IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 10, 2021, accepted March 22, 2021, date of publication April 22, 2021, date of current version May 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3075066

Machine Learning for Misuse-Based Network Intrusion Detection: Overview, Unified Evaluation and Feature Choice Comparison Framework

LAURENS LE JEUNE[1,2], TOON GOEDEMÉ[2], AND NELE MENTENS[1,3], (Senior Member, IEEE)

[1]ES&S and imec-COSIC, Department of Electrical Engineering (ESAT), KU Leuven, 3000 Leuven, Belgium
[2]EAVISE, PSI, Department of Electrical Engineering (ESAT), KU Leuven, 3000 Leuven, Belgium
[3]Leiden Institute of Advanced Computer Science (LIACS), Leiden University, 2311 Leiden, The Netherlands

Corresponding author: Laurens Le Jeune (laurens.lejeune@kuleuven.be)

This work was supported in part by the COllective Research NETworking (CORNET) and funded by VLAIO under Grant HBC.2018.0491, and in part by the CyberSecurity Research Flanders under Grant VR20192203.

ABSTRACT Network Intrusion detection systems are essential for the protection of advanced communication networks. Originally, these systems were hard-coded to identify specific signatures, patterns and rule violations; now artificial intelligence and machine learning algorithms provide promising alternatives. However, in the literature, various outdated datasets as well as a plethora of different evaluation metrics are used to prove algorithm efficacy. To enable a global comparison, this study compiles algorithms for different configurations to create common ground and proposes two new evaluation metrics. These metrics, the detection score and the identification score, together reliably present the performance of a network intrusion detection system to allow for practical comparison on a large scale. Additionally, we present a workflow to process raw packet flows into input features for machine learning. This framework quickly implements different algorithms for the various datasets and allows systematic performance comparison between those algorithms. Our experimental results, matching and surpassing the state-of-the-art, indicate the potential of this approach. As raw traffic input features are much easier and cheaper to extract when compared to traditional features, they show promise for application in real-time deep learning-based systems.

INDEX TERMS Intrusion detection, machine learning, neural networks, security.

I. INTRODUCTION

Today, more and more devices are connected to the internet. Cisco forecasts that by 2023 there will be 29.3 billion devices connected to the internet [1]. As the attack surface increases, the need for security rises. For example, in 2015 a massive *brute force* attack [2] on Alibaba resulted in the potential compromisation of 21 million user accounts. In 2016, Internet-of-Things (IoT) devices infected with the *Mirai* botnet were used in a large Distributed Denial of Service (DDoS) attack against Domain Name System provider Dyn, resulting in the unavailability of many major internet platforms such as Spotify, Twitter and Netflix[1] [3].

One important link in the chain of protection against attacks is the intrusion detection system (IDS), responsible for identifying malicious activity and attacks [4]. Network intrusion detection systems then aim to detect attacks by investigating network traffic. While they historically functioned through hard-coded rules, more and more research is being conducted to investigate the application of machine learning (ML). Academic research proposes many different network intrusion detection techniques, also comparing against other techniques. However, the plethora of publicly available and potentially outdated datasets, the differences between those datasets, the variety of evaluation methods and the occasional unclear reporting of proposed techniques significantly complicate making a fair comparison. This paper aims at solving this issue and sets out a work flow that is used to run existing solutions on relevant datasets and that is made open source such that it can easily be applied to future solutions. Our contribution is four-fold:

- We give an overview of the most frequently used datasets and we summarize the pros and cons of each dataset (Sect. IV).

[1]https://splinternews.com/here-are-the-sites-you-cant-access-because-someone-took-1793863079

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

- We present existing evaluation methods and their drawbacks, and we propose newly derived unifying metrics for fair and reliable comparison of network intrusion detection performance (Sect. V).
- We provide a profound overview of ML techniques in the literature for network intrusion detection, with a focus on recent deep learning (DL) approaches, and we quantitatively compare and discuss these techniques based on results reported in related work as well as our own recalculations (Sect. VI) using our proposed metrics.
- We propose a workflow that allows for the use of raw network traffic in machine learning, as raw traffic-based features are more suitable for real-time application when compared to traditional machine learning features for network intrusion detection (Sect. VII). The promising experimental results for various datasets and algorithms are comparable to the state-of-the-art.

Before expounding on these contributions, we will first provide background information on intrusion detection systems (Sect. II) as well as compare our work against other, related work (Sect. III).

II. BACKGROUND

Intrusion detection systems are systems that are able to detect malicious behaviour. This section inspects network intrusion detection as one of multiple intrusion detection applications, as well as different approaches to actually build a network intrusion detection system.

A. INTRUSION DETECTION SYSTEMS

One of the first mentions of detecting malicious activity was made in 1980 by J.P. Anderson [5], who outlines the required components for what is now known as an IDS. In 1987 D. Denning introduced IDES (Intrusion-Detection Expert System), which was the foundation for many subsequent IDSs [6]. Currently, various IDSs are used for various goals. Host-based intrusion detection systems (HIDS), for example, aim to detect intrusions in a specific host [4], such as a computer or a server. Notably, this not only comprises network-based intrusions, but includes unauthorized use of the host. A HIDS example is given in [7], in which the anomalous use of applications is detected by monitoring system calls. Although this technique helps to identify intrusions via a specific host, it provides no insight into intrusions in other parts of the network.

By contrast, network-based IDSs (NIDS) detect network traffic intrusions [4]. Rather than keeping track of one host, NIDSs monitor network attacks, by inspecting network traffic to detect malicious communication through the flows and generated features of the network packets.

Collaborative IDSs (CIDS) consist of multiple IDS nodes that exchange information in a centralized, decentralized or distributed manner [8]. This way, a large network can be better protected against exceedingly distributed attacks [9]. One of the earlier mentions of such a system is given in [10]. Building what effectively is an intrusion detection network (IDN) allows the different IDS nodes to communicate and pass on useful information. This however unlocks a new security risk: malicious or compromised IDS nodes can try to give false feedback, or forgo giving necessary feedback. Therefore, CIDS research involves verifying the trustworthiness of a node, e.g. [9], [11], [12].

Recently, a new branch of IDS research has emerged, investigating the security of the Internet-of-Things (IoT). IoT nodes require IDS technology different from regular networks, for three reasons: The limited resources of IoT nodes, their specific network topologies and their new communication protocols [13]. Some examples are [14], [15].

Another increasingly important domain concerns Wireless Sensor Networks (WSN). These networks are characterized by infrastructure absence, wireless links, limited physical protection, a lack of central management and limited resources as defined in [16]. And while some WSN solutions are being evaluated on NIDS datasets [17]–[20], WSN environments are intrinsically different from the traditional NIDS environment we consider in this paper, with a fixed, wired infrastructure and abundant resources.

In practice, IDS implementations can be combinations of the different types of IDSs. Reference [11] for example uses different HIDS nodes in a CIDS system, where the HIDS nodes communicate to improve detection accuracy. IDSs that combine HIDS and NIDS technology are sometimes called hybrid IDS [21], [22]. Note that *hybrid* IDS can also denote an IDS that combines misuse-based and anomaly-based intrusion detection [14], [23], [24]. This will be more thoroughly investigated in Sect. II-B. In this paper, we focus on network-based intrusion detection systems.

B. MISUSE-BASED OR ANOMALY-BASED

Generally, the functioning of any IDS can be described as being either misuse-based or anomaly-based. Misuse-based[2] intrusion detection, also known as knowledge-based [4], in principle simply means that the IDS knows what certain attacks look like, and that it detects attacks based on that knowledge. Therefore, misuse-based intrusion detection algorithms obtain low false positive rates (see Sect. V) when inspecting network traffic. Moreover, they can effectively detect known attacks and label them accordingly, which facilitates following up on the detection. There is however one glaring weakness of misuse-based intrusion detection systems: Their inability to detect unknown and zero-day attacks. Since they are conditioned to detect known attacks, other attacks that do not share similarities with them will go unnoticed.

The counterpart to the misuse-based intrusion detection is anomaly-based intrusion detection. Anomaly-based or behaviour-based intrusion detection creates a model of what

[2]Traditionally, misuse-based approaches were signature-based, matching traffic against know attack patterns. In this paper, we include supervised machine learning solutions in this category, as they are trained to recognize specific attacks.

is supposed to be *normal* network traffic. Intrusions can be defined as traffic with significant deviations from that expected behaviour. This means that rather than detecting very specific attacks, anomaly-based intrusion detection catches abnormal traffic. Because abnormal traffic does not automatically correspond to an attack, anomaly-based intrusion detection is characterized by high false positive rates. As anomaly-based intrusion detection systems are able to detect unknown and zero-day attacks however, they remain relevant for research.

Note that in some literature, (network) anomaly detection is a term used to designate both misuse-based as well as behaviour-based intrusion detection. It is therefore advisable to always verify what exactly is meant by *anomaly detection*. In this paper we mainly concentrate on misuse-based detection, although our metrics can also be applied to anomaly-based techniques.

III. RELATED WORK

Since much research has gone into network intrusion detection, many different techniques and algorithms have been proposed over the years. As a result, there is a need to categorize and compare different approaches to get a better understanding of the field. In this section, we go over some survey papers, ordered by years, in order to investigate what exists and to state our contribution. The focus in this regard lies on more recent work, as machine learning and especially deep learning are fast evolving research fields.

Bhuyan *et al.* [25] present a categorization of network anomaly detection methods and systems, encompassing statistical, classification-based, knowledge-based, soft-computing, clustering-based, ensemble-based, fusion-based and hybrid approaches. Additionally, they consider several tools such as *nmap* or *Wireshark* that are useful in network anomaly detection, and they discuss the evaluation of detection systems. Finally, they conclude by providing recommendations for and stating challenges of network anomaly detection.

Ahmed *et al.* [26] consider different methods for anomaly detection, namely classification, statistical, information theory-based and clustering-based approaches. Note that their anomaly detection techniques also include misuse-based algorithms such as Support Vector Machines and rule-based approaches. Additionally, they discuss IDS datasets, and evaluate the anomaly detection methods according to their computational complexity, their output format and their attack priority. However, as this evaluation appears to be limited to DARPA/KDDCup attacks, its applicability with regards to more recent datasets is limited.

In [27], Buczak *et al.* examine machine learning and data mining techniques for intrusion detection. These techniques include artificial neural networks, (fuzzy) association rules, Bayesian Networks, clustering, decision trees, ensemble learning, evolutionary computation, hidden Markov models, inductive learning, naive Bayes, sequential pattern mining and support vector machines. For each of these techniques, they consider both misuse-based detection as well as anomaly-detection. Next, they provide an overview of the computational complexity and streaming capabilities of each technique. By commenting on IDS performance, on the difficulty of comparing different detection methods and on the (re)trainability of models, as well as by proving some recommendations, they conclude their work.

In [28], the authors provide an overview of machine learning and deep learning techniques that are being used in cybersecurity, with a focus on network intrusion detection. They consider Support Vector Machines, k-Nearest Neighbour, Decision Trees, Deep Belief Networks, Recurrent Neural Networks and finally Convolutional Neural Networks. In this overview, they observe three problems: The relative lack of benchmark datasets, the non-uniformity of evaluation metrics leading to difficult comparison and finally the insufficient attention to algorithm efficiency. Moreover, they also remark some trends in intrusion detection research, namely the study of hybrid models, the opportunities and challenges deep learning poses, the increased number of papers comparing different algorithms as well as their practicability, and the promise for new benchmark datasets.

Boutaba *et al.* [23] present a survey on the use of machine learning for different networking aspects, among which they include network security. Distinguishing between misuse-based, anomaly-based, Deep and Reinforcement Learning-based and hybrid intrusion detection, they consider 36 approaches, mainly evaluated on the KDDCup1999 and the NSL-KDD datasets. Overall, they observe a need for more recent datasets, the lack of anomaly-based detection systems in real implementations, insufficient real-time implementations and a general lack of systems fulfilling other specific requirements. Finally, they conclude with a perspective of ML for networking in general. Interestingly, they also denote a need for real-world data instead of synthetic datasets as well as a need for standard evaluation metrics to accommodate easier comparison.

