SoK: Pragmatic Assessment of Machine Learning for Network Intrusion Detection

Giovanni Apruzzese, Pavel Laskov, Johannes Schneider
School of Business Informatics – University of Liechtenstein
{*giovanni.apruzzese, pavel.laskov, johannes.schneider*}*@uni.li*

***Abstract*—Machine Learning (ML) has become a valuable asset to solve many real-world tasks. For Network Intrusion Detection (NID), however, scientific advances in ML are still seen with skepticism by practitioners. This disconnection is due to the intrinsically somewhat limited scope of research papers, many of which primarily aim to demonstrate new methods "outperforming" prior work—oftentimes overlooking the practical implications for deploying the proposed solutions in real systems. Therefore, the value of ML for NID depends on a plethora of factors, such as hardware, that are often neglected in scientific literature.**

This paper aims to reduce the practitioners' skepticism towards ML for NID by *changing* the evaluation methodology adopted in research. After elucidating which *factors* influence the operational deployment of ML in NID, we propose the notion of *pragmatic assessment*, which enable practitioners to gauge the real value of an ML method for NID. Then, we show that the state-of-research hardly allows one to estimate the value of ML for NID. As a constructive step forward, we carry out a pragmatic assessment. We reassess existing ML methods for NID, focusing on the classification of malicious network traffic, and consider hundreds of configuration settings, diverse adversarial scenarios, and four hardware platforms. Our large and reproducible evaluations enable estimating the quality of ML for NID. We also validate our claims through a user-study with security practitioners.

***Index Terms*—Cybersecurity, Machine Learning, Intrusion Detection, Deployment, Development, Network**

1. Introduction

Machine learning (ML) techniques have become an indispensable technology in many domains of computer science, such as computer vision [1], [2], natural language processing [3], audio and speech recognition [4], medical applications [5], and increasingly in cybersecurity, e.g., malware analysis [6], spam and phishing prevention [7], as well as network intrusion detection (NID). As stated by Arp et al. at the beginning of their paper: "No day goes by without reading machine learning success stories" [8].

However, deployment of ML methods in NID faces substantial skepticism [9]–[11] among practitioners—despite the fact that NID is one of the oldest applications of ML in cybersecurity [12]–[14]. The main difficulty, as pointed out by Sommer and Paxson [15], is that network environments exhibit "immense variability". Hence, most ML models for NID developed in research papers cannot be readily transferred into operational environments due to a large uncertainty about their genuine value. In the real world, what matters is not the improvements over prior work, but rather the balance between the performance and costs in routine deployment scenarios.

The main thesis of this paper is that *research evaluations of ML in NID should account for pragmatic aspects of operational deployment*. We elucidate all such aspects by proposing the notion of "pragmatic assessment", whose goal is ensuring that practitioners have all the necessary information to determine whether a given ML solution is applicable to a given NID context. From the viewpoint of researchers, conducting such pragmatic assessments is challenging: almost every paper on ML in NID published in recent top-conferences has some shortcomings. However, as we will show in this work, *it can be done*. As such, we endorse future efforts to adopt our proposed takeaways so as to facilitate the integration of ML research results into real network intrusion detection systems (NIDS).

Motivational Example. Let us elucidate why the state-of-the-art of ML in NID is still at an early stage with respect to practical deployment. For this purpose, we compare NID with two popular domains in which ML has found applications: computer vision and malware analysis.

In **computer vision**, the evaluation methodology adopted in research is now standardized. Current benchmarks, e.g., `ImageNet` [16], were created before 2010 and are still widely used today—even in production [17], because they contain data from the 'real' world. Abundant literature[1] implicitly established reference standards,[2] thereby removing the uncertainty on the real value of the findings obtained in research environments. A similar case can be said for **malware analysis**. After the release of `Drebin` [20] in 2014, containing real Android apps (benign and malicious), abundant[3] ML research has been carried out on `Drebin` (e.g., [21]–[23]). Agreeably, `Drebin` is not perfect (some papers found some 'duplication' issues [24]), and some malware families are not popular anymore. However, a crucial fact remains: malware is malicious *everywhere* and *everytime* [25]. Therefore, a proficient ML model trained on `Drebin` can be deployed on any system analyzing Android apps. For instance, [26] show that the method originally proposed in [20] is effective even on (real!) apps collected in 2017–2019.

1. As of March 2022, [16] has over 35K Google Scholar citations.
2. E.g., *ResNet* [18] models are known to consistently achieve very high *accuracy* on `ImageNet`, which now represent the reference benchmark for computer vision (even on different datasets, e.g., `CIFAR` [19]).
3. As of March 2022, [20] has over 2K citations on Google Scholar.

In contrast, **ML research on NID is far from such maturity**. Among the root causes is the lack of (open-source) data that is representative of the real world. For instance, thousands of proposals were validated on the KDD99 dataset, usually achieving near-perfect performance [27]. However, the data in KDD99 represents only a single network (from 1999), preventing to estimate the generalization capabilities of ML solutions [28]. We observe that, recently, more datasets are being publicly released (e.g. [29]). Surprisingly, however, such 'abundance' increased confusion. Consider, e.g., the recent findings of ML proposals evaluated on the popular CICIDS17 dataset [30]. Specifically, we focus on [31] and [32]—both involving ML models based on diverse ML algorithms, including Decision Trees (DT) and Neural Networks (NN). While [31] claims that NN are better than DT—as given by a superior F1-score (0.96 for NN, 0.95 for DT)—the opposite occurs in [32]—with the DT reaching 0.99 F1-score, against the 0.96 of the NN. Moreover, it is misleading to only consider a single performance metric for NID: even a high F1 score may conceal a suboptimal false positive rate, making a given ML solution impractical for realistic deployments [8]. To make all of this worse, recent efforts found that CICIDS17 is flawed [33]. Finally, the authors of [34] showed that ML models trained on the (fixed) version of CICIDS17 perform poorly against 'unknown' attacks. The current situation of ML in NID can be summarized with a statement from Markus de Shon (who was Lead of the Detection Engineering at NetFlix): "Application of ML in intrusion detection has been uneven at best, with deep and widespread (and generally justified) skepticism among subject matter experts" [9].

CONTRIBUTION. Our aim is *changing* the evaluation methodology adopted by research on ML for NID so as to remove the skepticism of practitioners towards the quality of scientific solutions. To reach our goal, we first summarize Machine Learning-based Network Intrusion Detection Systems (ML-NIDS). Then we provide four major contributions—each discussed in a dedicated section (§) revolving around a given research question (RQ).

- The RQ of §3 is: "What are the *factors* taken into account by practitioners when developing ML-NIDS?" To answer this RQ, we: (i) elucidate the business relationships between the end-users of ML-NIDS and the developers of such ML-NIDS; (ii) outline the challenges that such developers must face when devising their solutions; (iii) present the factors that contribute to the real value of a ML-NIDS; and (iv) validate our factors by directly asking the practitioners' opinion.
- The RQ of §4 is: "What should research on ML in NID do to *allow practitioners to estimate* the real value of the corresponding results?" To answer this RQ, we: (i) formalize our notion of a *pragmatic assessment*; (ii) explain how to conduct a pragmatic assessment through comprehensive guidelines.
- The RQ of §5 is: "Does the state-of-the-art allow us to estimate the real value of ML methods for NID?" To answer this RQ, we: (i) review *all* papers on ML-NIDS presented in top security conferences since 2017; (ii) analyze to what extent they meet the conditions of pragmatic assessments; and (iii) report the practitioners' viewpoint on the state of research.
- The RQ of §6 is: "Can pragmatic assessments be done in research?". We answer this RQ by performing the first pragmatic assessment of ML-NIDS. We do so through a large set of experiments focused on network traffic classification. Our evaluation reports the (statistically validated) performance of thousands of ML models, spanning across diverse datasets, algorithms, pipelines, and labeling budgets. Moreover, we perform our experiments on different platforms, and showcase the importance of *hardware*—which is often overlooked in literature (and also in practice!).

