *Computer Security: Protecting Digital Resources*. Jones and Bartlett Publishers.
[127] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. GEE: A gradient-based explainable variational autoencoder for network anomaly detection. In *IEEE Conference on Communications and Network Security*. 91–99.
[128] Fakhroddin Noorbehbahani, Ali Fanian, Rasoul Mousavi, and Homa Hasannejad. 2017. An incremental intrusion detection system using a new semi-supervised stream classification method. *Int. J. Commun. Syst.* 30, 4 (2017), e3002.
[129] Stephen Northcutt and Judy Novak. 2002. *Network Intrusion Detection*. Sams Publishing.
[130] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. 2017. Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security*. 361–369.
[131] The University of New South Wales. 2015. The UNSW-NB15 Dataset Description. Retrieved from https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.
[132] Kaunas University of Technology. 2020. LITNET-2020: An Annotated Real-world Network Flows Dataset for Network Intrusion Detection. Retrieved from https://dataset.litnet.lt/.
[133] Takwa Omrani, Adel Dallali, Belgacem Chibani Rhaimi, and Jaouhar Fattahi. 2017. Fusion of ANN and SVM classifiers for network attack detection. In *International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*. 374–377.
[134] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. 2012. A hybrid intelligent approach for network intrusion detection. *Procedia Eng.* 30 (2012), 1–9.
[135] Mrutyunjaya Panda and Manas Ranjan Patra. 2007. Network intrusion detection using naive bayes. *Int. J. Comput. Sci. Netw. Secur.* 7, 12 (2007), 258–263.
[136] Darsh Patel, Kathiravan Srinivasan, Chuan-Yu Chang, Takshi Gupta, and Aman Kataria. 2020. Network anomaly detection inside consumer networks–A hybrid approach. *Electronics* 9, 6 (2020), 923.
[137] Y. Peng, J. Su, X. Shi, and B. Zhao. 2019. Evaluating deep learning based network intrusion detection system in adversarial environment. In *IEEE International Conference on Electronics Information and Emergency Communication*.
[138] Robi Polikar, Lalita Udpa, Satish Udpa, and Vasant Honavar. 2004. An incremental learning algorithm with confidence estimation for automated identification of NDE signals. *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.* 51, 8 (2004), 990–1001.
[139] H. E. Poston. 2012. A brief taxonomy of intrusion detection strategies. In *IEEE National Aerospace and Electronics Conference*. 255–263.
[140] Mahendra Prasad, Sachin Tripathi, and Keshav Dahal. 2020. An efficient feature selection based Bayesian and Rough set approach for intrusion detection. *Appl. Soft Comput.* 87 (2020), 105980.
[141] M. R. Gauthama Raman, Kannan Kirthivasan, and V. S. Shankar Sriram. 2017. Development of rough set–hypergraph technique for key feature identification in intrusion detection systems. *Comput. Electric. Eng.* 59 (2017), 189–200.
[142] N. Ravi and S. M. Shalinie. 2020. Semi-supervised learning based security to detect and mitigate intrusions in IoT network. *IEEE Internet Things J.* 7, 11 (2020), 11041–11052.
[143] Paulo Angelo Alves Resende and André Costa Drummond. 2018. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.* 51, 3 (2018).
[144] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-based network traffic generation using generative adversarial networks. *Comput. Secur.* 82 (2019), 156–172.
[145] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. 2017. Creation of flow-based data sets for intrusion detection. *J. Inf. Warf.* 16, 4 (2017), 40–53.
[146] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. 2017. Flow-based benchmark data sets for intrusion detection. In *European Conference on Cyber Warfare and Security*. 361–369.
[147] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. 2019. A survey of network-based intrusion detection data sets. *Comput. Secur.* 86 (2019), 147–167.
[148] A. Sahu, Z. Mao, K. Davis, and A. E. Goulart. 2020. Data processing and model selection for machine learning-based network intrusion detection. In *IEEE International Workshop Technical Committee on Communications Quality and Reliability*. 1–6.

[149] Roberto Saia, Salvatore Carta, Diego Reforgiato Recupero, Gianni Fenu, and Maria Madalina Stanciu. 2019. A discretized extended feature space (DEFS) model to improve the anomaly detection performance in network intrusion detection systems. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 322–329.

[150] Fadi Salo, Ali Bou Nassif, and Aleksander Essex. 2019. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Comput. Netw.* 148 (2019), 164–175.

[151] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. 2021. An explainable machine learning-based network intrusion detection system for enabling generalisability in securing IoT networks. *arXiv preprint arXiv:2104.07183* (2021).

[152] Martin Sarnovsky and Jan Paralic. 2020. Hierarchical intrusion detection using machine learning and knowledge model. *Symmetry* 12, 2 (2020), 203.