Berman *et al.* [29] present a survey portraying the application of deep learning in different cybersecurity problems, among which network intrusion detection is covered. In their overview of network intrusion detection, they consider recurrent neural networks, convolutional neural networks, deep neural networks, deep belief networks and autoencoders and other algorithms. They remark that, among all security problems, restricted Boltzmann machines, autoencoders and recurrent neural networks comprised the most popular approaches. Moreover, they state that the intrusion detection ability of a system depends strongly on both the number of classes as well as the kind of attack, in addition to being subject to the benign-malicious ratio in the training data. Finally, they consider the impact of false alarms as well as missed attacks, and draw attention to the fact that adversaries might try to actively circumvent protection measures.

Mahdavifar *et al.* [30] investigate the application of deep learning for a number of cybersecurity tasks, namely for malware detection, intrusion detection, phishing detection, spam

detection and website defacement detection. For intrusion detection, they mainly examine papers that employ generative deep learning technology, as opposed to discriminative or hybrid deep learning approaches. Based on the surveyed papers, the authors also present a generic framework showing the use of deep learning for cybersecurity. Moreover, they note the potential of semi-supervised learning,[3] as the field presents lots of such unlabelled data. Besides some other learning-related remarks, they finally conclude by pointing out that that deep learning should only be used in fields where complex non-linear models are required, if sufficient data is available.

Chaabouni *et al.* [31] present an overview of NIDS-based IoT security, considering IoT threats, public datasets and tools as well as current open-source NIDSs. Additionally, they also consider machine learning-based approaches which have been validated for general network intrusion detection datasets. One of their major remarks is the need for an IoT IDS dataset that is representative of the real world, with more attention to the semantic relation between detection performance and the learning process.

In [32], Ring *et al.* provide a substantial overview of network intrusion detection datasets, focussing on 15 properties in the analysis. Their work can serve as a guideline in selecting suitable public datasets for a specific goal, as they provide both a simplified overview as well as a large in-depth table and discussion. Moreover, they also consider other sources for data, namely data repositories and traffic generators. Finally, they draw some conclusions that are relevant for any other research involving NIDS datasets. For example, they discuss the unlikeliness of ever creating a perfect dataset, and recommend using multiple datasets for evaluation. Thus, while not providing insight regarding performance of specific algorithms or approaches, their investigation does yield pertinent knowledge.

Ferrag *et al.* [33] provide insight in deep learning techniques for cybersecurity purposes. They discuss 35 public datasets, classifying them into one of seven categories: Network traffic-based, electrical network-based, internet traffic-based, virtual private network-based, android apps-based, IoT traffic-based and internet-connected devices-based datasets. Besides considering existing deep learning-based intrusion detection algorithms, they also train and evaluate several deep learning models for the CSE-CIC-IDS2018 and the Bot-IoT datasets. The detection performance for these deep neural network, recurrent neural network, convolutional neural network, restricted Boltzmann machine, deep belief network, deep Boltzmann machine and deep autoencoder models appears to be comparable.

In comparison to other surveys, in this paper we focus on a number of novel things. Firstly, we tackle the issue of comparing between different approaches, potentially across different datasets, which is a challenge in network intrusion detection research [23], [27], [28], [34]. We introduce two metrics that can be calculated using standard measures and allow for fairer and easier comparison between datasets. Moreover, we also study the application of machine learning-based network intrusion detection in constrained, real-time environments, comparing different techniques for different datasets. We further propose a workflow to generate features in such a setting, and run experiments to validate this workflow as well as to compare against the state-of-the-art.

[3]In semi-supervised learning, the training set comprises a small subset of labelled samples and a large subset of unlabelled samples.

IV. DATASETS

Whenever developing machine learning algorithms, it is paramount to have a dataset at one's disposal. This dataset allows for training an algorithm and/or for evaluating the performance of that algorithm. In this section we discuss the publicly available data sets that are relevant for our research, as also listed in Table 1. These datasets usually contain both feature files as well as raw traffic files. While the feature files provide manually selected features and labelling, the raw traffic files provide the unlabelled binary traffic packets from which those features were extracted. Generally, the format of such raw traffic files is PCAP (Packet CAPture) or PCAPNG (PCAP Next Generation).

A. DARPA1998

One of the oldest important intrusion detection datasets is the 1998 DARPA Intrusion Detection Evaluation Dataset (DARPA1998) [35], [36] generated by MIT Lincoln Laboratory. For this dataset, the network traffic of over 50 air force bases during 4 months was examined and then recreated in a simulation. By including attacks corresponding to the DoS, R2L, U2R and Probe categories, the dataset simulates intrusion in the network. Concretely, 7 weeks of simulation were used to generate training data while the final 2 weeks provided test data. Interestingly, these test data contain some attack types (for example *mailbomb*, *UDP Storm* or *httptunnel*) that are not present in the training data. This allows for testing whether the NIDS can detect attacks it has not encountered before. This dataset has been criticized however, as John McHugh already did in 2000 [37]. While somewhat outdated in some regards (mentioned traffic flows are much smaller than today), the critiques remain relevant. One critique is that not enough proof is given that testing on the synthesised data is representative of the result in real-world traffic. Other critiques of McHugh include the taxonomy and distribution of the attacks or the analysis of the results. Moreover, [38] detected simulation artifacts in multiple attributes of network traffic, such as too few different TTL values, high regularity for TCP SYN packets or source address predictability. Additionally, [39] indicate even more issues in the DARPA1998 dataset. Besides once again denoting that the synthetic origin of the data can prove to be a problem, they argue that *tcpdump* (the traffic collector used) might drop packets in heavy traffic. As no examination of dropped packets was undertaken, it is impossible to account for these packets. Furthermore, they point out the lack of

exact definitions of the attacks. Not every probing action of a host can immediately be seen as an attack, for example. However, in DARPA1998, there are no clear specifications of when for example a probe becomes an attack [39]. Although being the first real intrusion detection dataset, DARPA1998 is often used or referred to, in combination with its successor KDDCup1999.

B. KDDCup 1999

The KDDCup1999 dataset (Knowledge Discovery and Data Mining Tools Competition) [40], or KDD99 for short, was created by taking a version of DARPA1998. This version was then used as the dataset for the KDD99 competition. KDD99 is a very important benchmark in the intrusion detection community, still being used for the evaluation of algorithms in 2019 [41]. This highlights one of the most important critiques on KDD99 nowadays: It is severely outdated [27], [34]. Attacks that were relevant 20 years ago, might not be relevant anymore. Moreover, many new attacks have been developed since (for example, DDoS attacks are not featured at all in KDD99). This critique of course also counts for DARPA1998, of which the data from KDD99 is derived. Likewise, as KDD99 is based on DARPA1998, some of the critiques of DARPA1998 also count for KDD99. However, the critiques of [38] are not applicable to KDD99, as the features used in KDD99 remain unaffected [39]. In [42] the unbalanced character of the attacks appears. While dividing the dataset into 10 seperate parts, they noticed that 4 parts only contained *smurf* attacks, and that one part only contained *neptune* attacks (both DoS attacks). The presence of so many issues led to the development of a better version of KDD99, namely NSL-KDD.

C. NSL-KDD

In [39], various problems of KDD99 are discussed by Tavallaee *et al.*. Firstly, they note that 78% of the records in the training set, as well as 75% of the records in the test set are duplicated. They argue that this causes a model to become biased towards the more frequent records in the dataset. Secondly, as they trained 21 ML models on the data, every model classified 98% of the training and 86% of the test data correctly. Tavallaee *et al.* argue that this signifies that the dataset is fairly easy to classify. For this reason, it is hard to actually compare different IDSs on the same dataset, as they all obtain high scores. Therefore, the authors in [39] present a new dataset, NSL-KDD. In order to achieve this new dataset, they first removed redundant records from KDD99. In order to solve the ease of classification for the dataset, they have also created a specific, difficult subset of the original dataset. By including mostly packets that were poorly classified with the test classifiers, this subset represents the packets that are hard to classify. Often, it is denoted as NSL-KDD^{21} or NSL-KDD-21, since no packets that were correctly classified by all 21 classifiers are included. The full NSL-KDD dataset (containing all packets) is also known as NSL-KDD+ or NSL-KDDC^{+}.

It is important to note that, since it is still based on DARPA1998, the network traffic featured in NSL-KDD remains outdated and less relevant for modern networks. NSL-KDD is the final iteration of the datasets based on DARPA1998.

D. ISCX2012

In [43], the improved generation of intrusion detection datasets is investigated. The authors aim to dynamically generate datasets from specific profiles that simulate agents' traffic behaviours. This way, it is possible to regenerate certain configurations or to create different scenarios. ISCX2012 is the result of generating such a dataset by using α-and β-profiles in a testbed of physical network devices and hosts. α-profiles represents human-designed attacks while β-profiles represent the mathematically modelled behavior of the users of the Information Security Centre of Excellence (ISCX). The α-profiles comprised 4 large attack scenarios that were executed during the generation, namely infiltrating the network from the inside, HTTP DoS attacks using *Slowloris*, DDoS attacks using an IRC Botnet and brute force SSH attacks.

E. UNSW-NB15

Motivated by the growing deprecation of KDD99 and NSL-KDD, University of New-South Wales researchers in [44] created a new dataset striving to improve on the shortcomings of those benchmark datasets. By generating both normal as well as malicious traffic using the IXIA PerfectStorm tool[4] they created the UNSW-NB15 dataset. They used the Bro [45][5] and Argus[6] systems to extract features from the packet capture files and additionally calculated more in-depth features with their own algorithms. The nearly 2 million packet flows present normal traffic and malicious traffic corresponding to nine attack classes: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The same authors statistically analyze the dataset in [46], determining that, when splitting the dataset into training and test sets, both sets are similarly distributed and share statistical properties. Moreover, they indicate that the features are correlated and that UNSW-NB15 is more complex than KDD99 by comparing classification results on both datasets.

Results in both other work [47] as well as our experiments appear to suggest that UNSW-NB15 also is more complex to classify than both ISCX2012 and CICIDS2017, when comparing performances of the same approach for those datasets.

F. CICIDS2017

In [48], the authors list eleven characteristics that are essential for a valid intrusion detection dataset: attack diversity, anonymity, available protocols, complete capture, complete

[4]https://www.ixiacom.com/products/perfectstorm
[5]Formerly Bro IDS, is now known as Zeek: https://zeek.org/
[6]https://openargus.org/

TABLE 1. Different network intrusion detection datasets.

Name	Year	Attack classes	Storage Size‡	Labelled Flows	PCAP Included
DARPA1998	1998	4	20.8GB	4,554,364	✓
KDD99	1999	4	742.6MB	4,898,431†	
NSL-KDD	2009	4	55.9MB	34,394†	
ISCX2012	2012	4	91.1GB	2,450,324*	✓
UNSW-NB15	2015	9	107.1GB	2,540,047	✓
CICIDS2017	2018	14	53.6GB	2,830,743	✓

† This dataset contains records instead of flows. Besides network-based information, these record also contain host-based information.
* Including all flows of Friday that, while not labelled, are considered to be benign traffic.
‡ Only considering PCAP files and label CSV files.

interaction, complete network configuration, complete traffic, feature set heterogeneity, labelling and metadata. Moreover, they created a modern dataset complying to those characteristics, the Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset (CICIDS2017). By setting up 10 benign user agents that generate realistic traffic data, and simultaneously attacking the network from 4 attack hosts, they simulated a dataset spanning one workweek. For each day a number of attacks were applied, resulting in attack data for brute force attacks, heartbleed attacks, botnets, DoS attacks, DDoS attacks, web attacks and infiltration attacks. Globally, this results in 15 attack classes corresponding to these 6 categories. While this course of action provides a modern and diverse dataset, there are still some shortcomings. Researchers in [49] identify 4 issues for CICIDS2017. Firstly, they argue that, as the dataset is divided over 8 files, it is difficult to work with. Secondly, the dataset contains a huge amount of data, with 3119345 instances across those files. Thirdly, 288,805 instances in CICIDS2017 miss either a class label or other information. Finally, the dataset has serious issues with class imbalance, with the heartbleed attack containing as little as 11 instances while there are over 2 million benign instances. The first three issues can be dealt with easily, by respectively aggregating the dataset files, sampling the dataset and by removing instances missing information. For the class imbalance, the authors suggest combining minority attack classes belonging to the same attack category. The three different web attacks can for example be combined into one web attack class. While this provides relief for some attack classes, it still does not help in case of the infiltration class with only 36 instances.