We discuss our results and compare our paper with related work in §7. To ensure reproducibility we release our code [35]: hence, our SoK can also serve as a benchmark for future studies. Due to the sheer size of our experimental campaign, the low-level details and results are provided in our code repository. Finally, we provide details on our survey with practitioners in the Appendix (App.B).

2. Background and Problem Statement

We first outline the general context of NIDS (§2.1). Then, we delve into the specific application of ML in NIDS (§2.2), and explain how research on ML-NIDS is typically carried out (§2.3). Finally, we elucidate the problem tackled by our SoK paper (§2.4).

2.1. Network Intrusion Detection Systems

The security of IT systems spans over three tasks: prevention, detection, and reaction [36]. It is well-known that perfect prevention is unattainable, whereas the reaction phase implicitly assumes that the attack has already taken place. Hence, to minimize (or nullify) the damage resulting from a breach, a major role is played by the detection of cyber threats. In the case of network security, such a role is devoted to "NIDS." Such term encompasses diverse meanings (e.g., [37], [38]). Let us provide the definition of NIDS adopted in our paper:

> **DEF. *1.*** A NIDS is a *system* that protects a *network*, i.e., a set of (IT) systems that interact with each other.

(we refer the reader to the RFC [39] for the exhaustive definitions of "network" and "system")

Since their conception [40], NIDS have undergone significant improvements. Initially, NIDS only analyzed data pertaining to network traffic, such as raw packet captures (PCAP), and the detection was performed by "static" methods, i.e., through human-defined "signatures" encoding patterns of known attacks. Due to the growing complexity of network environments as well as the appearance of adaptive attackers, however, static detection methods became infeasible. To cope with such a dynamic ecosystem, NIDS started to adopt automated detection techniques stemming from the data analytics domain, enabled by the availability of "big data" (potentially originating from diverse sources) and by advancements in computational power [41]. Such data-driven techniques, which include (among others) ML methods, improved NIDS while reducing the burden on human operators.

We provide an illustration of the typical NIDS deployment in Fig. 1, where the NIDS monitors all communications performed by a given organization. The output

of a NIDS is in the form of *alerts* (which can be post-processed by dedicated modules, e.g., [42]), which must be inspected and triaged by security operators. We stress that NIDS can be deployed anywhere in a given network, not just at the border (e.g., [43]).

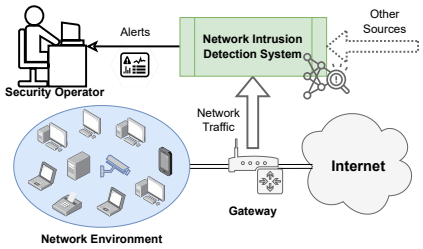

Fig. 1: Typical deployment scenario of a NIDS.

Current state-of-the-art NIDS—and related tools, such as SIEM, (e.g., [44], [45])—may correlate information from various sources (e.g., whois geolocation [46], DNS logs [47], or internal ACL), and can combine multiple detection approaches (e.g., either *misuse-* or *anomaly-*based [48]) on diverse data-types [49]. For example, a growing trend (even among practitioners [50]) is analyzing NetFlows [51], i.e., high-level metadata summarizing the raw-communications between two endpoints [52]. We outline the advantages of NetFlow analyses in App. A.1.

2.2. Machine Learning and NIDS

A NIDS is a system that must orchestrate multiple components. Each component may consider diverse inputs, and its output may also be used by other components. All such components can adopt different analytical techniques, including those belonging to Machine Learning.

The underlying principle of ML is to leverage the functionality of an ML *model*. By applying a given ML *algorithm* to some *training* data, it is possible to develop a ML *model* that can autonomously 'predict' (new) data—e.g., determining whether an activity is legitimate or not. We provide our definition of a Machine Learning-based Network Intrusion Detection System (ML-NIDS):

> **DEF. 2.** An ML-NIDS is a NIDS that includes, among its components, a (trained) Machine Learning model.

To better understand DEF. 2, we provide an exemplary architecture of an ML-NIDS in Fig. 2. The ML-NIDS can receive different types of input data (either in batches or in real-time [53]), which are forwarded to specific *pipelines*; such pipelines are made up of one or more *components*, and analyze the given input(s). In the case of an ML-NIDS, at least one pipeline will include an ML model—typically preceded by a *preprocessing* component tasked to transform the input data to a format accepted by the ML model (e.g., by extracting the relevant 'features'). The output of all such pipelines can then be used as input to other pipelines (and respective components), which may leverage additional ML models (for the same, or a different task). All such results are then aggregated into the output of the NIDS (i.e., alerts).

Among the ML community, it is common to distinguish between *supervised* and *unsupervised* ML algorithms [15]. The difference revolves around the notion of

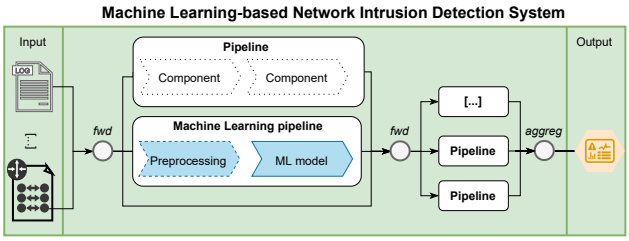

Fig. 2: Architecture of an ML-NIDS.

"label" which denotes the ground truth of a given sample. By providing such labels during the training stage of a supervised ML model, it is possible to 'guide' the learning process and enable, e.g., classification tasks [54]. Obtaining such labels is, however, *expensive* and often error-prone (as shown in [33]). We provide in App. A.2 a more exhaustive description of supervised and unsupervised ML in the NID context—which is followed by an exemplary application of ML to detect malicious traffic (in App. A.3).

As pointed out by many reviews (e.g., [55]–[58]), the applications of ML for NID are highly successful. Accordingly, ML has been shown not only to automate crucial triaging operations [46], but also to exceed the detection capabilities of non-ML NIDS (e.g., [59], [60]).

2.3. ML-NIDS in Research

Let us illustrate the common workflow adopted in research to assess ML-NIDS, schematically depicted in Fig. 3. This workflow—typically borrowed from domains that are unrelated to cybersecurity—begins by acquiring a *dataset*, $\mathbb{D}$. Such $\mathbb{D}$ is divided into a *train* and *evaluation* (or "test") partition—$\mathbb{T}$ and $\mathbb{E}$ respectively—by following a given *split* (e.g., 80:20, i.e., 80% of $\mathbb{D}$ is put in $\mathbb{T}$, and the remaining 20% in $\mathbb{E}$). Then, by using a given learning *algorithm* $\mathcal{A}$ (e.g., DT) on $\mathbb{T}$, a ML model $\mathcal{M}$ is developed: such $\mathcal{M}$ is then evaluated on $\mathbb{E}$, and its quality is measured according to some performance *metric*, μ (e.g., F1-score, Accuracy). The intuition is that if μ is 'good enough' and 'better' than existing proposals, then the respective research has achieved its purpose (e.g. [31], [32]).

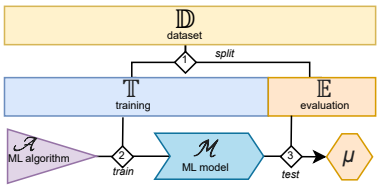

Fig. 3: Typical ML workflow adopted in research.

Despite being correct in principle, such a workflow has two intrinsic limitations from a practitioner's viewpoint.

1) **The lack of an "universal" dataset for NID**. If such a dataset existed, it could be used in any assessment to generalize the performance of an ML-NIDS. However, the immense variability of networks [15] makes creating such a dataset close to impossible (even security companies share such an opinion [61]). Furthermore, this problem prevents [61] a reliable 'transferring' of ML models across different networks (in contrast, transfer is feasible in other domains in which ML has found applications [62]).