[153] K. Selvakumar, Marimuthu Karuppiah, L. SaiRamesh, S. K. Hafizul Islam, Mohammad Mehedi Hassan, Giancarlo Fortino, and Kim-Kwang Raymond Choo. 2019. Intelligent temporal classification and fuzzy rough set-based feature selection algorithm for intrusion detection system in WSNs. *Inf. Sci.* 497 (2019), 77–90.

[154] Kamalakanta Sethi, Rahul Kumar, Nishant Prajapati, and Padmalochan Bera. 2020. Deep reinforcement learning based intrusion detection system for cloud infrastructure. In *International Conference on COMmunication Systems & NETworkS*. 1–6.

[155] A. Shafee, M. Baza, D. A. Talbert, M. M. Fouda, M. Nabil, and M. Mahmoud. 2020. Mimic learning to generate a shareable network intrusion detection model. In *IEEE Annual Consumer Communications Networking Conference*. 1–6.

[156] Shahaboddin Shamshirband, Amineh Amini, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ying Wah Teh, and Steven Furnell. 2014. D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. *Measurement* 55 (2014), 212–226.

[157] Z. Shi, J. Li, and C. Wu. 2019. DeepDDoS: Online DDoS attack detection. In *IEEE Global Communications Conference*. 1–6.

[158] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi. 2018. A deep learning approach to network intrusion detection. *IEEE Trans. Emerg. Top. Comput. Intell.* 2, 1 (2018), 41–50.

[159] W. Shuyue, Y. Jie, and F. Xiaoping. 2011. Research on intrusion detection method based on SVM co-training. In *International Conference on Intelligent Computation Technology and Automation*. 668–671.

[160] Kamran Siddique, Zahid Akhtar, Farrukh Aslam Khan, and Yangwoo Kim. 2019. KDD Cup 99 data sets: A perspective on the role of data sets in network intrusion detection research. *Computer* 52, 2 (2019), 41–51.

[161] A. Singla, E. Bertino, and D. Verma. 2019. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In *IEEE International Conference on Smart Computing*. 69–74.

[162] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*. 305–316.

[163] Tongtong Su, Huazhi Sun, Jinqi Zhu, Sheng Wang, and Yabo Li. 2020. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access* 8 (2020), 29575–29585.

[164] P. Kola Sujatha, C. Suba Priya, and A. Kannan. 2012. Network intrusion detection system using genetic network programming with support vector machine. In *International Conference on Advances in Computing, Communications and Informatics*. 645–649.

[165] Mateusz Szczepański, Michał Choraś, Marek Pawlicki, and Rafał Kozik. 2020. Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In *International Joint Conference on Neural Networks*. 1–8.

[166] Z. Tan, A. Jamdagni, X. He, and P. Nanda. 2010. Network intrusion detection based on LDA for payload feature selection. In *IEEE Globecom Workshops*. 1545–1549.

[167] Bo Tang and Haibo He. 2017. A local density-based approach for outlier detection. *Neurocomputing* 241 (2017), 171–180.

[168] Shahroz Tariq, Sangyup Lee, and Simon S. Woo. 2020. CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network. In *Annual ACM Symposium on Applied Computing*. 1048–1055.

[169] Yogita Thakran and Durga Toshniwal. 2012. Unsupervised outlier detection in streaming data using weighted clustering. In *International Conference on Intelligent Systems Design and Applications*. 947–952.

[170] I. S. Thaseen and C. A. Kumar. 2016. An integrated intrusion detection model using consistency based feature selection and LPBoost. In *International Conference on Green Engineering and Technologies*. 1–6.

[171] M. Thottan and Chuanyi Ji. 2003. Anomaly detection in IP networks. *IEEE Trans. Sig. Process.* 51, 8 (2003), 2191–2204.

[172] M. Usama, M. Asim, S. Latif, J. Qadir, and Ala-Al-Fuqaha. 2019. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *International Wireless Communications Mobile Computing Conference*. 78–83.