In this section, we discussed the most-used datasets in the field that are relevant for this paper. Among these datasets, UNSW-NB15 and CICIDS2017 are most recommended for use in [32], as they contain a wide range of modern attack scenarios. ISCX2012 is also suitable, but features somewhat older network traffic, which has to be kept in mind.

V. EVALUATION METHODS

If different intrusion detection algorithms and implementations are to be compared, it is important this happens on a common ground. Besides an understanding of the different datasets, this also requires evaluation metrics that are usable for every algorithm with each dataset. In the literature, we observe that each author makes an own choice from the plethora of metrics that can be used to evaluate NIDSs: accuracy, precision, detection rate, false positive rate, F-measure, etc. Typically, they are used in two scenarios: The binary scenario and the multiclass (n-ary) scenario. While metrics such as the precision are very straightforward to calculate in a binary scenario, the multiclass scenario requires metric values for the different classes to be aggregated. In practice, this can be done using macro averaging (denoted with M), micro averaging (denoted with μ) and weighted averaging (denoted with w). Macro averaging considers every class equally, while both micro averaging and weighted averaging favor large classes. We present a more in-depth overview and formulas of these evaluation metrics in appendices A-A and A-B.

As will be elaborately discussed in Sect. VI-D, one issue in the field of network intrusion detection is that many authors report their results in various ways, complicating comparison between different work. Therefore, we propose two metrics that reliably evaluate the overall performance of an intrusion detection system. These metrics, the *detection score* and *identification score*, are ideally used in tandem.

A. DETECTION SCORE

The main task of any intrusion detection system is to reliably detect intrusion. In the most basic sense, this comes down triggering an alarm whenever an attack is detected. In the case of network intrusion detection with a normal class and an attack class, we define the *detection score* (DS) as the F-measure with the attacks as positive class: $DS = F1$. This is the case for anomaly-based approaches, as these generally identify attacks in normal traffic without elaborating into different attack classes. However, in the case of multiple attack classes, this calculation no longer is straightforward. Rather than aggregating individual F-measures per class, we propose a method to map the multiclass confusion matrix to a binary class confusion matrix, which would allow for the use of the binary F-measure as DS. This mapping is demonstrated in Fig. 1, with more information about confusion matrices in appendix A-C. Specific to this approach is that any predicted attack that corresponds with any true attack is considered

a true positive, even if the attack classes differ. We reason that any detected legitimate attack is beneficial, even if the attack is initially detected for the wrong reasons. Thus, the DS allows for assessing the base performance of a NIDS, namely its ability to detect attacks. In our implementation, we consider the F-measure to account for both the precision and detection rate of an NIDS. The detection rate outlines the ability to not miss any attacks, while the precision serves to limit false alarms. Both properties are required for a real-life NIDS, as it should neither miss attacks nor produce too many false alarms. While it is possible to use either metric instead of the F-measure in the calculation of the DS, we argue that both should be kept in mind concurrently. In the scenario that either precision or recall is more important than the other, the F_β*-measure* can be used instead. We will only consider $\beta = 1$, where precision and recall are equally important.

In order to evaluate actually identifying (attack) classes, we also propose the *identification score*.

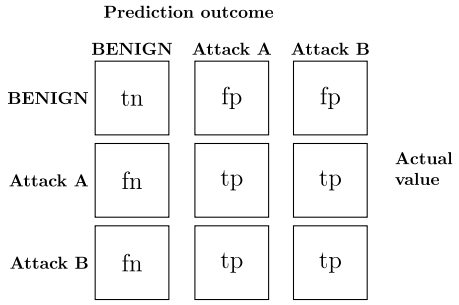


FIGURE 1. **Mapping of attack classes to binary.**

B. IDENTIFICATION SCORE

The second task of an NIDS is the classification of detected attacks into different attack classes, as correctly identifying attacks helps in formulating a response. However, network intrusion detection datasets suffer from class imbalance: Minority classes are underrepresented in the dataset, while majority classes are overrepresented. On the one hand, the results for the majority classes will dictate the overall outcome for weighted averages. This provides a reliable overview of the overall performance, but might mask poor performance for a minority class. On the other hand, as macro averages do consider minority classes, they can mask a good overall performance in the case of poorly detected minority classes. For network intrusion detection, both the overall performance as well as the performance per class are important. Therefore, only using one instead of both aggregation approaches could unreliably display performance. As also discussed in Sect. VI-D, most other works only report weighted averages, forgoing the macro averages. Therefore, we propose the *identification score* (IS), which we define as the harmonic mean of the weighted average F-measure and the macro average F-measure:

$$IS = H\left(F1_w, F1_M\right) = \frac{2 \cdot F1_w \cdot F1_M}{F1_w + F1_M} \tag{1}$$

Since the IS is a harmonic mean, it will penalize small values of either $F1_w$ of $F1_M$, only resulting in a high value if both inputs are also high. It will lie between 0 and 1, but a value of for example 0.9 already indicates that at best, both averages $F1_w$ and $F1_M$ have a value of 0.9. However, for unbalanced classes, one average may be significantly higher than 0.9. In such a case, the other value must be significantly lower than 0.9. Smaller values for the IS can therefore indicate that either one average is very small, or that both values are equally small. Both situations are undesirable.

The IS is not the only metric that aims to handle class imbalance. For example, Receiver Operating Characteristic (ROC) or Precision-Recall (PR) curves with an Area Under Curve (AUC) value [50] use thresholding to evaluate the performance of a dataset. According to [51], the Matthews Correlation Coefficient (MCC) is the least biased metric for binary classification that takes into account both classification successes and errors. In principle, training a model implies minimizing the misclassification cost [52]. For IDSs in particular, this cost encompasses not only the impact of intrusion, but also operational costs and the hostility of the environment [53]. It is however hard to find this cost, as it depends on uncertain parameters. When comparing against such other evaluation approaches, there is a clear difference with the IS. First, unlike for ROC or PR, no thresholding is required in the calculation. This is valuable in a setting where not all data to apply thresholding are present, for example when comparing results of various authors that did not provide their models. Moreover, comparing to many different curves in one or more graphs can quickly become cluttered, complicating the comparison process. MCC similarly heavily depends on a confusion matrix to be computed, which is not always provided by authors. It also incorporates the number of true negatives (normal traffic that does not raise alarms), which is not really relevant to the performance of network intrusion detection systems. The advantage of the IS, when compared to other metrics, is that it can easily be deduced when the F-measures for each individual class, as well as the class weights are known. These values are among the most reported evaluation metrics.[7] Moreover, as mentioned in section V-A, the F-measure also accounts for both the false alarms (in the precision) and missed attacks (in the recall).

VI. STATE-OF-THE-ART ALGORITHMS

Many different methods are used to perform network intrusion detection on specialized datasets, of which some datasets are discussed in Sect. IV. This paper however does not strive to exhaustively list every network intrusion detection technique, as that would be nearly impossible, but rather to provide an overview of the current state-of-the-art for ML approaches. This overview is further divided into two major segments: Traditional machine learning and deep learning. While both are important and relevant, the main focus will

[7]The overall accuracy similarly is very often reported, but far less insightful for unbalanced classification,

lie on deep learning. Tables 2 and 3 give the performance of the algorithms that are discussed in the remainder of this section, for the datasets introduced in Sect. IV. The tables report performance following two methods: The first provides the accuracy and weighted averages for precision, detection rate (DR), false positive rate (FPR) and F-measure as originally reported, if applicable. Secondly, whenever possible, we calculated the performance following our proposed metrics: DS and IS. In order to visualize the effect F-measures have on the IS, we also provide their weighted and macro averages. Before discussing different machine learning techniques however, we will first consider what features are used in network intrusion detection problems in the next section.

A. FEATURES

Feature selection and extraction is essential for any machine learning task. For network intrusion detection, proposed techniques can grossly be divided into two groups relating feature usage: One group starts from provided features, while the other extracts their own features. The first group uses dataset-specific features, that were extracted at the dataset's conception, manually selected by experts. Most methods using these features simply apply normalization for numerical features and one-hot encoding for the categorical features. For example, the 41 features provided in KDD99 are usually preprocessed into 122-dimensional input vectors. If the preprocessing for a specific study is very similar to this process, we will not discuss it here. However, the second group of research encompasses all other approaches that significantly modify the provided features or extract completely new features. If for example a research uses raw traffic header bytes as features, that will be discussed here.

B. TRADITIONAL MACHINE LEARNING METHODS

In this section, we will discuss traditional ML-based NIDSs, namely Support Vector Machines, Decision Trees and Random Forests, Extreme learning Machines, Restricted Boltzmann Machines and some other approaches. We strive to give a very general overview of what is being done.

1) SUPPORT VECTOR MACHINES

Even to date, one of the most prevalent ML-based intrusion detection techniques is the Support Vector Machine (SVM). SVMs are classifiers that work by finding the hyperplane separating classes with a maximal margin [86].

In 2003, researchers explored the use of Robust SVMs to classify process and session data from the DARPA1998 dataset [87]. Starting from the observation that datasets supposedly clean usually contain some noise consisting of malicious traffic, they strove to design a robust classifier trainable on noisy data. By modifying the constraints and objective function of the SVM, they obtained a recall of 100% with a FPR of 8% where the reference SVM was unable to even reach a recall of 100% without having the FPR reach 100% as well. The authors in [63] describe another way to deal with dataset issues, proposing a modified K-means approach to generate a high-quality dataset from KDD99. By using a distance threshold, they add a cluster centroid to the set whenever a sample exceeds this threshold for every other centroid. Furthermore, they construct a multi-level classifier, detecting a traffic class at each level. Notably, they use an Extreme Learning Machine (ELM) rather than an SVM to detect *Probe* attacks, as they argue an ELM is better suited for that task. With every other classifier being an SVM, they achieved state-of-the-art results. In [81], the authors use SVM and Multilayer Perceptron classifiers on UNSW-NB15, while employing the feature dimensionality reduction approaches Principal Component Analysis (PCA) and a chi-squares test. They achieve both a high accuracy and F-measure of above 90%, surpassing other work on the same dataset.

In [88], the authors strive to implement a stable and accurate incremental SVM learning scheme. This allows for manageable retraining in real-time to counteract increasing training set and training time. By using reserved sets with weighted non-support vectors, they can retrain the classifier without having to use the entire training set. Moreover, they also modify the RBF kernel to further reduce training and testing speed and increase reliability.

2) DECISION TREES AND RANDOM FORESTS

Decision trees give structure to a set of rules, derived from and using the input features for, among other things, classification [89]. Random forests (RF) then combine many different decision trees for the same problem and take an ensemble of their results to achieve a more robust classifier [90]. In [54], the RF approach is used to construct a misuse-based classifier, an anomaly-detector and a hybrid approach evaluated on KDD99. For the misuse-based approach, a RF is used as a classifier to differentiate between normal traffic and intrusion, resulting in a reported error rate of 7.07%. By using the proximity of samples of traffic in the RF, outliers can be detected for anomaly detection with a reported DR of 65% and FPR of 1%. Finally, by running all traffic unclear to the misuse-classifier through the anomaly-detector, the authors propose a hybrid IDS approach with an overall DR of 94.7%.

As NIDS datasets suffer from class imbalance, Wei Zong *et al.* [83] propose a two-stage approach for network intrusion detection. After over-sampling the minority classes and down-sampling the majority classes, they first classify input data into a minority class or *other*. This *other* class is then classified into the different majority classes in the second step. Both classifiers are RF, but can be exchanged for other techniques. After testing on UNSW-NB15, they achieve a result comparable to the best result in the analysis of that dataset [46].

Combining ensemble methods and DL, the authors in [65] aim to achieve high detection performance with a low training time. They apply both Multi-Grained Traversing as well as Cascade Forest in an effort to surpass the work they compare against, which succeeds.

TABLE 2. State-of-the-art results for ML-based NIDSs. Values in *italics* are originally reported rather than recalculated. $F1_W$ and $F1_M$ in binary scenarios are only relevant for IS calculation. The best value for each calculated metric is in **bold**.