2) **The primary focus is on the ML model, $\mathcal{M}$.** Such $\mathcal{M}$, however, is just a single component within the ML-NIDS (see DEF. 2), which is a complex system. For instance, there are many elements that come both *before* and *after* $\mathcal{M}$; moreover, $\mathcal{M}$ can be *physically* deployed on devices mounting diverse *hardware*.

We claim that research papers can provide valuable insights for practitioners. In this paper, we accept the first problem (which cannot be solved today), and focus on rectifying the latter. Specifically, we argue that practitioners are more interested in the (general) ML *method* rather than in the (specific) ML *model*. Hence, practitioners will appreciate if a research: accounts for the most likely scenarios to be faced by the ML-NIDS; and also allows to estimate the *costs* required to sustain the ML-NIDS during its entire operational lifecycle [63], [64].

2.4. Skepticism of ML-NIDS Practitioners

According to a recent survey, over 75% of companies employ ML solutions for network security [65]. Most of such companies, however, *delegate* their cybersecurity to third-party vendors [66]. Indeed, several commercial products for NID actively leverage ML (e.g., [67]–[69]). Yet, all such products adopt ML methods that are decades old and mostly in their unsupervised form (e.g., the one-class SVM of [50] was proposed in 2002 [70]). Simply put, the integration of research endeavours into operational environments is slow in the context of ML-NIDS.

Such slow-pace stems from the skepticism [9] of practitioners towards the 'successes' of research papers. Such skepticism is well-founded: as we will show in our SoK, the current state of research hardly 'complies' with the demands of professional ML-NIDS developers (§5). Indeed, our own survey (§5.3) reveals that research papers—instead of providing answers—leave practitioners with *uncertainty*, which can be summarized as: "It works in *your network*. But will it work equally well in *my network*, and is it *affordable* (now, and in the long-term)?"

Our Goal. We firmly believe that the research community *can* answer such a question. However, providing such an answer (*which not necessarily needs to be always positive* [71]) requires a radical change of the current assessment methodology—which should account for the necessities of real developers. To the best of our knowledge, such necessities have never been formalized in the context of ML-NIDS (related work is discussed in §7.3). Therefore, we first elucidate all the *factors* that practitioners must take into account whenever real deployment of ML in NID is considered. Then, we propose the notion of *pragmatic assessment* which explains what research papers must do to satisfy the needs of practitioners. Finally, we *perform the first* pragmatic assessment of ML in NID.

3. Practical Deployment of ML in NID

Our first contribution addresses the RQ: "What are the *factors* taken into account by practitioners when developing ML-NIDS?" To answer this RQ, we must first elucidate the business perspective of ML-NIDS, and then describe the deployment challenges faced by developers when designing ML solutions for NID.

3.1. Business Perspective of ML for NID

Consider an organization that uses a NIDS (which may or may not already leverage ML) to protect its network, and that wants to enhance such NIDS with a new ML solution for a given detection problem. To this purpose, the organization can develop the ML solution *in-house*, or rely on *commercial-off-the-shelf* (COTS) products [72]. Let us elucidate the implications of these two use-cases.

- **In-house.** The organization must first design the ML solution, which can be done either by replicating existing proposals or by devising an original one. Then, the organization must oversee the ML solution for its entire *lifecycle*, which includes: data collection, preprocessing, and labeling (for both training and testing the ML model); development of the ML model (including repeated testings for parameter calibration); deployment of this ML model in the NIDS infrastructure; as well as any maintenance [73], [74].
- **COTS.** The organization must choose among available products on the market the one that best fits their NIDS. Such a choice depends on the characteristics of a given COTS solution, as advertised by its *vendor*.

In both use cases, the deployment of a ML-NIDS entails two players: the *end user* (i.e., the organization), and the *developers* (i.e., either an external vendor, or the same organization). Such a relationship is represented in Fig. 4: The organization needs a solution according to its security strategy, and the developers provide a product to meet this demand. In any case, it is the *developer* who has to make technical decisions and ensure the operational quality of the final product—during its entire lifecycle.

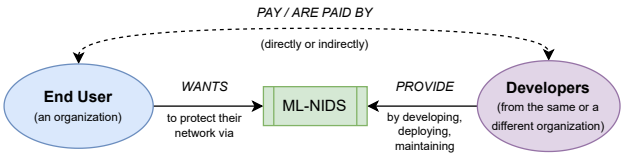

Fig. 4: The business perspective of ML-NIDS.

Remark: Real ML-NIDS require *developers* who are responsible for the lifecycle of the ML model (Fig. 4).

3.2. Deployment Challenges of ML for NID

Three main challenges affect the real deployment of ML-NIDS: (1) each network is unique, (2) each network perpetually evolves over time, and (3) the implicit presence of adversaries. These challenges (which contribute to the "lack of an universal dataset" mentioned in §2.3) are emblematic of ML in NID and exist irrespective of who oversees the lifecycle of the ML-NIDS. Let us explain.

1) The **uniqueness of networks** has been pointed out in various works (old [15] and recent [28]): some activities are legitimate in one network and illegitimate in another network. Hence, deployment of ML-NIDS requires training *and* testing operations performed on data originating from the monitored network [75].
2) The **dynamic nature** of modern networks is another major hurdle for deployment of ML in NID. Every day, new hosts can appear or be removed; new services may be adopted; and new network segments

may be attached—all of which may introduce new types of vulnerabilities. Such phenomena represent the well-known problem of "concept drift" [76].

3) The implicit **presence of adversaries** implies a different, more serious type of concept drift [34]. While the natural network evolution may be controllable to some degree[4], this is hardly the case for attackers who *want* to evade a NIDS [77]. Such adversaries are well motivated and may even devise evasion strategies that specifically target the ML model [72]).

We observe that the last two challenges (intrinsic to most ML applications in cybersecurity [8]) are unpredictable[5], and hence cannot be solved *during the development* of an ML model. Overcoming these challenges *is possible* but requires re-assessments of the ML model *after its deployment*. If the prediction quality of the ML model deteriorates, it must be updated or replaced. Such maintenance is indispensable for ML-NIDS, and accounting for its costs—unique to each network—is crucial for determining the pragmatic value of ML solutions for NID.

Remark: Real ML models for NID must be *developed*, *deployed*, and *maintained* via periodic re-assessments. Such operations must be performed *independently* for each network monitored by an ML-NIDS.

3.3. Factors affecting the real value of ML in NID

The deployment challenges of ML for NID are well-known by practitioners, who must take into account several factors *before* developing any ML model. We now answer our first RQ by connecting all the foundations described insofar and elucidate all such factors.

Overview. The *value* of any security solution can be expressed as the tradeoff between operational effectiveness and expenses. An ML method for NID is *effective* if it yields an ML model $\mathcal{M}$ that exhibits, e.g., a high detection rate while raising few false alarms. The *expenses* reflect all costs incurred during the lifecycle of $\mathcal{M}$. We denote the value of an ML model as $\Psi(\mathcal{M})$. From the research perspective, $\Psi(\mathcal{M})$ depends (at the high level) on the ML algorithm $\mathcal{A}$ and on the dataset $\mathbb{D}$ (§2.3). However, from the practical viewpoint, $\mathbb{D}$ must be collected, and $\mathcal{M}$ must be deployed in the real NIDS. These operations introduce additional dependencies that crucially affect both the effectiveness and the expenses in $\Psi(\mathcal{M})$.

Factors. We propose to formalize the dependencies contributing to $\Psi(\mathcal{M})$ through the following five factors.