[173] K. Keerthi Vasan and B. Surendiran. 2016. Dimensionality reduction using principal component analysis for network intrusion detection. *Perspect. Sci.* 8 (2016), 510–512.
[174] Cheng-Ru Wang, Rong-Fang Xu, Shie-Jue Lee, and Chie-Hong Lee. 2018. Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowl.-based Syst.* 147 (2018), 68–80.
[175] Maonan Wang, Kangfeng Zheng, Yanqing Yang, and Xiujuan Wang. 2020. An explainable machine learning framework for intrusion detection systems. *IEEE Access* 8 (2020), 73127–73141.
[176] Quanmin Wang and Xuan Wei. 2020. The detection of network intrusion based on improved Adaboost algorithm. In *International Conference on Cryptography, Security and Privacy*. 84–88.
[177] Wei Wang, Thomas Guyet, René Quiniou, Marie-Odile Cordier, Florent Masseglia, and Xiangliang Zhang. 2014. Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks. *Knowl.-based Syst.* 70 (2014), 103–117.
[178] W. Wong, H. Chen, C. Hsu, and T. Chao. 2011. Reinforcement learning of robotic motion with genetic programming, simulated annealing and self-organizing map. In *International Conference on Technologies and Applications of Artificial Intelligence*. 292–298.
[179] Binhan Xu, Shuyu Chen, Hancui Zhang, and Tianshu Wu. 2017. Incremental k-NN SVM method in intrusion detection. In *IEEE International Conference on Software Engineering and Service Science*. 712–717.
[180] C. Xu, J. Shen, and X. Du. 2020. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans. Inf. Forens. Secur.* 15 (2020), 3540–3552.
[181] C. Xu, J. Shen, X. Du, and F. Zhang. 2018. An intrusion detection system using a deep neural network with gated recurrent units. *IEEE Access* 6 (2018), 48697–48707.
[182] S. Xu, Y. Qian, and R. Q. Hu. 2019. Data-driven edge intelligence for robust network anomaly detection. *IEEE Trans. Netw. Sci. Eng.* 7, 3 (2019), 1481–1492.
[183] H. Yang and F. Wang. 2019. Wireless network intrusion detection based on improved convolutional neural network. *IEEE Access* 7 (2019), 64366–64374.
[184] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang. 2020. Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. *IEEE Access* 8 (2020).
[185] Yang Yi, Jiansheng Wu, and Wei Xu. 2011. Incremental SVM based on reserved set for network intrusion detection. *Exp. Syst. Applic.* 38, 6 (2011), 7698–7707.
[186] Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D. Cubuk, and Justin Gilmer. 2019. A Fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988* (2019).
[187] Ying Wang, Yongjun Shen, and Guidong Zhang. 2016. Research on intrusion detection model using ensemble learning methods. In *IEEE International Conference on Software Engineering and Service Science*. 422–425.
[188] S. Youm, Y. Kim, K. Shin, and E. Kim. 2020. An authorized access attack detection method for realtime intrusion detection system. In *IEEE Annual Consumer Communications Networking Conference*. 1–6.
[189] D. YuanTong. 2019. Research of intrusion detection method based on IL-FSVM. In *Joint International Information Technology and Artificial Intelligence Conference*. 1221–1225.
[190] F. Zhang and D. Wang. 2013. An effective feature selection approach for network intrusion detection. In *IEEE Eighth International Conference on Networking, Architecture and Storage*. 307–311.
[191] Hongpo Zhang, Lulu Huang, Chase Q. Wu, and Zhanbo Li. 2020. An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. *Comput. Netw.* 177 (2020).
[192] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque. 2008. Random-forests-based network intrusion detection systems. *IEEE Trans. Syst., Man, Cybern.* 38, 5 (2008), 649–659.
[193] Wenhao Zhang, Ramin Ramezani, and Arash Naeim. 2019. WOTBoost: Weighted oversampling technique in boosting for imbalanced learning. In *IEEE International Conference on Big Data*. 2523–2531.
[194] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang. 2019. PCCN: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows. *IEEE Access* 7 (2019), 119904–119916.
[195] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo. 2019. Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access* 7 (2019), 37004–37016.
[196] Y. Zhang, Q. Yang, S. Lambotharan, K. Kyriakopoulos, I. Ghafir, and B. AsSadhan. 2019. Anomaly-based network intrusion detection using SVM. In *International Conference on Wireless Communications and Signal Processing*. 1–6.
[197] J. Zhao, S. Shetty, and J. W. Pan. 2017. Feature-based transfer learning for network security. In *IEEE Military Communications Conference*. 17–22.
[198] Juan Zhao, Sachin Shetty, Jan Wei Pan, Charles Kamhoua, and Kevin Kwiat. 2019. Transfer learning for detecting unknown network attacks. *EURASIP J. Inf. Secur.* 2019, 1 (2019), 1.
[199] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. 2020. Error-bounded graph anomaly loss for GNNs. In *ACM International Conference on Information and Knowledge Management (CIKM'20)*.

[200] Ming Zheng, Tong Li, Rui Zhu, Yahui Tang, Mingjing Tang, Leilei Lin, and Zifei Ma. 2020. Conditional Wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. *Inf. Sci.* 512 (2020), 1009–1023.

[201] Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, and Keqin Li. 2020. HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning. *Comput. Netw.* 169 (2020), 107049.

[202] Yingying Zhu, Junwei Liang, Jianyong Chen, and Zhong Ming. 2017. An improved NSGA-III algorithm for feature selection used in intrusion detection. *Knowl.-based Syst.* 116 (2017), 74–85.

[203] Wei Zong, Yang-Wai Chow, and Willy Susilo. 2020. Interactive three-dimensional visualization of network intrusion detection data for machine learning. *Fut. Gen. Comput. Syst.* 102 (2020), 292–306.

Received September 2020; revised May 2021; accepted June 2021