Source	Classes	Technology	Year	Reported					Calculated			
				Acc	Prec	DR	FPR	F1	DS	$F1_w$	$F1_M$	IS
KDD99												
[54]	/†	RF	2008			94.7	2					
[55]	/‡	MLP	2017	99.08		99.81	0.05					
[56]	2	Jordan/Elman	2007			91.91	8.08					
[57]	2	CNN + LSTM	2015	99.7				99.8	***99.8***			
[58]	2	CNN	2019	98.34	99.20	90.64						
[58]	2	CNN	2019	98.34	99.20	90.64						
[59]	5	Reduced RNN	2010			94.10	0.38		97.06	94.26	60.73	73.87
[60]	5	LSTM	2015	93.82					96.20	91.56	58.62	71.48
[57]	5	CNN + LSTM	2015	98.7								
[57]	5	LSTM	2016			98.96	7.78					
[61]*	5	ELM	2016							93.17	72.60	81.61
[62]	5	DBN	2016	97.9					99.23	97.47	65.38	78.26
[63]	5	SVM-ELM	2017	95.75		95.17	1.87					
[64]	5	NDA-RF	2018	97.85	99.99	97.85	2.15	98.15		**98.60**	63.21	77.03
[65]	5	DDF	2019		91.7	86.2	2.8	83.1		94.25	57.76	71.63
[66]	5	GABP-DS	2019			97.5	2.1			92.88	**91.56**	**92.22**
[47]	5	MLP	2019	93.5	92.0	93.5		92.5		*92.5*		
NSL-KDD+												
[67]	2	DA-ROS-ELM	2016	99.16			0.86					
[68]	2	RNN	2017	83.28					83.24	83.28	83.28	83.28
[69]	2	ResNet50	2017	79.14	91.97	69.41		79.12	79.12	79.14	79.14	79.14
[69]	2	GoogLeNet	2017	77.04	91.66	65.64		76.50	76.50	76.96	77.03	76.99
[70]	2	GoogLeNet	2018					82††	*82*			
[71]	2	1D-CNN	2018					76††	*76*			
[72]	2	SAE + SVM	2018	84.96	96.239	76.57		85.28	*85.28*			
[73]	2	KA-ANN	2019	88.7	95.6	84.0		89.4	***89.4***			
[74]	2	AE-CNN	2019	89.41								
[68]	5	RNN	2017	81.29					83.22	79.26	**62.86**	70.11
[75]	5	DBN	2015	97.45								
[76]	5	CNN	2018	80.13		80.13	13.57					
[72]	5	SAE + SVM	2018	80.48	93.92	68.29		79.08		*79.08*		
[64]	5	NDAE-RF	2018	85.42	100	85.42	14.58	87.37		**91.79**	61.19	**73.43**
[77]	5	MDPA-DBN	2019	82.08	97.27	70.51	2.62	81.75	81.75	76.47	56.96	65.29
[78]	5	DST-TL	2019	84.60								
[47]	5	MLP	2019	78.1	78.5	78.1		76.4		*76.4*		
[73]	5	KA-ANN	2019	80.7	81.8	80.7		79.3		*79.3*		
NSL-KDD21												
[68]	2	RNN	2017	68.55								
[69]	2	ResNet50	2017	81.57	81.81	99.63		89.85	89.85	73.60	45.11	55.93
[69]	2	GoogLeNet	2017	81.84	81.84	100		90.01	**90.02**	73.68	45.01	55.88
[70]	2	GoogLeNet	2018					75††	*75*			
[71]	2	CNN	2018					63††	*63*			
[74]	2	AE-CNN	2019	80.36								
[68]	5	RNN	2017	64.67								
[76]	5	CNN	2018	63.32		68.20	9.37					
[77]	5	MDPA-DBN	2019	66.18	95.51	61.57	13.06	74.87	74.87	63.81	51.76	57.16
[78]	5	DST-TL	2019	79.90								
ISCX2012												
[79]	5	CNN	2017	99.69		96.91	0.22					
[79]	5	CNN + LSTM	2017	99.89		96.96	0.02					
[80]	5	RBM	2018	88.6 ± 0.6		88.4 ± 0.8						

† Hybrid IDS, first detects known attacks, then detects anomalies.
‡ Not specified.
* Metrics were only reported per class, not globally.
†† Reported as graph only.

3) EXTREME LEARNING MACHINES

Extreme Learning Machines (ELM) are single hidden layer feedforward neural networks that use randomly generated input weights and calculated output weights. They excel in their high training speed while providing adequate generalization [91]. Yuanlong Yu *et al.* [61] aim to improve

TABLE 3. Continued: State-of-the-art results for ML-based NIDSs.

Source	Classes	Technology	Year	Reported Acc	Prec	DR	FPR	F1	Calculated DS	$F1_w$	$F1_M$	IS
				UNSW-NB15								
[70]	2	GoogLeNet	2018	83††				86††	*86*			
[81]	2	ANN	2019	92	92	92		91	*91*			
[81]	2	SVM	2019	91	93	92		92	***92***			
[82]	2	MLP	2019	99.5			0.47					
[73]	2	KA-ANN	2019	89.0	88.6	91.8		90.2	*90.2*			
[83]	10	RF	2018	85.78			15.64		88.40	77.60	**52.69**	**62.76**
[77]	10	MDPA-DBN	2019	90.21	87.3	96.22	17.15	91.54	91.54	**78.92**	44.70	57.07
[47]	10	MLP	2019	66.0	62.3	66.0		59.6	*59.6*			
[73]	10	KA-ANN	2019	77.8	77.2	77.8		77.3	*77.3*			
				CICIDS2017								
[70]	2	GoogLeNet	2018					82††	*82*			
[84]	2	PCA-RF	2019	99.7		99.0	0.1	99.7	*99.7*			
[84]	2	SAE-RF	2019	99.7		98.9	0.1	99.7	*99.7*			
[47]	7	MLP	2019	96.2	97.2	96.2		96.5	*96.5*			
[85]	12	PCCN	2019	99.92								
[84]	15	PCA-RF	2019	99.6				99.7	99.21	**99.64**	78.15	87.60
[84]	15	SAE-RF	2019	99.6				99.6		*99.6*		
[84]	15	SAE-RF-UDBB	2019	98.97	98.9	98.8	0.1	98.8	**100.00**	98.97	**98.84**	**98.91**

†† Reported as graph only.

unbalanced multiclass classification by designing a cascaded scheme of binary ELM classifiers, in which each ELM is trained on KDD99 to detect one class. While retaining high performance on the majority classes, they succeed in improving the classification of the minority classes.

Yi Yu *et al.* [67] propose a Dual Adaptive Regularized Online Sequential ELM (DA-ROS-ELM) to allow for real-time training of the output weights. They report a high accuracy while maintaining a high training speed.

4) RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann Machines (RBM) are generative models consisting of a visible and a hidden layer [92]. They can be used for both unsupervised learning as well as classification. The authors in [80] implemented an RBM that is trained and evaluated on a balanced subset of ISCX2012, achieving an accuracy and a DR of about 88%.

In [93], a Discriminative RBM is used to perform classification with an accuracy of about 84.65% on KDD99.

5) OTHER APPROACHES

Traditional ML is not limited to SVMs, decision trees and neural networks. For example, Genetic Algorithms (GA) are usually used in combination with other approaches to improve their performance: In [41], [94] genetic algorithms are used to improve SVM performance, while [66] improve the functionality of a MLP.

Rather than using DL, in [73] Manuel Lopez-Martin *et al.* employ shallow neural networks for intrusion detection. They apply three different kernel approximation methods to project the input features to a higher dimension, in order to be able to linearly separate the classes in the neural network.

Other, older techniques include Hidden Markov Machines [95], k-Nearest Neighbours [96] and clustering [97].

C. DEEP LEARNING METHODS

DL-based NIDSs, as opposed to regular ML approaches, use a multitude of subsequent hidden network layers to perform complex calculations. These networks can range from regular multilayer perceptrons to convolutional neural networks and (auto)encoders. Each of these architectures has its own strengths and weaknesses regarding the detection of intrusions in network data. In the following sections we discuss the different architectures and relevant techniques proposed in research.

1) MLP

Multilayer Perceptrons (MLP) can be regarded as a baseline neural network structure. While many more complex or elaborate techniques are used, MLPs remain relevant and are used in NIDSs. Liu Zhiqiang *et al.* [82] use a 10-layer MLP to binarily classify the traffic of UNSW-NB15. With an accuracy of 99.5% and FPR of 0.47%, they obtain very high results. The authors of [55] achieved similarly high results by using a 4-layer MLP on KDD99, achieving an accuracy of 99.08%. However, they do not report whether this concerns binary or multiclass classification.

Chunlin Lu *et al.* [66] apply a genetic algorithm along with a Dempster-Shafer (DS) decision fusion in an effort to better leverage the potential of back-propagated neural networks. After removing 12 features with little influence from the KDD99 dataset, they group the remaining features in three subsets that are used to train a neural network with one hidden layer. By fusing the outputs of these three networks with

the DS-model, the resulting GABP-model (Genetic Algorithm BackPropagation) achieves state-of-the-art results.

In [47] an extensive research is conducted, developing a DNN that can be used for NIDS and HIDS and is tested on KDD99, NSL-KDD, UNSW-NB15, WSN-DS [98], CICIDS2017 and Kyoto [99]. By comparing different configurations of a DNN for these different datasets and against other ML approaches, the authors provide an insightful overview of performance. It must be noted however that not the entirety of CICIDS2017 has been included in the research, as classes such as *Infiltration* appear to be excluded.

MLPs are also often used as baseline comparison for a new technology, for example in [67], [69].

2) CNN

Starting from the inception of AlexNet [100] in 2012, Convolutional Neural Networks (CNN) have been used in diverse settings for diverse purposes. Some important domains include image recognition [101], object detection [102] and Natural Language Processing [103]. Their success in these domains can be partially attributed to their ability to interpret spatial characteristics. Therefore, CNNs have also been applied in intrusion detection applications in recent years.

Generally, convolution layers for the processing of traffic data are 1-dimensional or 2-dimensional. In [71], 1D convolutional layers are used to perform binary classification on data from NSL-KDD, MAWILab [104] and Kyoto [99]. Kwon *et al.* test three different configurations using either one, two or three 1D convolutional layers as well as max pooling layers and fully connected layers. From their experiments, they conclude that the shallow model with only one convolution layer consistently outperforms the other models. With F-measures of about 76% for NSL-KDDTest$^+$, about 63% for NSL-KDDTest^{21} and a little over 60% for MAWILab, it does not outperform other algorithms for the same datasets. For Kyoto, their results fluctuate heavily and are hard to draw conclusions from. Another example of 1D-convolution can be found in [76], where the authors use a custom 1D CNN to classify the entirety of NSL-KDD. With high FPRs and relatively low accuracies and DRs, the system fails to distinguish itself from similar technology however.

Besides 1D, CNNs can also use 2D convolution on input data. This means that features are turned into 2D images that are classified. How these images are generated, varies between different implementations. In [105], the 41 NSL-KDD features are turned into binary vectors. By using 10-bit vectors for the discretized numerical features and n-bit vectors for categorical features, where n is the number of categories, an 8-bit grayscale 8×8 image can be created. Using ResNet50 and GoogLeNet, they reach accuracies of 80% on both NSL-KDD$^+$ as well as NSL-KDD^{21}. However, as they receive false positive rates of respectively 99.81% and 100%, their results for NSL-KDD^{21} are not very useful. For this dataset, their implementations simply report each data sample as an attack. Researchers in [70] argue that using this grayscale encoding provides numerical features with 1.25 pixels (10 bits, 8 bits per pixel), while 2-bit categorical features only account for 0.25 pixels in the image. Moreover, they also argue that discretizing numerical features in 10 values provides too rough an estimation of their actual value. Therefore, they propose RGB-like encoding where numerical features range between 1 and $2^{24} - 1$. The categorical features are one-hot encoded by using 0×000000 and 0xffffff instead of 0 and 1. In their experiments, they report a considerable improvement when compared to the grayscale encoding technique, obtaining F-measures of about 82% on NSL-KDD$^+$, about 75% on NSL-KDD^{21}, about 86% on UNSW-NB15 and about 82% on CICIDS2017. We are unable to provide calculated metrics for more insight, as insufficient information is given in their article.

Another approach to using KDD99 features is given in [58], where the authors use VGGNet to classify 11×11 images derived from those features. This approach results in an accuracy of 98.34%, on par with other techniques proposed on the same dataset.