- Preprocessing ($\mathcal{P}$). There exist a plethora of mechanisms (each with its own operational costs) meant to transform raw data into the format accepted by an ML model $\mathcal{M}$. These mechanisms affect the information included in $\mathbb{D}$ and utilized by $\mathcal{M}$ to make its predictions—hence influencing the effectiveness of $\mathcal{M}$. Consider, for instance, the generation of NetFlows from PCAP, for which many tools are available—each having its own logic [52]: As shown in [29], exactly the same raw data (in PCAP) yields different NetFlows (even if generated via similar tools), leading to ML models with different performance (we will also show this in our experiments).
- Data availability ($\mathcal{D}$). The quality of a given $\mathbb{D}$ is linked to its size and sample diversity, so that $\mathcal{M}$ can properly 'learn' how to predict future data [78]. However, obtaining such a $\mathbb{D}$ has a cost [79], which is higher when $\mathcal{M}$ requires *labelled* data for training[6]. Ground truth verification is costly and error-prone [61], and it can lead to noisy samples [80]. For instance, [33] found many labelling issues in a well-known dataset for NID (the `CICIDS17` [30]). Finally, although some tools can (synthetically) generate malicious data (e.g., CALDERA [81]), some companies require several months to obtain a representative dataset of 'normal' network activities (e.g., CAIAC [75]).
- System Infrastructure ($\mathcal{S}$). Any $\mathcal{M}$ is just a single element within the NIDS, and hence its effectiveness depends on the NIDS infrastructure (§2.2). The infrastructure determines, e.g., the type of data analyzed by $\mathcal{M}$. For instance, the information included in the NetFlows analyzed by an $\mathcal{M}$ is dictated by the sensors deployed in the NIDS infrastructure. The infrastructure, furthermore, affects (i) the type of decisions expected from $\mathcal{M}$, (e.g., binary or multi-class classification); as well as (ii) the logical arrangement of the individual decision units within the ML pipeline. For instance, a pipeline can include a standalone ML model, an ensemble of ML models, or a cascade of ML models (e.g., [82]–[84]). We provide a schematic of an ML pipeline including a cascade of a binary and multi-class classifier in Fig. 10.
- Hardware ($\mathcal{H}$). The *detection* capabilities of a ML model are hardly affected by the computational resources available. However, hardware influence the *runtime* for both the *training* and the *inference* stage of $\mathcal{M}$. The former is necessary for the periodic re-training[7] of $\mathcal{M}$; the latter is crucial to determine where $\mathcal{M}$ can be physically deployed. Indeed, ML models for NID can be placed anywhere in a network [65], spanning from low-power IoT devices [85] to high-end computing platforms [86].
- Unpredictability ($\mathcal{U}$). It is impossible to know *in advance* how the threat landscape and the network environment will evolve. Moreover, ML methods introduce further uncertainty by using randomized algorithms (e.g., Random Forests); but also because it is not possible to know a-priori how to collect a $\mathbb{T}$ that maximizes the effectiveness of $\mathcal{M}$ (and that does so in the long-term).

We can hence express the value of a ML method for NID as a function f defined with the following equation (Eq.):

$$\Psi(\mathcal{M}) = f(\mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{H}, \mathcal{U}). \quad (1)$$

Because of $\mathcal{U}$, we note that $\Psi(\mathcal{M})$ is not deterministic.

We stress that all the factors above influence each other. For instance, $\mathcal{S}$ also implicitly affects $\mathcal{H}$, but also $\mathcal{P}$. Furthermore, ML solutions for NID should be continuously assessed ($\mathcal{U}$), which requires both human and computational resources. For instance, updating $\mathcal{M}$ with new $\mathbb{T}$

4. E.g., administrators know when organizations adopt new services.

5. E.g., even if administrators are aware of major changes, they do not know if such changes will impact the performance of their ML-NIDS.

6. We observe that while $\mathbb{T}$ is required for supervised methods, a labelled $\mathbb{E}$ is always necessary to validate performance [49].

7. Training-time is also crucial for fine-tuning: an optimal configuration will be found in less time for methods that are faster to train.

may require additional labeling efforts [87] ($\mathcal{D}$); however, such retraining can be computationally expensive ($\mathcal{H}$), and overlooking the training runtime can be detrimental [88].

Practitioner Validation. We conducted a survey asking the opinion of practitioners on our proposed set of factors. Our population entails 12 practitioners with hands-on experience in ML and NID; overall, our participants work (or have worked) in the SOC of renown companies. (We provide all details in `App.B`.) The results of our survey are summarized in Table 1, which reports the percentage of our interviewees that believed whether each of our factors was: "not important" (◯); "important" (!); or "crucial" (!) for real deployments of ML-NIDS.

TABLE 1: Viewpoint of practitioners on our set of factors.

Factor	◯	!	!
$\mathcal{P}$	0%	9%	91%
$\mathcal{D}$	9%	18%	73%
$\mathcal{S}$	9%	27%	64%
$\mathcal{H}$	9%	64%	27%
$\mathcal{U}$	9%	18%	73%

On average, 66% of practitioners consider all our factors to be "crucial" for estimating the real value of a ML-NIDS. Interestingly, 0% believe that preprocessing ($\mathcal{P}$) is "not important," which was ranked as the most crucial factor by all our respondents. The unpredictability ($\mathcal{U}$) and data availability ($\mathcal{D}$) are also deemed to be pivotal by 73% of our population. The least relevant factor is hardware ($\mathcal{H}$), which is considered "important" by 64%. However, as we will show, $\mathcal{H}$ can be the deciding factor to assert which ML solution is truly the best (§6.3).

Takeaway: Our proposed five factors ($\mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{H}, \mathcal{U}$) are considered to be relevant for estimating the real value of ML in NID by most practitioners.

4. Pragmatic Assessment of ML-NIDS

We now address our second RQ: "What should research on ML in NID do to *allow practitioners to estimate* the real value of the proposed solutions?" Indeed, practitioners must account for all the factors in Eq. 1: they will not implement an ML method without knowing how much training data is required. They would also be reluctant to reproduce an ML method if it is not clear whether such a method is truly superior to existing solutions. Finally, an ML method for NID that has not been tested in an adversarial environment may contain security risks [89].

To answer our second RQ, we propose the following notion of *pragmatic assessment* which draws on several past works from both the research (e.g., [8], [90], [91]) and industrial (e.g., [92], [93]) domains.

DEF. 3. A *pragmatic assessment* allows practitioners to assert the value of an ML method for NID iif:

- the reported results are free of any experimental bias, and present high degree of confidence;
- the evaluation is carried out on testbeds resembling the (likely) operational scenarios of the NIDS;
- all requirements for developing the proposed ML method are clearly specified.

Let us explain how these three conditions can be met *in research* and at a high level, starting from the last one.

4.1. Development Requirements

A pragmatic assessment must transparently disclose *all* information pertaining to the requirements for developing (and maintaining) a given solution. In the context of research on ML-NIDS, such information must include:

- The schematic of the *NIDS infrastructure* with respect to the proposed ML method ($\mathcal{S}$ in Eq. 1). Such schematic must pinpoint 'where' the corresponding ML model is meant to be deployed. Such information serves to establish: (i) the function of the ML model; (ii) which components/specifications are required to operate the ML model; and (iii) whether additional components are required to post-process its output.
- The *hardware specifications* of the platforms used to train and test the ML model, which affect its runtime ($\mathcal{H}$ in Eq. 1). Such specifications must include the RAM, the CPU (i.e., model, threads, maximum frequency) and—if necessary—the GPU. It is also important to report the CPU utilization during its runtime (i.e., how many threads were used, and at what frequency), because it plays a crucial role in the energy consumption. In particular, especially for the CPU, the *exact* model must be reported[8]. For instance, stating that "the CPU is an Intel Core i5" (e.g., [95]) is misleading because there are hundreds of such CPUs with significantly different performance: according to PassMark, an i5-470M is 35 times slower than a i5-12600KF [96]. To demonstrate the effects of 'superficial' hardware specifications, we perform an original experiment §4.4.
- The *dataset composition* for both the training $\mathbb{T}$ and evaluation $\mathbb{E}$ partitions ($\mathcal{D}$ in Eq. 1). Such information is crucial for supervised ML methods, as it allows determining the amount of labeled data necessary to develop the respective ML model. Such information, however, is also relevant for unsupervised ML algorithms, because even unlabelled data has a cost [79].
- The *details of the ML method* used to develop the ML model. Such details include the feature set, the exact algorithm (e.g., DT) and its parameters, the task (e.g., binary or multi-class classification), and the design of its pipeline (e.g., stand-alone or ensemble). All such information contributes to $\mathcal{P}$ and $\mathcal{S}$ in Eq. 1.