The authors of [85] take the image generation one step further by no longer using precalculated features, but by instead using fragments of the raw packets themselves. In their experiments, they evaluate three different methods of feature generation by using packet headers, payloads or a combination of both. Header features for example are a string of the first 50 bytes of 5 subsequent packets in a network flow, separated by an 0×00, resulting in a 16×16 grayscale image. Correspondingly, payload features contain the 51st to the 100th byte of 5 subsequent packets, generating different 16×16 images. Unlike the header and payload features, the combination generates 22×22 images by concatenating the first 96 bytes of 5 subsequent packets and adding 0×00s accordingly. Note that packets belong to a flow if they share their source and destination addresses, their source and destination ports and the protocol used. Besides their novel feature generation, they also apply a specific Parallel Cross-Convolutional neural Network (PCCN) to better classify the highly imbalanced classes of CICIDS2017. While reporting results for each attack class, they only differentiate between those attacks as they do not include normal traffic. Therefore, it is impossible to draw definite conclusions about the algorithm's ability of detecting attacks in normal network traffic. We have implemented this algorithm for three datasets with normal traffic included, and present the results in Sect. VII-B

In [106], researchers examine the possibilities of transfer learning for intrusion detection. Concretely, the authors first train a base CNN on UNSW-NB15, which then remains fixed to train a second CNN that is added onto the first. The aggregate of both CNNs is then trained on NSL-KDD and tested for NSL-KDD+ and NSL-KDD21.

3) RNN, GRU AND LSTM

Recurrent Neural Networks (RNN), Gated Recurrent Networks (GRU) and Long-Short Term Memory (LSTM) have successfully been applied in areas such as Natural Language

Processing [107], traffic forecasting [108] or remaining useful life estimation [109]. LSTM [110] and GRU [111] are alternative gate units that can replace the standard gate unit of RNN. They share the ability to take the temporal aspect of data into account: data samples are processed based on preceding samples [112]. Consequently, their use for intrusion detection has also been investigated. More specifically, as they are able to inspect time series of variable length, they might be able to identify attacks based on their relation to the previously transmitted packets. One very early example of this is [113], where a RNN analyzes audits in a HIDS in 1992. For NIDSs, early uses of RNN include [56], [59] where specific RNN models are used to classify the KDD99 dataset. [56] uses, among others, Jordan/Elman recurrent networks [114], [115] to train on on a very small subset of KDD99. With a 91.91% overall detection rate, a corresponding false alarm rate of 8.08% and comparable results for MLP and PCA-based approaches on binary classification, the algorithm does not quite meet the requirements for real-life implementation.

The authors of [59] use a partially connected RNN architecture, employing KDD99 for training and testing. They report a DR of 94.1% while maintaining a FPR of 0.38% on multiclass classification, outperforming similar research at the time. More recently, Chuanlong *et al.* employed a RNN to perform classification on the NSL-KDD dataset [68]. By testing out a number of different network configurations, they obtained to optimal accuracies. Concretely, they show binary classification accuracies of 83.28% for NSL-KDD$^+$ and 68.55% for KDD^{21}, and multiclass accuracies of 81.29% for NSL-KDD$^+$ and 64.67% for NSL-KDD^{21}. As the results for NSL-KDD$^+$ hover only slightly above 80%, it is clear that there is still room for improvement for this approach.

Not only the basic RNN sees use in intrusion detection research, the more advanced LSTM option has also shown its potential. For example in [116], a LSTM Sequence to Sequence (LSTM-Seq2Seq) model is used for NSL-KDD and Kyoto-Honeypot data. This model consists of an encoder turning an input sequence into a fixed-length context vector, and a decoder giving the probability of the input sequence being either benign or an attack by analyzing the context vector. As the deep version of the model, featuring multiple LSTM layers, has an F-measure of over 99% for NSL-KDD and 100.0% for Kyoto, it appears to be very effective. Evidently, this greatly outperforms all other algorithms that were also tested in the study, being Fully Connected Network and Variational Autoencoder. However, no confusion matrices or source code are provided to more deeply analyze these results.

Another example of the potential of LSTM is given in [117], once again for NSL-KDD. Among a large selection of ML algorithms, LSTM surpasses its competition by obtaining accuracies of 89% on KDDTest$^+$ and 83% on KDDTest^{21}. [60] experimented with many different configuration settings for the use of LSTM for KDD99, testing both the use of all features as well as the possibility of reducing the number of features with a J4.8 decision tree. While using all features, an accuracy of 93.82% is obtained. Important to note however, are the very low DR values for attacks U2R and R2L in this model. Reducing the number of features to respectively 8 or 4 then results in accuracies of 93.69% and 93.72%. Although the overall accuracy is maintained, the detection of minority classes U2R and R2L is increasingly inhibited. Consequently, even if it is proven that KDD99 can be classified using a very small number of features, the performance for minority classes needs to be considered.

Another research focusing on the use of LSTMs for KDD99 is [118], where the authors use a small subset of the dataset consisting of 1000 normal samples and 300 samples for each attack except the very underrepresented U2R. This approach results in acceptable detection of normal and DoS attacks, but only a limited number of the other attacks is detected. The overall DR of 98.96% is comparable to the state-of-the-art, but the FPR of 7.78% remains too high.

4) CNN + RNN

With CNNs exploring the spatial characteristics and RNNs considering the temporal aspect of traffic data, a combination of both methods has great potential. In [57], CNNs are used to generate feature maps from the base 41 features of KDD99. After a maxpooling operation, these new features are then presented as input features to the recurrent model. With either a RNN, GRU or LSTM as recurrent network, the ideal configuration is sought. For binary classification, the CNN with 2 LSTM layers, an accuracy of 99.7% and F-measure of 99.8% outperforms other solutions. In multiclass classification, the CNN with 3 LSTM layers obtains the highest accuracy of 98.7%. It is however not possible to give any weighted averages, as the weight of each class is not clearly specified. More extensive research is conducted in [79], in which HierArchical Spatial-Temporal features are used to create an IDS (HAST-IDS). Defining HAST-I and HAST-II as configurations that process either an entire flow or a limited number of packets in a flow, enables intrusions in DARPA1998 and ISCX2012 to be detected. HAST-I uses a CNN to classify $m \times n$ flow images created by concatenating the first n one-hot encoded m-dimensional bytes of a flow. Similarly, HAST-II generates $r \times p \times q$ images concatenating the first q one-hot encoded p-dimensional bytes of a packet, it then repeats this step for r packets. However, after applying a CNN to generate features, HAST-II uses an LSTM to temporally analyze the sequence of packets as opposed to only using a CNN to analyze the entire flow. After determining the ideal values for n, q and r, the authors in [79] achieve high values for accuracy, DR and FPR. The results for DARPA1998 are not discussed here for two reasons: HAST-IDS would be the only relevant investigation to use the dataset. and HAST-II does not use DARPA1998 as r is 1 or 2 for over 63% of flows. For ISCX2012, HAST-I achieves an accuracy of 99.69%, a DR of 96.91% and a FPR of 0.22%. Similarly, the results for HAST-II, show an equivalent detection performance with a near negligible FPR of 0.02%.

5) DBN

Deep Belief Networks can be viewed as stacks of RBMs that evade the disadvantages of backpropagation in DL by using unsupervised learning [119]. In [75], DBNs are used for NSL-KDD to achieve an accuracy of 97.45%. Similarly, authors in [62] apply a DBN with logistic regression to detect intrusions in KDD99. By removing duplicate samples from the training set and designing a 4-layer network, they achieve an accuracy of 97.9%.

Yang *et al.* [77] modify the density peak clustering algorithm (DPCA) [120] to better suit complex intrusion detection datasets. After dividing the training data into clusters, they train a DBN on each cluster and calculate a fuzzy membership matrix for each test sample related to the clusters. They then use this membership to aggregate the outputs of the different DBNs for a test sample. The resulting Modified Density Peak Clustering Algorithm and Deep Belief Networks (MDPCA-DBN) approach is tested on NSL-KDD and UNSW-NB15, yielding high results. Calculating the same metrics on the confusion matrices they provide produces somewhat lower values.

6) AUTOENCODERS

Autoencoders are DL networks that allow for unsupervised learning. Generally, they consist of two parts: An encoder and a decoder. The encoder first encodes the input data, typically to a lower dimension. This encoded data is then decoded back to its original dimension, the result of which should be equal to the original input data. In the field of intrusion detection, autoencoders are usually combined with other mechanisms to improve detection. The autoencoders then serve as dimensionality reduction devices that generate features that can be more efficiently processed by the following classifier. For example in [72] a sparse autoencoder (SAE) is trained on NSL-KDD features in order to improve both the classification accuracy as well as the processing time of the SVM intrusion detector. The resulting detection performance is on par with other approaches for NSL-KDD, but lacks a reported FPR. While the autoencoder in this set-up presents DL, the SVM in practice still is a traditional ML technique.

In [84] this principle is extended, using either a SAE or Principle Component Analysis (PCA) to reduce the features' dimensions before classification. Between the Random Forest (RF), Bayesian Network (BN), Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) classifiers, the RF classifier obtained the most promising results. After also applying Uniform Distribution Based Balancing (UDBB) to combat class imbalance, the authors reach near perfect detection performance for CICIDS2017.

Researchers in [78] investigate the use of transfer learning for their SAE, implementing it as the neural network in their Deep neural network and adaptive Self-Taught-based Transfer Learning (DST-TL) approach. They use layers trained on power forecasting data of European wind farms and argue that NSL-KDD data has the same time-series-based and unpredictable nature. While they obtain results comparable to the state-of-the-art, the transfer-learning approach only is a slight improvement compared to the same architecture trained from scratch.

In [74], researchers try to improve the classification of network traffic by incooperating Stacked AutoEncoders for feature encoding. Each stacked autoencoder consists of 3 autoencoders that are trained on normal traffic. By then taking the error vector of a stacked autoencoder, they obtain a measure of the deviation from normal traffic behaviour. Multiple error vectors from different stacked autoencoders are then assembled as the channels of a 1D-image that is classified using a CNN.

Instead of using stacked regular autoencoders, in [64] the authors define a Non-symmetric Deep Autoencoder (NDAE) in which only the encoding phase is utilised. Using this structure to reduce the dimensionality, they performed RF classification on KDD99 and NSL-KDD obtaining state-of-the-art results.

Another application of autoencoders is given in [121], where Stacked Denoising Autoencoders are used to learn useful features from traffic data. More specifically, they first extract session-based features from the raw network traffic. These features include session metadata and 983 payload bytes, and are used as input for the SDA. However, during training these input data are first corrupted by setting some units to zero, in order to more robustly process noisy data. The authors then combine normal traffic of ISCX2012 with botnet attacks from CTU-13 [122] to train the system, obtaining accuracies of 99.48% on binary classification and 98.11% in a multiclass scenario, surpassing other configurations they explored. However, as no further information is given regarding the merging of two different datasets, it is hard to draw further conclusions.

Besides these examples, some authors also use autoencoders in comparison with other algorithms such as in [116] where the LSTM-SeqSeq model surpassed all other models, including the autoencoders. Similarly, the autoencoder-based designs in [117] are surpassed by LSTM and CNN alternatives.

D. DISCUSSION

In order to be able to compare approaches, Tables 2 and 3 present the results for the KDD99, NSL-KDD$^+$, NSL-KDD^{21}, ISCX-2012, UNSW-NB15 and CICIDS2017 datasets. Following these results, we can make a number of observations.

Firstly, the need for standardized result reporting becomes apparent. Clearly, we observe that every author uses different evaluation measures to report performance. Moreover, when reporting multiclass results, researchers use various averaging approaches. While globally the weighted average appears to be most prevalent, some authors calculate this in a different way. This can be seen in how some reported metrics deviate from our calculated metrics, for example for [65] or [77]. Moreover, some authors only provide very limited reporting regarding the different metrics, often disregarding to report

the precision, DR, FPR or F-measure. While they sometimes alleviate this by explicitly reporting other metrics such as a Receiving Operator Characteristics curve (ROC-curve), that is not always the case. If for example the provided confusion matrices for NSL-KDD[21] in [69] are disregarded, the FPRs of practically 100% resulting in very poor ISs do not stand out. Transparently reporting these metrics allows for a better performance assessment.

Secondly, some trends are visible in the data, as data suggest that binary classification is generally more accurate than multiclass classification, following the DS values. Moreover, trends regarding dataset complexity can be suggested. Clearly, NSL-KDD[21] is significantly harder than NSL-KDD$^+$. CICIDS2017 appears to be reasonably manageable, while UNSW-NB15 and KDD99 remain challenging even for very recent techniques.

Thirdly, the issue of dataset imbalance remains important. While not directly clear in weighted averages, metrics such as the average F-measure and the IS reveal that not all classes are well classified, even in the case of otherwise high results. Upon closer inspection for specific architectures, this appears to be caused by attacks such as U2R and R2L in KDD99. These attacks are so underrepresented that systems fail to accurately learn their pattern or behaviour. Although the authors all report very high ($> 90\%$) scores on accuracy and DR, our IS measure clearly provides a more nuanced view.

Fourthly, even very recent research is still only being validated on datasets such as KDD99 and NSL-KDD. As argued, these flawed and outdated datasets are a very poor representation of contemporary network traffic. The relatively limited amount of research on more recent datasets such as UNSW-NB15 and CICIDS2017 in comparison underlines this issue.

Finally, our proposed evaluation metrics allow for a more straightforward comparison between results, requiring only 1 or 2 values to be investigated while giving a clear overview of the performance of the different techniques. Take for example the results reported in [59] or [60]: While it is possible to investigate the results of individual papers by looking up specific graphs or confusion matrices, this is hardly practical on a larger scale. For large scale comparisons, it is necessary to use only one or a few metrics that can be applied to each method. When using the metrics provided by the authors, their results appear to be very good, while in practice their methods performed poorly for certain classes, as demonstrated in their low identification scores. This trend can be seen throughout the tables: Using only reported metrics conceals certain issues, while the combination of *DS* and *IS* indicates whenever something is amiss. However, actually identifying the root of such issues will most likely require actually investigating the original work. As such, our proposed metrics are excellent tools for large-scale comparisons of different work, but they do not replace individual metrics that assess specific properties and can be counselled in a follow-up investigation.

When we observe the trend of our unified measures over the last years, we see network intrusion detection has made clear and significant progress to date, however there remain evident challenges for the future. These challenges are in line with the challenges first formulated in [34], and remain relevant for practical applications. The important challenges we address in the following section are the difficulty of feature extraction and comparison across datasets.

VII. WORKFLOW AND COMPARATIVE EXPERIMENTS

Currently, most of the machine learning-based NIDS research is conducted on complete datasets using precomputed features. This approach, however, is not representative of real-time environments. In an on-line, high-speed networking environment, an NIDS needs to be able to inspect incoming traffic at high rates, lest it causes congestion or does not inspect all traffic. A system might therefore not have ample time to capture an entire flow for analysis, which is necessary to find some of the necessary features. For example, in the UNSW-NB15 features, *sload* indicates the number of source bits per second, and *ct_flow_http_mthd* tracks the number of Hypertext Transfer Protocol (HTTP) methods in the traffic. Tracking such features in real-time is problematic for a number of reasons:

1) Memory efficiency: As it is unknown when the flow will end, it is unknown how long the NIDS needs to keep specific data in memory. If many different flows are being transmitted simultaneously, the system would need to keep many features in memory, exceeding the available memory capacity or occupying memory that is necessary in other parts of the detection pipeline.
2) Calculation overhead: Many features need to be calculated from incoming data flows, requiring time and resources not going to the actual detection process. For example, most of the 48 features in UNSW-NB15 require counters, timers or application layer interpretation of data.
3) Detection delay: Following the principle of keeping a flow in memory until it has been completely transmitted would also imply to delay detection until that point in time. For flows that are spread over a longer period,[8] it takes too much time before an attack is detected.

An attacker could easily abuse the system by starting many flows and never ending them to throttle IDS resources, or by having an attack flow that never stops in order to avoid detection. In practice, one workaround is to set (timing) thresholds that determine when to process an ongoing flow. This would however introduce noise in the detection process, as this process would be based on the original, non-thresholded features instead of their thresholded counterparts. While methods might exist to implement an NIDS based on those traditional features, in this section we explore a different approach: Raw network traffic-based features.

For this approach, raw network packets and flows are used as ML features, rather than their derived characteristics. One example of this is the work presented in [85], where the

[8]Some flows in UNSW-NB15 are spread over more than 2 hours.

first n bytes of 5 consecutive packets in the same flow are used to form a square image. As this image is directly fed into a CNN, the only feature selections are the number of packet bytes and the number of packets to include. Clearly, using this approach minimizes feature extraction overhead. Instead of keeping track of many potentially long lasting flows, the first n bytes of incoming packets are used instead. This introduces another advantage, namely that these features are dataset agnostic: They can be extracted from any network traffic, thus allowing for evaluation on different datasets. On the contrary, traditional features are dataset-dependent, complicating inter-dataset comparisons.

Starting from this observation, we built a workflow[9] to alleviate the comparison of raw traffic-based network intrusion detection algorithms. This workflow features the use of different recent and publicly available datasets, namely ISCX2012, UNSW-NB15, CICIDS2017. Moreover, we implemented novel algorithms such as HAST-II, [79] and PCCN [85] for each of these datasets. Both algorithms boast high reported performance, for ISCX2012 and CICIDS2017 respectively. Implementing them in our experiments allows for easy comparison of on three different datasets. This section will discuss the functioning of the workflow and present results of the comparison between different configurations of raw traffic-based network intrusion detection algorithms.

A. PROPOSED WORKFLOW

Our workflow is inspired by the workflow presented in [85], but is more elaborate and generic. It also uses other methods to extract the relevant data, for example custom Python code instead of tshark[10] for parsing, and will be made fully available online. In order to be manageable, the workflow is divided into 3 large steps: Network traffic processing to extract and label flows, feature extraction and intrusion detection, as visualized in Fig. 2a.

Typically, raw dataset traffic is stored inside PCAP and PCAPNG files[11] that first need to be decoded before they can be used. Decoding PCAP files yields a chronological sequence of packets with additional metadata such as timestamps. Extracted packets can then be sorted according to their IP source and destination addresses as well as their TCP/UDP source and destination ports and their protocol number. These 5 parameters, sometimes referred to as *5-tuple*, constitute a flow identifier (flow ID).

After sorting all packets according to their flow ID, they need to be labelled. The datasets used in this workflow provided labelling information in a unidirectional fashion for flows based on their flow ID. If a flow ID is represented by (A, B, x, y, p), where A and B are source and destination addresses, x and y are source and destination ports and p is the transport layer protocol, the unidirectionally labelled flows comprise both the (A, B, x, y, p) as well as the (B, A, y, x, p) traffic. As different flows may have identical flow IDs, in the case that they were transmitted at different points in time, it is required to distinguish between packets of different flows with the same flow ID. By only selecting the first k packets for a flow ID, where k is number of forward or backward packets specified for each flow, we differentiate between different flows. These k packets are then removed from the tree, assuring that the next packets correspond to the next flow. This approach depends on the chronological order of packets in PCAP(NG) files. Note that for some flows, the indicated number of forward and backward packets differs from the actual captured packets. Similarly, some flow IDs featured in the labelling information are not featured at all in captured traffic files. Both inconsistencies might introduce some noise in the dataset.

Once all eligible[12] individual flows have been labelled, the required features for each flow can be extracted in the second step of the workflow. While this procedure is specific for each feature extraction strategy, all extraction strategies utilize the packet bytes in a flow. For both PCCN [85] and HAST-II [79], these bytes are structured in 2D images, with HAST-II also creating sequences of such images.

Finally, after acquiring the features, the corresponding ML algorithms can be trained, validated and tested in the final step. For PCCN this algorithm is a CNN while for HAST-II a CNN and an LSTM are combined.

B. EXPERIMENTS AND RESULTS

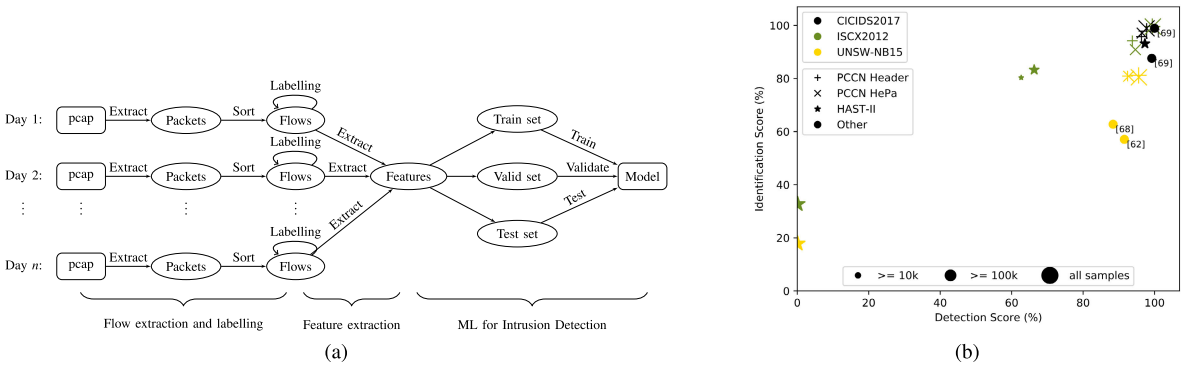
We implemented the proposed workflow, and extracted the required features from the raw flow data as described for PCCN [85] and HAST-II [79]. All code is written in Python, using PyTorch for the implementation of the ML algorithms. The train/validation/test split of the dataset was 75%/15%/10%. The initial learning rate of each experiment was 0.01 or 0.001, which was divided by 10 every time the loss stagnated for 10 epochs. Every experiment used a batch size of 256, and ran for at least 35 epochs, depending on how fast it converged. The specific hyperparameters for each experiment can be retrieved from https://gitlab.com/EAVISE/raw-traffic-nids. In an effort to reduce class imbalance, a constraint was introduced during training. This constraint limits the maximum number of training samples to use per class, in order to get a more even distribution during training. No changes were made in the testing distribution. Besides introducing these constraints, we also aggregated the samples of the three separate web attacks in CICIDS2017 to further combat class imbalance. The results following these experiments are presented in Table 4 and visualized in Fig. 2b.

[9]Software code publicly available at https://gitlab.com/EAVISE/raw-traffic-nids

[10]https://www.wireshark.org/docs/man-pages/tshark.html

[11]Not the case for KDD99 and NSL-KDD, which is why these datasets cannot be included in this workflow and the described experiments.

[12]Not all traffic flows in a PCAP(NG) file are labelled, as some TCP/IP flows remain unlabelled. All other traffic, such as for example Internet Message Control Protocol (ICMP) packets, is disregarded and not even included in the sorting and labelling phase.



(a) (b)

FIGURE 2. **Proposed workflow for raw traffic-based feature extraction (left) and experimental results as well as other published results (right).**

TABLE 4. **Experimental results for each dataset, feature extraction strategy and ML network. The best value for each metric is emboldened. *Constraint* indicates the maximum number of samples per class during training.**

				Weighted averages					Proposed scores			
Features	Classes	Network	Constraint	Acc	Prec	DR	FPR	F1	DS	F_w	F_M	IS
ISCX2012												
HAST-II	5	HAST-II	All	97.17	94.41	97.17	97.17	95.77	0.00	95.77	19.71	32.70
HAST-II	5	HAST-II	12000	96.63	98.46	96.63	**0.33**	97.23	62.67	97.23	68.42	80.32
HAST-II	5	HAST-II	100000	97.13	98.56	97.13	0.52	97.59	66.31	97.59	72.62	83.27
PCCN Header	5	PCCN	All	99.89	99.89	99.89	0.58	99.89	99.36	99.89	**99.26**	**99.58**
PCCN Header	5	PCCN	100000	98.91	99.05	98.91	0.39	98.95	93.78	98.95	89.86	94.18
PCCN HePa	5	PCCN	All	**99.92**	**99.92**	**99.92**	0.52	**99.92**	**99.52**	**99.92**	99.20	99.56
PCCN HePa	5	PCCN	100000	99.07	99.24	99.07	0.52	99.12	94.63	99.12	83.87	90.86
UNSW_NB15												
HAST-II	10	HAST-II	All	96.65	93.40	96.65	96.65	95.00	0.00	95.00	9.83	17.82
HAST-II	10	HAST-II	50000	96.65	93.40	96.65	96.65	95.00	0.00	95.00	9.83	17.82
HAST-II	10	HAST-II	100000	96.65	93.40	96.65	96.65	95.00	0.00	95.00	9.83	17.82
PCCN Header	10	PCCN	All	**99.66**	**99.65**	**99.66**	5.94	**99.65**	95.54	**99.65**	**68.47**	**81.17**
PCCN Header	10	PCCN	400000	99.48	99.64	99.48	**0.02**	99.53	92.44	99.53	68.19	80.93
PCCN HePa	10	PCCN	All	99.65	99.64	99.65	5.38	99.64	**95.61**	99.64	67.46	80.45
PCCN HePa	10	PCCN	400000	99.48	**99.65**	99.48	0.22	99.54	92.75	99.54	68.06	80.84
CICIDS2017												
HAST-II	13	HAST-II	200000	98.95	99.08	98.95	0.07	98.98	97.29	98.98	87.89	93.11
PCCN Header	13	PCCN	All	99.51	99.51	99.51	1.56	99.51	**97.77**	99.51	96.25	97.85
PCCN Header	13	PCCN	100000	99.17	99.31	99.17	**0.03**	99.21	96.33	99.21	92.62	95.80
PCCN HePa	13	PCCN	All	**99.56**	**99.56**	**99.56**	1.53	**99.56**	97.76	**99.56**	**98.23**	**98.89**
PCCN HePa	13	PCCN	100000	99.28	99.41	99.28	**0.03**	99.32	96.51	99.32	94.99	97.10