Finally, it is (obviously) desirable that the implementation code is openly released, and if the adopted dataset is publicly available. As stated by Lindauer et al. [97], scientific reproducibility "facilitates progress": if the entire testbed is publicly accessible, then developers can determine if there are any similarities between the real and experimental environments—potentially enabling a direct transfer of the resulting ML model (if the environments are similar).

Reporting all the above-mentioned details also allows to roughly estimate the expenses for maintaining the ML solution (therefore accounting for part of $\mathcal{U}$ in Eq.1).

4.2. Likely Operational Scenarios

Security systems must face real threats, hence pragmatic assessments must consider scenarios that are likely to occur in reality. To meet this condition, we propose three

8. Note that CPUs can be under/overclocked and therefore exhibit different frequencies than those reported by their manufacturers [94].

complementary use-cases that *can* be taken into account in research on ML for NID. Given the lack of an universal dataset (§2.3), our underlying intuition is to *maximise the utility of a given dataset*. Doing this requires the researcher to use their domain expertise and 'creativity'.

4.2.1. Closed and Open World. It is not wrong to consider "closed world" scenarios, i.e., where the ML model expects each sample to resemble those seen during its training stage. However, ML methods should be assessed *also* in "open world" scenarios [91], due to the unpredictability of the threat landscape ($\mathcal{U}$ in Eq. 1). Indeed, these are the scenarios that ML methods originally intended to address [15]. For unsupervised anomaly detection, open world scenarios are implicit: after learning a given concept of 'normality', no pre-existing knowledge is required to detect anomalous behaviors (unsupervised methods have no notion of 'classes'). In contrast, for classification problems (common in NID) assessing open world scenarios requires additional effort: Testing ML classifiers *only* on a $\mathbb{E}$ having the exact same classes as $\mathbb{T}$ (closed world) prevents estimating any form of adaptability of the ML-NIDS. For a pragmatic assessment, the ML classifier should be evaluated *also* on an $\mathbb{E}$ containing attacks different than those in $\mathbb{T}$ (open world).

This can be done by (a) injecting in $\mathbb{E}$ some malicious classes *not included* in the original $\mathbb{D}$ – e.g., by borrowing malicious samples from other datasets [32]; or by entirely creating novel attack classes via, e.g., [81] (as done in [98]). Alternatively, it is possible to (b) *exclude* some malicious classes in $\mathbb{D}$ from being put in $\mathbb{T}$, and put such classes in $\mathbb{E}$ instead. Both approaches are viable and can be combined in principle. However, as pointed out by Apruzzese et al. [28], "*mixing data from different networks presents some fundamental issues*". For instance, if two networks are considerably different then it is difficult to trust the resulting performance of an ML model. Therefore, mixing data from different networks should be done only after thorough topological analyses.

4.2.2. Static and Temporal Data Dependency. ML methods were originally conceived by assuming the validity of the *iid* principle, i.e., "independent and identically distributed random variables" [99]. However, the iid principle does not always hold in network environments because the data (both benign and malicious) analyzed by a NIDS is likely to present temporal dependencies. As an example, a botnet-infected machine will first contact its CnC, and only afterwards it will execute the malicious commands received by the CnC. For this reason, it is recommended (e.g., [8]) to choose $\mathbb{E}$ so that its samples come 'after' $\mathbb{T}$. Investigating only this 'temporal' case, however, prevents a generic assessment: the results will only resemble the 'sequence' of the samples captured by a given $\mathbb{D}$. Hence, to provide more general results, we propose to consider *both* cases, i.e., by assuming that: (a) samples are all independent of each other; (b) temporal dependencies may be present in the data stream.

Investigating both cases in research[9] requires a dataset $\mathbb{D}$ containing time-related information. Assessing the 'static' case is straightforward: it is sufficient to compose $\mathbb{T}$ and $\mathbb{E}$ by randomly sampling from $\mathbb{D}$. On the other hand, for the 'temporal' case, it is necessary to split $\mathbb{D}$ into $\mathbb{T}$ and $\mathbb{E}$ according to sensible temporal criteria. For instance, the split can be based on the *timestamp* associated to each sample; it is also possible to choose as $\mathbb{E}$ the 'last' portion of $\mathbb{D}$, and use as $\mathbb{T}$ the 'first' part (assuming that $\mathbb{D}$ is chronologically ordered). Nevertheless, the *time-gap* between $\mathbb{T}$ and $\mathbb{E}$ should not be overlooked. For example, the results can differ if only minutes pass between $\mathbb{T}$ and $\mathbb{E}$, compared to when the gap is days or weeks.

4.2.3. Naive and Adaptive Adversaries. Security systems must always assume the presence of adversaries. Such adversaries can be 'naive' and rely only on known offensive strategies (i.e., hoping to bypass an unpatched system). However, the most serious threats come from 'adaptive' attackers who actively attempt to exploit the specific vulnerabilities of their target. In the case of ML methods, such vulnerabilities involve the so-called adversarial examples [100]. After more than a decade of research demonstrating their effectiveness, it is paramount for pragmatic assessments to also consider such a threat.

There are dozens of ways to bypass ML systems via adversarial examples [101] and considering all such ways is clearly infeasible since they are ultimately unpredictable ($\mathcal{U}$ in Eq. 1). As stated by Biggio and Roli, priority should be given to the "more likely threats" [89]. The idea is endorsing *defensive proactivity*: the developer evaluates an ML method in advance against the adaptive "adversarial" attacks that are more likely to occur in reality. To this end, it is crucial to consider adaptive attacks that conform to a *threat models that are both viable and feasible*. We provide the following recommendations (extending those by [72]) to facilitate the design of such threat models.

- *Adversarial Mindset*. Real attackers adopt a cost/benefit rationale [64]: they will not launch attacks requiring huge resource investments—even if they are likely to succeed, there may be other targets (i.e., different from ML models) that yield a better 'profit'.
- *Consider the right "Box"*. Adversarial ML threat models are often expressed with the notion of a "box" that identifies the system targeted by the attacker. In the case of ML methods for NID, the "box" is the entire NIDS—and not just the specific ML model. Hence, when considering a "white-box" attacker, such an attacker would have complete knowledge of the entire NIDS—i.e., a rather extreme circumstance, as such information is well-protected [72]. For this reason, we recommend not to place "white-box" settings at top priority (contrarily to [8]): such worst-case scenarios are feasible in general security, but not very likely against NIDS.[10]
- *Realizable Attacks*. Aside from conforming to the assumed threat model, the perturbation used to create an adversarial example should be physically realizable [102]. This does not mean that it must be created in the "problem-space" [103], as this may

9. We note, however, that investigating both cases may not be 'universally' possible. Sequential ML methods that specifically look for temporal patterns (e.g., [53]) implicitly assume the presence of temporal dependencies; whereas some datasets may simply not provide time-related information to investigate any form of temporal dependencies.