C. DISCUSSION

From the results, it is clear that using the PCCN approach yields the highest results, with both the header-based as well as the header and payload-based approaches (HePa) reaching very high DS and IS values. HePa appears to be about 1% better regarding the IS when compared to the header approach for CICIDS2017 and UNSW-NB15. This however is not the case for ISCX2012, as both approaches are very near 100%. Furthermore, the HAST-II approach obtains significantly worse results in comparison, as for unconstrained training these models classify everything as normal traffic. While applying constraints alleviates this issue somewhat, the PCCN-based approaches still excel. This can be partially attributed to the number of features generated in the HAST-II approach. HAST-II only uses the first 6 packets in a flow, while all PCCN-based approaches use all available flow packets. Concretely, the number of HAST-II features in a dataset is proportional to the number of flows in that dataset, while the number of PCCN features is proportional to the number of packets. Therefore, while both approaches generate a very large amount of benign features, only PCCN approaches generate sufficient attack features. Besides these observations, it is also clear that the results for UNSW-NB15 are significantly lower than those obtained for CICIDS2017 and ISCX2012. As this can also be observed in [47] and for the comparative Tables 2 and 3, UNSW-NB15 is more challenging, and thus of higher academic interest. When comparing the experimental results against the corresponding results in tables 2 and 3, it is clear that raw traffic-based network intrusion detection is able to

compete with detection methods based on traditional features. However, the raw traffic-based approach also raises some concerns: The first n bytes of packets in a flow may not contain the information that is necessary for reliable detection. Moreover, it might be possible that different datasets (or networking environments) are different in such a way that they require different features. More ideally, it might be possible to combine both raw traffic-based as well as certain traditional features into one approach, in an effort to provide more information in a format that can be efficiently extracted. More research is necessary to investigate this avenue.

VIII. CONCLUSION

In this paper we provide an overview of the state of the art of ML-based network intrusion detection, and discuss a number of issues within this area. These issues include the non-standardized reporting of results as well as the outdatedness and imbalance of most network intrusion detection datasets. We propose the detection score and identification score metrics as reliable methods of fair evaluation, and strongly encourage their use in future work. We also enrolled these metrics in our comparison of both existing work as well as our own experiments and show that they are able to accurately display NIDS performance. Finally, we extended the work of previous research by implementing previously published raw traffic-based detection approaches in a new workflow for multiple datasets to compare against other research. These experiments show promising results, and provide more insight in the ability of ML algorithms for real-time network intrusion detection purposes.

Future work on NIDSs should consider the challenges we discuss in this paper. This may include further investigation of the use of alternative features, as well as the ability of an NIDS to work in a real-world scenario, not only for synthesized datasets. Addressing these issues will significantly contribute to the adoption of ML-based network intrusion detection.

APPENDIX A
EVALUATION METRICS BACKGROUND

In this appendix, we introduce the metrics that are commonly used for network intrusion detection evaluation (A-A). Moreover, we also consider the averaging options in a multiclass scenario (A-B). Lastly, we concisely describe the function and construction of a confusion matrix.

A. BINARY CLASSIFICATION

For a dataset of arbitrary size, each element belongs to a specific class. The entity of classes can either be binary or n-ary. In the case of binary classification, an element is either benign or an attack. Usually, the attack is denoted as positive, while the benign class is negative. Considering this, a true positive (tp) is an element of a positive class that is predicted to be positive by an algorithm. Similarly, true negatives (tn) are elements of a negative class that are correctly predicted to be negative. On the contrary, false positives (fp) or false negatives (fn) are elements that are respectively negative or positive, but are wrongly predicted to be their opposites. By indicating tp, tn, fp and fn to have a value equal to the number of corresponding elements after a series of classifications or predictions, five metrics can be defined.

The accuracy simply is the fraction of elements that was predicted correctly:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2}$$

Precision, sometimes also denoted as *Positive Predictive Value*, signifies the fraction of elements that were correctly predicted of all elements predicted to be attacks.

$$Precision = \frac{tp}{tp + fp} \tag{3}$$

While precision resembles how many elements were correctly classified as positive, the *detection rate* (DR) indicates how many attacks were detected, as shown in Eq. 4. In the literature, this metric is also referred to as *recall*, *sensitivity* or *true positive rate*.

$$DR = \frac{tp}{tp + fn} \tag{4}$$

On the contrary, the *false positive rate* (FPR) shows the amount of attacks that would have gone unnoticed in the same scenario. It is defined in Eq. 5.

$$FPR = \frac{fp}{tn + fp} \tag{5}$$

Finally, the *F-measure*, *F1-score* or *F-score* provides the harmonic mean of the precision and the recall. This measure penalizes low values of either precision or recall, requiring both to be high to result in a high output value.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6}$$

The F-measure can also be generalized to the F_β*-measure*, where the β value can be used to tweak the weight of either the precision or the recall relative to the other metric.

$$F_\beta = \left(1 + \beta^2\right) \frac{Precision \cdot Recall}{\left(\beta^2 \cdot Precision\right) + Recall} \tag{7}$$

In eq. 7, the weight of the precision is altered.

B. n-ARY CLASSIFICATION

While these metrics are straightforward in a binary situation, caution is required in scenarios where attacks are represented in multiple classes. As the metrics are calculated for each class individually by considering each other class as negative, a number of values is obtained for every metric. These values then have to be aggregated into a singe value reflecting the performance of the classification task for that metric.

Commonly, three averaging approaches serve this purpose, namely micro-averaging (denoted with μ), macro-averaging (denoted with M) and weighted averaging (denoted with w) Consider the precision for binary classification as given in Eq. 3, used in a multiclass scenario with l classes, including normal traffic. In micro-averaging, the resulting $Precision_\mu$ consists of the sum of the tp per class divided by the sum of all tp and fp per class [123].

$$Precision_\mu = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fp_i)} \tag{8}$$

The macro averaged precision then simply is the arithmetic mean of all precision values:

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l} \tag{9}$$

Finally, for a dataset of N samples, where n_i is the number of samples for class i, the weighted average of the precision is:

$$Precision_w = \sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i} \cdot \frac{n_i}{N} \tag{10}$$

While weighted and micro averages are biased towards larger classes, macro averages consider each class to be equal [123]. From this point, we will only consider weighted and macro averages, as micro averages are very similar to weighted averages but appear less often in NIDS literature. While we only provide the equations for the precision, we can similarly average recall and F-measure results.

C. CONFUSION MATRIX

Confusion matrices are a useful tool to present and evaluate the classification performance of any classification algorithm. Each column in the square matrix presents the number of samples that were predicted as members of a corresponding class. Each row then describes to what actual class those samples belonged. Ideally, the matrix is a diagonal matrix, which means that all samples are predicted correctly. A confusion matrix structure is given in Fig. 1, where we demonstrate how to map the multiclass scenario to a binary scenario as described in Sect. V.

REFERENCES

[1] (Mar. 2020). *Cisco Annual Internet Report—Cisco Annual Internet Report (2018–2023) White Paper—Cisco*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
[2] (Feb. 2016). *Massive Brute-Force Attack on Alibaba Affects Millions*. [Online]. Available: https://www.infosecurity-magazine.com/news/massive-bruteforce-attack-on
[3] (Oct. 2016). *Dyn Analysis Summary of Friday October 21 Attack*. [Online]. Available: https://web.archive.org/web/20200620203923/ and https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/
[4] H. Debar, "An introduction to intrusion-detection systems," in *Proc. Connect*, 2002, pp. 1–18.
[5] J. P. Anderson, "Computer security threat monitoring and surveillance," James P. Anderson Co., Washington, DC, USA, Tech. Rep., 1980.
[6] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.
[7] S. Wunderlich, M. Ring, D. Landes, and A. Hotho, "Comparison of system call representations for intrusion detection," in *Proc. Int. Joint Conf., 12th Int. Conf. Comput. Intell. Secur. Inf. Syst. (CISIS), 10th Int. Conf. Eur. Transnational Educ. (ICEUTE)*, F. M. Álvarez, A. T. Lora, J. A. S. Muñoz, H. Quintián, and E. Corchado, Eds. Cham, Switzerland: Springer, 2020, pp. 14–24, doi: 10.1007/978-3-030-20005-3_2.
[8] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–33, Jul. 2015, doi: 10.1145/2716260.
[9] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A trust-aware, P2P-based overlay for intrusion detection," in *Proc. 17th Int. Conf. Database Expert Syst. Appl. (DEXA)*, 2006, pp. 692–697.
[10] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating security managers: A peer-based intrusion detection system," *IEEE Netw.*, vol. 10, no. 1, pp. 20–23, Jan. 1996.
[11] C. J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust management for host-based collaborative intrusion detection," in *Managing Large-Scale Service Deployment*, F. De Turck, W. Kellerer, and G. Kormentzas, Eds. Berlin, Germany: Springer, 2008, pp. 109–122.
[12] W. Li, W. Meng, Y. Wang, J. Han, and J. Li, "Towards securing challenge-based collaborative intrusion detection networks via message verification," in *Information Security Practice and Experience*, C. Su and H. Kikuchi, Eds. Cham, Switzerland: Springer, 2018, pp. 313–328.
[13] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517300802
[14] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870513001005
[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," 2018, *arXiv:1804.07474*. [Online]. Available: https://arxiv.org/abs/1804.07474
[16] K. Khan, A. Mehmood, S. Khan, M. A. Khan, Z. Iqbal, and W. K. Mashwani, "A survey on intrusion detection and prevention in wireless ad-hoc networks," *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101701. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762119305089
[17] B. Riyaz and S. Ganapathy, "A deep learning approach for effective intrusion detection in wireless networks using CNN," *Soft Comput.*, vol. 24, no. 22, pp. 17265–17278, Nov. 2020, doi: 10.1007/s00500-020-05017-0.
[18] R. Vijayanand and D. Devaraj, "A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network," *IEEE Access*, vol. 8, pp. 56847–56854, 2020.
[19] M. Safaldin, M. Otair, and L. Abualigah, "Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 2, pp. 1559–1576, Jun. 2020, doi: 10.1007/s12652-020-02228-z.
[20] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101752. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820300365
[21] S. Anwar, J. M. Zain, M. F. Zolkipli, Z. Inayat, S. Khan, B. Anthony, and V. Chang, "From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions," *Algorithms*, vol. 10, no. 2, p. 39, Mar. 2017.
[22] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja, *Network Intrusion Detection Using Deep Learning* (Springer Briefs on Cyber Security Systems and Networks). Singapore: Springer, 2018. [Online]. Available: http://www.springer.com/series/15797 and http://link.springer.com/10.1007/978-981-13-1444-5
[23] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, pp. 1–99, Dec. 2018.
[24] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "New anomaly network intrusion detection system in cloud environment based on optimized back propagation neural network using improved genetic algorithm," *Int. J. Commun. Netw. Inf. Secur.*, vol. 11, pp. 61–84, Apr. 2019.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.
[26] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804515002891
[27] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[28] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[29] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, Apr. 2019, doi: 10.3390/info10040122.
[30] S. Mahdavifar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231219302954
[31] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.
[32] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.
[33] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212619305046
[34] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, Los Alamitos, CA, USA, May 2010, pp. 305–316.
[35] *1998 DARPA Intrusion Detection Evaluation Dataset*. Accessed: Jan. 5, 2021. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset
[36] R. Lippmann, R. Cunningham, D. Fried, I. Graf, K. Kendall, S. Webster, and M. Zissman, "Results of the DARPA 1998 offline intrusion detection evaluation," in *Proc. 2nd Int. Workshop Recent Adv. Intrusion Detection*, Jan. 1999, pp. 1–29. [Online]. Available: http://www.raid-symposium.org/raid99/
[37] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000, doi: 10.1145/382912.382923.
[38] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection," in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds. Berlin, Germany: Springer, 2003, pp. 220–237.
[39] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[40] *KDD Cup 1999 Data*. Accessed: Jan. 5, 2021. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[41] H. Xu, Q. Cao, H. Fu, C. Fu, H. Chen, and J. Su, "Application of support vector machine model based on an improved elephant herding optimization algorithm in network intrusion detection," in *Artificial Intelligence*, K. Knight, C. Zhang, G. Holmes, and M.-L. Zhang, Eds. Singapore: Springer, 2019, pp. 283–295.
[42] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proc. ACM CSS Workshop Data Mining Appl. Secur. (DMSA)*, 2001, pp. 5–8.
[43] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404811001672
[44] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[45] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, Dec. 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128699001127
[46] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016, doi: 10.1080/19393555.2015.1125974.
[47] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[48] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Portugal, Jan. 2018, pp. 108–116.
[49] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 479–482, 2018.
[50] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surv.*, vol. 49, no. 2, pp. 1–50, Nov. 2016, doi: 10.1145/2907070.
[51] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognit.*, vol. 91, pp. 216–231, Jul. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320319300950
[52] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320396001422
[53] A. A. Cardenas, J. S. Baras, and K. Seamon, "A framework for the evaluation of intrusion detection systems," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2006, p. 15.
[54] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.
[55] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 313–316.
[56] R. Beghdad, "Training all the KDD data set to classify and detect attacks," *Neural Netw. World*, vol. 17, pp. 81–91, Jun. 2007.
[57] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.
[58] L. Zhang, M. Li, X. Wang, and Y. Huang, "An improved network intrusion detection based on deep neural network," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 563, Aug. 2019, Art. no. 052019, doi: 10.1088%2F1757-899x%2F563%2F5%2F052019.
[59] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1185–1190, Sep. 2012, doi: 10.1007/s00521-010-0487-0.
[60] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, pp. 136–154, Jul. 2015.
[61] Y. Yu, Z. Ye, X. Zheng, and C. Rong, "An efficient cascaded method for network intrusion detection based on extreme learning machines," *J. Supercomput.*, vol. 74, no. 11, pp. 5797–5812, Nov. 2018.
[62] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 195–200.
[63] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Syst. Appl.*, vol. 67, pp. 296–303, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416305310
[64] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[65] B. Hu, J. Wang, Y. Zhu, and T. Yang, "Dynamic deep forest: An ensemble classification method for network intrusion detection," *Electronics*, vol. 8, no. 9, p. 968, Aug. 2019, doi: 10.3390/electronics8090968.
[66] C. Lu, L. Zhai, T. Liu, and N. Li, "Network intrusion detection based on neural networks and D-S evidence," in *Image and Video Technology—PSIVT 2015 Workshops*, F. Huang and A. Sugimoto, Eds. Cham, Switzerland: Springer, 2016, pp. 332–343.