10. We argue that attackers with full knowledge of the whole NIDS would opt for more disruptive strategies than data perturbations.

not be feasible[11] in research when operating on a pre-collected dataset $\mathbb{D}$. Indeed, as observed by [72], even perturbations in the feature space can be realistic if the manipulation preserves the dependencies between features, and considers features on which a real attacker has some influence.[12]

- *Unbounded Perturbations*. Research on adversarial ML usually aims at devising minimal perturbations that are subject to self-imposed constraints (e.g., one pixel attacks [105]). However, as also remarked by Carlini et al. [106] (and, more recently, also by [107]), real attackers are not interested in 'bounded' perturbations, as long as they achieve their goal (e.g., evading a security system).

We make an important remark. Assessing the robustness to adversarial perturbations serves to gauge the vulnerabilities (or strengths) of an ML method *before* its deployment. It is up to the end user of such an ML method to determine whether the envisioned threat deserves a dedicated treatment—which should be economically justified [71].

4.3. Unbiased and Statistically Validated Results

The recent paper by Arp et al. [8] provides sensible recommendations on how to conduct a meaningful evaluation of ML in cybersecurity. For instance, the base-rate fallacy should be considered, the right performance metrics should be measured, and comparisons should be made with the right baselines. All such guidelines are relevant for NIDS and must be followed also for our proposed pragmatic assessments. Such guidelines, however, lack a crucial piece: the performance of an ML method should be *statistically validated*. The motivation is simple: to account for the (intrinsic) randomness of ML ($\mathcal{U}$ in Eq. 1); and to mitigate the (intrinsic) sampling bias in $\mathbb{T}$ and $\mathbb{E}$.

Such statistical validation is achieved by repeating the experiments[13] for a sufficient amount of trials, whose focus is establishing the (unbiased) performance of the ML method—and not of a single ML model. Indeed, only by measuring the performance of a large 'population' of ML models—all trained/tested in similar settings—it is possible to estimate the real value of the corresponding ML method. Moreover, large populations enable *statistical comparisons*, a powerful tool for determining which ML method is truly the best. Carrying out comparisons that are statistically significant (i.e., assuming a target $\alpha < 0.05$), however, requires many trials. For instance, an ML method yielding an ML model with 0.992 accuracy cannot be claimed to be 'better' than another ML method whose ML model exhibits 0.991 accuracy *over a single trial*. Therefore, in cases where two methods yield models with similar performance, a large amount of trials may be required[14]. We thus discourage relying just on cross-validation techniques, as they do not provide a sufficient amount of measurements for pragmatic assessments.[15]

Takeaway: Accounting for all the factors contributing to the real value of ML for NID requires pragmatic assessments, summarized in Fig. 5. Extensive information must be provided, diverse likely scenarios must be considered, and multiple trials must be made to provide statistically significant results.

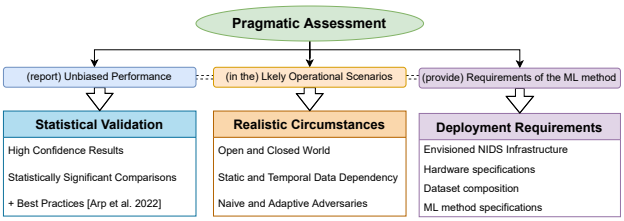

Fig. 5: Characteristics of Pragmatic Assessments of ML in NID.

4.4. Experiment: the importance of CPU specs

We perform a simple experiment to demonstrate the importance of reporting the *complete* CPU specifications.

Objective. We consider the simple task of measuring the runtime for training and testing an ML model on a given dataset. Specifically, we train and test a Decision Tree (DT) binary classifier on the `GTCS` [110] dataset (i.e., $\mathbb{D}$); more details in `App.D.1`. We randomly sample 80% (i.e., $\mathbb{T}$) of $\mathbb{D}$ to train the DT, and test it on the remaining 20% (i.e., $\mathbb{E}$). We repeat such experiments 10 times.

Specifications. We consider two different platforms, whose setup are nearly identical "on the surface": they both mount 8GB of DDR3 RAM (using the same frequencies), both run Windows 10 OS, and the experiments are done on the exact same version of Python and scikit-learn. The only difference is the exact model of the CPU: one is an Intel i5-4670, and the other is an Intel i5-430; both CPUs use their default clock speeds. Both training and testing the DT require only a single CPU core.

Results. On average, *training* the DT on the i5-4670 requires 11.1s, but it takes 34.7s on the i5-430 (a 310% increase). Whereas *testing* requires an average of 0.39s on the i5-4670, and 1.38s on the i5-430 (a 350% increase). Hence, reporting only a portion of the specifications (e.g., "an Intel i5 CPU") introduces a lot of uncertainty on the *actual* performance of the final ML model.

5. State-of-the-Art (in Research)

As a final motivation for this paper, we answer the following RQ: "Does the state-of-the-art allow one to estimate the real value of ML methods for NID?" We hence review recent literature to determine how much existing works 'comply' to our notion of pragmatic assessment.

11. Complete realistic fidelity is almost impossible as it would require to reproduce the attacker's operations in the specific targeted network.

12. A very recent work [104] pointed out that attackers may even be able to directly control the feature representation of a given example.

13. We stress that pragmatic assessments require such statistical validation for all the 'likely' scenarios (§4.2). For instance, the adversarial robustness should be repeated many times (as also recommended in [106]), each applying the same perturbation but to different samples.

14. Some tests require a sample-size of at least 50 [108]. However, the test may also be inconclusive: in this case, no claim can be made.

15. As an example, consider a $\mathbb{D}$ that is partitioned into $\mathbb{T}$ and $\mathbb{E}$ with an 80:20 split. Such a split allows to apply 5-fold cross validation, which produces only 5 results and hardly valid to determine whether an ML model is statistically better than another. In contrast, a more convincing and unbiased approach is to perform a large amount of trials by randomly sampling $\mathbb{T}$ and $\mathbb{E}$ from $\mathbb{D}$ many times (e.g., 50), each time with the same 80:20 split. Such an approach allows to compare two populations of 50 samples (via, e.g., a Welch's t-test [109]), enabling to derive sound conclusions on which ML method is better.

Disclaimer. Similarly to [8], [107], [111], the following analysis is not meant to invalidate previous works: ultimately, none of such works aimed at realistic deployment. Our intention is highlighting that the current evaluation protocol adopted in research papers can (and should) be improved. We provide a case-study describing some 'practical redundancies' of a recent work (by the same authors of this SoK) in App. A.4.

5.1. Methodology (literature review)

Scope and inclusion criteria. The amount of papers that propose to use ML for NID is off-the-charts. To perform a feasible but comprehensive analysis, we investigated all papers published in nine of the most reputable cybersecurity conferences.[16] For each venue, we investigated the proceedings from 2017 to 2021,[17] and selected all papers that fell within our scope. Such selection resulted in 30 papers,[18] considering diverse types of networks (from enterprise [46] to IoT [112]) and cyber threats (from anomalous traffic [113] to APT [114], and even adaptive attacks [115]). Nonetheless, all such papers shared the same underlying assumption: the usage (and evaluation) of ML to detect 'intrusions' in networks.[19]

Analysis. We inspected each selected paper from the perspective of our 'pragmatic assessment' notion. Because each paper had different assumptions, we performed our analysis by asking ourselves six questions—each having a set of standardized answers (⇒). Specifically:

1) "Are the *hardware* specifications clearly reported?"⇒Yes (✓); partially (◐, e.g., no details on CPU model); not provided (✗).
2) "Is the *runtime* clearly specified?"⇒Yes (✓); only training time (𝕋); only inference time (𝔼); no (✗).
3) "Is the vulnerability to *adaptive adversarial attacks* mentioned?"⇒Yes, and it is evaluated (✓); yes, but only stated as a limitation (◐); not mentioned (✗).
4) "Is the *statistical significance* used to provide more convincing results?"⇒Yes (✓); no (✗).
5) "Is the training dataset ever changed to account for diverse *data availability*?"⇒Yes (✓); no (✗).
6) "Is the evaluation done on (at least some) *public data*?"⇒Yes (✓); yes, but it is not available today (◐); no, such data has always been kept private (✗). We also noted how many datasets were used.

The results of such analysis are summarized in Table 2. We also remark that we considered two additional criteria, namely: (i) whether the ML models were tested only in a "closed world" setting; and (ii) whether the paper considered different preprocessing operations. Such criteria are not included in Table 2 because the *response was the same for all papers*, i.e.: all 30 papers evaluated their models (also) against unknown attacks (most of such papers are on anomaly detection, which implicitly assumes an "open world" setting); and none of the 30 papers considered different preprocessing mechanisms.

16. We consider: IEEE SP and EuroSP; ACM CCS, AsiaCCS, ACSAC; NDSS and USENIX Security; as well as DIMVA and RAID.

17. Some of these conferences still have to be held in 2022.

18. We went through the proceedings four times over 5 months.

19. E.g., the ML-NIDS may analyze network data (e.g., NetFlows [46]), or may account for data generated from an entire network (e.g., finding 'anomalies' in the measurements of all sensors in a given network [116]). We do not consider "malware detectors" (analyzing, e.g., android apps [117] or javascript [118] or PE files [119]) as ML-NIDS.

TABLE 2: State-of-the-Art: papers published since 2017 in top cybersecurity conferences that consider applications of ML linked with NID.

Paper	Year	Hardware	Runtime	Adaptive	Stat. Sign.	Avail.	Pub. Data
Bortolameotti [113]	2017	✗	✗	✓	✗	✗	✗ (1)
Ho [120]	2017	✗	✗	◐	✗	✗	✗ (1)
Cho [121]	2017	✗	✗	✓	✗	✗	✗ (1)
Siadati [122]	2017	✗	✗	◐	✗	✗	✗ (1)
Oprea [46]	2018	✗	𝕋	◐	✗	✗	✗ (1)
Pereira [95]	2018	◐	𝕋	◐	✗	✓	◐ (1)
Kheib [123]	2018	✗	✗	◐	✗	✗	✗ (1)
Araujo [124]	2019	✗	𝔼	✗	✗	✓	✗ (1)
Mudgerikar [112]	2019	✗	✓	✗	✗	✗	✗ (1)
Mirsky [60]	2019	◐	✓	◐	✗	✗	✓ (1)
Feng [125]	2019	✗	✗	◐	✗	✗	✓ (2)
Milajerdi [114]	2019	◐	✓	◐	✗	✗	✓ (1)
Liu [126]	2019	◐	✗	◐	✗	✗	✓ (2)
Du [127]	2019	✗	𝕋	◐	✗	✗	✓ (3)
Erba [116]	2020	◐	𝔼	✓	✗	✓	✓ (2)
Bowman [98]	2020	◐	𝔼	✗	✗	✗	✓ (2)
Leichtnam [128]	2020	◐	✗	✗	✗	✗	✓ (1)
Singla [129]	2020	✗	✗	✗	✗	✓	✓ (2)
Han [130]	2020	✓	✓	◐	✗	✗	✓ (2)
Jan [131]	2020	✗	✗	✓	✓	✓	✗ (1)
Ghorbani [132]	2021	✓	𝔼	◐	✗	✗	✗ (1)
Nabeel [133]	2021	✗	✗	◐	✗	✗	✗ (1)
Wang [115]	2021	✗	𝔼	✓	✗	✗	✓ (2)
Piszkozub [134]	2021	✗	✗	◐	✗	✗	◐ (2)
Yuan [135]	2021	✗	✗	◐	✗	✓	✓ (1)
Yang [136]	2021	✗	✗	◐	✓	✗	✓ (1)
Barradas [137]	2021	◐	✓	◐	✗	✗	✓ (1)
Han [138]	2021	✓	✓	✓	✗	✓	✓ (2)
Liang [139]	2021	✗	𝕋	◐	✓	✓	✓ (1)
Fu [140]	2021	◐	✓	✓	✗	✗	✓ (3)

5.2. Major Findings (and our interpretation)

From Table 2, we see that *no one fits all*: despite being published in top conferences, no single paper allows to estimate the deployment value of the considered ML solutions. Nonetheless, we highlight some intriguing trends.

Only a snapshot. Most papers assess the quality of ML methods by training and testing the corresponding ML models on a single 'snapshot'. For instance, such ML models are often evaluated only once, preventing to derive more general conclusions; it is concerning that the term 'statistical significance' is mentioned only in 2 papers (i.e., [131], [136]). Moreover, most papers (almost 70%) do not vary the composition of their training dataset, preventing to estimate the value of the ML method when a company cannot afford to invest many resources in the data collection procedures. We acknowledge that some of these papers propose 'unsupervised' ML techniques; however, even unlabelled data has a cost [79]. In addition, no paper considers different preprocessing mechanisms (§5.1): we appreciate that most papers thoroughly describe the preprocessing operations of their solutions; however, such procedures (including all parameters) are never changed, preventing to determine their impact on the ML pipeline. Finally, most papers use a single dataset.

Neglected Requirements. Only three papers (i.e., [130], [138]) provide a holistic vision of the hardware and runtime requirements used to develop the corresponding ML models. For instance, the proposal in [139] requires 2.5 hours to train, but no hardware information is provided. We find it concerning that even papers that specifically focus on IoT settings do not provide such details. For instance, the authors of E-Spion [112] rightly state that "E-Spion is specifically designed for resource-constrained IoT devices": they do measure the CPU utilization, but without reporting *which* CPU was used. Such an omission can be acceptable in research, but not when real deployments are considered.

Smart Attackers. On a positive note, the majority of papers considers an "open world" setting in which adversaries try to actively bypass the considered ML-NIDS. Some papers even evaluate the impact of adaptive

attacks *in addition* to measuring the performance in their absence—which is commendable. We remark that [115], [116] specifically focus on such a threat, and hence have slightly diverse assumptions: for instance, not reporting the hardware or runtime is less of a problem for [115], [116]. However, the lack of multiple trials ensuring statistically significant results is still an issue.

5.3. Practitioners' Opinion

In our survey with practitioners, we also asked for their opinion on Table 2. Specifically, *after* asking the questions related to our factors (§3.3), we inquired whether the fact that some columns have many "✗" was: "not very problematic" (◯), "problematic" (!), or "very problematic" (!). The results are shown in Table 3. Most practitioners (90%) agree that the lack of statistically significant comparisons is "very problematic." Moreover, 59% believe that lack of data diversity is an issue. Perhaps surprisingly, 75% can overlook the absence of evaluations against adaptive adversarial attacks. Finally, the lack of hardware specifications was also deemed to be not a crucial shortcoming—our evaluation will prove otherwise.

TABLE 3: Practitioners' opinion on the results displayed in Table 2.

Column	◯	!	!
Hardware	25%	75%	0%
Runtime	0%	75%	25%
Adversarial	8%	67%	25%
Stat. Sign.	0%	10%	90%
Avail.	16%	42%	42%
Pub. Data	0%	41%	59%

Nonetheless, at the end of our questionnaire we posed one last question to our interviewees: "In general, do you think that research papers facilitate the practitioners' job in determining the *real value* of the proposed ML methods?" The answers were enlightening: 92% are "uncertain", whereas 8% are "left with more questions than answers after reading a research paper".

Takeaway: Despite abundant work proposing ML methods for NID, the state-of-the-art the art does not allow practitioners to determine the real value of existing ML solutions. We attempt to change the current evaluation protocols with our proposed *pragmatic assessment* notion—which *can be done*, as we will now show.