[67] Y. Yu, S. Kang, and H. Qiu, "A new network intrusion detection algorithm: DA-ROS-ELM," *IEEJ Trans. Electr. Electron. Eng.*, vol. 13, no. 4, pp. 602–612, Apr. 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/tee.22606
[68] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[69] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 858–866.
[70] T. Kim, S. C. Suh, H. Kim, J. Kim, and J. Kim, "An encoding technique for CNN-based network anomaly detection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2960–2965.
[71] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1595–1598.
[72] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.
[73] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Syst. Appl.*, vol. 124, pp. 196–208, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419300843
[74] N. Chouhan, A. Khan, and H.-U.-R. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105612. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619303928
[75] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proc. Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2015, pp. 339–344.
[76] Y. Ding and Y. Zhai, "Intrusion detection system for NSL-KDD dataset using convolutional neural networks," in *Proc. 2nd Int. Conf. Comput. Sci. Artif. Intell. (CSAI)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 81–85, doi: 10.1145/3297156.3297230.
[77] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Appl. Sci.*, vol. 9, no. 2, p. 238, Jan. 2019, doi: 10.3390/app9020238.
[78] A. S. Qureshi, A. Khan, N. Shamim, and M. H. Durad, "Intrusion detection using deep sparse auto-encoder and self-taught learning," *Neural Comput. Appl.*, vol. 32, pp. 1–13, Mar. 2019.
[79] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[80] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection," *Comput. Netw.*, vol. 144, pp. 111–119, Oct. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618306005
[81] N. Aboueata, S. Alrasbi, A. Erbad, A. Kassler, and D. Bhamare, "Supervised machine learning techniques for efficient network intrusion detection," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–8.
[82] L. Zhiqiang, G. Mohi-Ud-Din, L. Bing, L. Jianchao, Z. Ye, and L. Zhijun, "Modeling network intrusion detection system using feed-forward neural network using UNSW-NB15 dataset," in *Proc. IEEE 7th Int. Conf. Smart Energy Grid Eng. (SEGE)*, Aug. 2019, pp. 299–303.
[83] W. Zong, Y.-W. Chow, and W. Susilo, "A two-stage classifier approach for network intrusion detection," in *Information Security Practice and Experience*, C. Su and H. Kikuchi, Eds. Cham, Switzerland: Springer, 2018, pp. 329–340.
[84] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, Mar. 2019.
[85] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "PCCN: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows," *IEEE Access*, vol. 7, pp. 119904–119916, 2019.
[86] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[87] W. Hu, Y. Liao, and V. R. Vemuri, "Robust support vector machines for anomaly detection in computer security," in *Proc. Int. Conf. Mach. Learn. Appl. (ICMLA)*, M. A. Wani, K. J. Cios, and K. Hafeez, Eds., Los Angeles, CA, USA, Jun. 2003, pp. 168–174.
[88] Y. Yi, J. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7698–7707, Jun. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410015046
[89] S. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining Knowl. Discovery*, vol. 2, pp. 345–389, Mar. 2000.
[90] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.
[91] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231206000385
[92] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 599–619, doi: 10.1007/978-3-642-35289-8_32.
[93] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, Dec. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213005547
[94] M. R. G. Raman, N. Somu, K. Kirthivasan, R. Liscano, and V. S. S. Sriram, "An efficient intrusion detection system based on hypergraph–genetic algorithm for parameter optimization and feature selection in support vector machine," *Knowl.-Based Syst.*, vol. 134, pp. 1–12, Oct. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705117303209
[95] R. Jain and N. S. Abouzakhar, "Hidden Markov model based anomaly intrusion detection," in *Proc. Int. Conf. Internet Technol. Secured Trans.*, Dec. 2012, pp. 528–533.
[96] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl.-Based Syst.*, vol. 78, pp. 13–21, Apr. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705115000167
[97] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proc. ACSC*, 2005, pp. 1–10.
[98] I. Almomani, B. A. Kasasbeh, and M. Al-Akhras, "WSN-DS: A dataset for intrusion detection systems in wireless sensor networks," *J. Sensors*, vol. 2016, pp. 4731953:1–4731953:16, Aug. 2016.
[99] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+dataset for NIDS evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exper. Returns Secur. (BADGERS)*. New York, NY, USA: Association for Computing Machinery, 2011, pp. 29–36, doi: 10.1145/1978672.1978676.
[100] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[101] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[102] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[103] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://www.aclweb.org/anthology/D14-1181
[104] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. 6th Int. Conf. Co-NEXT*. New York, NY, USA: Association for Computing Machinery, 2010, pp. 1–12, doi: 10.1145/1921168.1921179.

[105] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 858–866.
[106] P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection," in *Proc. IEEE 4th Int. Conf. Big Data Anal. (ICBDA)*, Mar. 2019, pp. 281–285.
[107] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 4, pp. 694–707, Apr. 2016.
[108] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, Mar. 2017.
[109] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 275, pp. 167–179, Jan. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217309505
[110] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 80–1735, 1997.
[111] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[112] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*. Madison, WI, USA: Omnipress, 2011, pp. 1017–1024.
[113] H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, May 1992, pp. 240–250.
[114] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Neural-Network Models of Cognition* (Advances in Psychology), vol. 121, J. W. Donahoe and V. P. Dorsel, Eds. North-Holland, 1997, ch. 25, pp. 471–495. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166411597801112, doi: 10.1016/S0166-4115(97)80111-2.
[115] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1
[116] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 893–898.
[117] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.
[118] J. Kim, J. Kim, H. L. Thi Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.
[119] G. Hinton, *Deep Belief Nets*. Boston, MA, USA: Springer, 2010, pp. 267–269, doi: 10.1007/978-0-387-30164-8_208.
[120] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.
[121] Y. Yu, J. Long, and Z. Cai, "Session-based network intrusion detection using a deep learning architecture," in *Modeling Decisions for Artificial Intelligence*, V. Torra, Y. Narukawa, A. Honda, and S. Inoue, Eds. Cham, Switzerland: Springer, 2017, pp. 144–155.
[122] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814000923
[123] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, Jul. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457309000259

LAURENS LE JEUNE received the B.Sc. and M.Sc. degrees in electronics and ICT engineering technology in a joint program of KU Leuven, Leuven, Belgium, and Hasselt University, Diepenbeek, Belgium, in 2018 and 2019, respectively. He is currently pursuing the Ph.D. degree in engineering technology with KU Leuven. For his master's thesis, he investigated the classification of camera trap wildlife footage for biodiversity research. In his Ph.D. research, he works for the Embedded Systems and Security (ES&S) as well as the Embedded and Artificially Intelligent Vision Engineering (EAVISE) research groups. He is also investigating the application of deep learning technology for hardware-accelerated real-time network intrusion detection. His research interests include machine learning and deep learning, FPGAs, network security, and intrusion detection.

TOON GOEDEMÉ studied electrical engineering at KU Leuven. He received the Ph.D. degree in vision-based topological navigation from KU Leuven, in December 2006, under the guidance of Prof. L. Van Gool and T. Tuytelaars. Afterwards, he started teaching at the Technical University De Nayer, Sint-Katelijne-Waver, where he founded his research group Embedded and Artificially Intelligent Vision Engineering (EAVISE), in 2008. Nowadays, his group is integrated in the KU Leuven and consists of three professors (Joost Vennekens, Patrick Vandewalle, and himself), four postdocs and about 20 researchers, playing a vital role in the transfer of computer vision and AI know-how from academic research towards the industry. Since 2014, he has been an Associate Professor with KU Leuven. He is the (co)author of more than 190 international publications and was a project leader of more than 75 industrially co-founded research projects. Together with his team, he won several awards, such as the Best Paper Award at Embedded Vision Workshop CVPR 2015, the Best Demo Award at BNAIC 2015, the Best Paper Award at CGVCVIP 2016, the Willy Asselman Award for research achievements in 2016, and the Best Paper Award at Embedded Vision Workshop ECCV 2020. He is also an Associate Editor of the *IET Computer Vision* journal and the MDPI *Journal of Imaging*.

NELE MENTENS (Senior Member, IEEE) received the master's and Ph.D. degrees from KU Leuven, in 2003 and 2007, respectively. She was a Visiting Researcher with Ruhr University Bochum, in 2013, and with EPFL, in 2017. She is currently a Professor with Leiden University and KU Leuven. She is the (co)author in over 100 publications in international journals, conferences, and books. She was/is the PI in around 20 finished and ongoing research projects with national and international funding. Her research interests include the domains of configurable computing for security, hardware acceleration of network security applications, and security in constrained environments. She serves as a program committee member for renowned international conferences on security and hardware design, such as NDSS, Usenix Security Symposium, CHES, DAC, DATE, FPL, and ESWEEK. She was the General Co-Chair of FPL, in 2017, the Program Chair of EWME and PROOFS, in 2018, and the Program Chair of FPL and CARDIS, in 2020. She also serves as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and *IEEE Circuits and Systems Magazine*.