(Given our findings, we wondered: "did the situation change in 2022?" We investigate this question in App. C.1)

6. Demonstration of a Pragmatic Assessment

To bridge the gap between research and practice, we now focus on our last RQ: "Can pragmatic assessments be done in research?", and make a constructive step towards the integration of state-of-the-art ML methods into real NIDS. Specifically, our goal is threefold: (**i**) Demonstrate that our guidelines *can* be followed in research experiments; (**ii**) showcase an exemplary case-study of ML for NID, malicious NetFlow classification, wherein we *pragmatically assess* existing ML methods; (**iii**) provide *statistically validated results* for future studies, by publicly disclosing the complete details and low-level implementation.

Our evaluation is massive, hence the complete details are reported in the Appendix (and repository [35]). Here, we summarize our testbed (§6.1), present some original results (§6.2), and derive practical considerations (§6.3).

6.1. Experimental Setup

Our evaluation revolves around the well-known problem of malicious NetFlow classification, which can be done via ML.[20] We chose this problem because it allows one to devise diverse ML pipelines. Indeed, NetFlow is generated by preprocessing raw PCAP data; moreover, *detecting* malicious NetFlows can be seen either as a binary or multi-class classification problem (because a sample can belong to diverse malicious classes). Such a problem can be tackled through diverse ML pipelines, e.g., it is possible to create an *ensemble* of 'specialized' binary classifiers (each trained on a subset of the available data—similarly to [60]); but it is also possible to create a cascade of a binary and multi-class classifier: the former determines whether a NetFlow is benign or malicious, and the latter infers the specific class of a malicious NetFlow, e.g., a DDoS or a Botnet (a schematic of such 'cascade' is shown in Fig. 10). Moreover, many (labeled) datasets are publicly available, ensuring scientific reproducibility.

These characteristics enable a broad coverage of use-cases. In particular, we consider *thousands* of different configurations, which vary depending on the following:

- **Source Dataset (5)**: `CTU13`, `NB15`, `UF-NB15`, `CICIDS17`, `GTCS`. Each of these datasets is created via a different NetFlow tool: Argus, nProbe, Zeek, FlowMeter. An overview of these datasets is in Table 4, while more details are in App. D.1.
- **Data Availability for training (4)**: *Abundant* (80% of $\mathbb{D}$), *Moderate* (40%), *Scarce* (20%), *Limited* (only 100 samples per class in $\mathbb{D}$). Refer to App. D.3.
- Size of the **feature set (2)**: *Complete* (i.e., using all features provided by the NetFlow tool) or *Essential* (using only half of such features). Refer to App. D.4.
- **ML Pipeline (6)**: a single *binary detector* (BD); a single *multi-class detector* (MD); a *cascade* of BD and MD (BMD); as well as three ensembles which vary depending on how the output is determined: via a *logical or* (ED-o), through *majority voting* (ED-v), or via a *stacked* classifier (ED-s). Refer to App. D.5.
- **ML Algorithm (4)**: Random Forest (RF), Logistic Regression (LR), Histogram Gradient-boosting (HGB), Decision Tree (DT). Refer to App. D.6.
- **Hardware specifications (6)**: a high-end computing appliance, a workstation, a common desktop, an old laptop, a virtual machine with reduced capabilities, and a Raspberry Pi 4B. Refer to App. D.2.

Each combination can be seen as an unique ML-NIDS, which is assessed against: *known* (by testing on the same attacks seen at training), *unknown* (by testing on attacks not seen during training), and *adversarial* attacks (based on [141], as they are feasible and hence likely to occur [72]). A detailed description of all these distinct operational scenarios' is provided in App. E. For each ML-NIDS, we compute the true and false positive rate (tpr and fpr); accuracy (Acc, but only for multi-classification tasks); and runtime (for both training and testing).

20. Out of the 30 papers in Table 2, 16 use NetFlow-related data: [46], [60], [113], [115], [124], [126], [128], [129], [132], [134]–[140].

TABLE 4: Summary of the datasets of our experimental evaluation.

Dataset Name	Benign Samples	Malicious Samples	Attack Classes	Features	NetFlow Software
CTU13 [142]	16.7M	403K	6	30	Argus [143]
NB15 [144]	2.2M	105K	7	45	Zeek [145]
UF-NB15 [29]	2.3M	78K	7	40	nProbe [146]
CICIDS17 [30]	1.6M	433K	9	76	FlowMeter [33]
GTCS [110]	140K	378K	4	80	FlowMeter [147]

To provide statistically significant results and remove any bias, we repeat all our experiments (both training and testing) multiple times, specifically: 1000 times for the *limited* data availability (as there is a high chance of bias), and 100 times for the three other availability settings. Such repetitions are done by randomly sampling $\mathbb{T}$ from $\mathbb{D}$ according to the data availability setting; whereas $\mathbb{E}$ is always chosen by randomly selecting 20% of the available samples of each class available in a given $\mathbb{D}$. Moreover, we always follow the "dos" proposed by Arp et al. [8]. (Our evaluation is *fair*: for each trial, we train all our models on the same $\mathbb{T}$, and evaluate them on the same $\mathbb{E}$.)

Finally, we also perform an extra set of experiments in which $\mathbb{T}$ and $\mathbb{E}$ are chosen by taking the temporal domain into account, i.e.: $\mathbb{E}$ contains only the 'last' 20% samples of a given dataset, and $\mathbb{T}$ contains the 'first' samples.

Remark: our evaluation is massive, and is due to our goal of providing a benchmark for future studies. A single research paper needs not to perform an evaluation of the same magnitude as the one in this SoK.

6.2. Main Results (Quantitative Analysis)

Let us discuss the results of detectors using HGB, since it is a very recent algorithm for NID. Here, we aggregate the results of *all* datasets, and we focus on the *detection* performance on the *high-end* platform. Fine-grained results are in App.F, which reports the *multi-classification* performance, and the runtime on different hardware.

6.2.1. Baseline Performance. We report in Fig. 6 the boxplots showing the tpr and fpr of our detectors for increasing (left to right) data availability settings. We can see that detectors using ED-v exhibit the worst tpr but the best fpr, which is understandable because they require multiple classifiers to agree on the maliciousness of a sample. In contrast, the other detectors appear to have comparable performance. We find it intriguing that MD detectors appear to be effective even using a very limited amount of labels (see rightmost plot in Fig. 6).

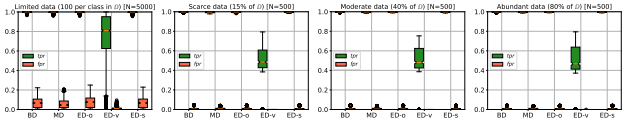

Fig. 6: Baseline Performance.

6.2.2. Detection of unknown attacks. We report in Fig. 7 the performance against unknown attacks—which is computed by excluding one malicious class from a given $\mathbb{T}$, re-training all the involved ML models on such new $\mathbb{T}$, and testing them on the benign portion of $\mathbb{E}$ (for the fpr), and on the 'excluded' malicious class (and then averaging the resulting tpr). From Fig. 7 (which has the same structure as Fig. 6) we can see that the tpr decreases, which is expected because the attacks are unknown. The detectors based on BD appear to be the most robust. It is intriguing that the best results are achieved in the *Limited* data availability setting. Such phenomenon can be explained by the fact that training on few samples allows ML models to generalize better on 'unseen' classes.

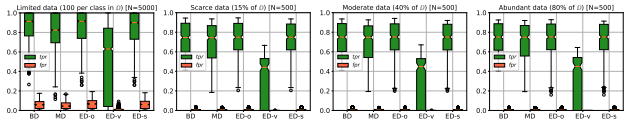

Fig. 7: Detection of unknown attacks.

6.2.3. Adversarial Robustness. We measure the robustness of our detectors against the evasion attacks proposed in [141]. The results are shown in Fig.8, reporting the tpr both before (green bars) and after (red bars) the application of the adversarial perturbations for all the detectors and for increasing amounts of training data (left to right). From these results, we can see that our detectors are *more robust* when they are trained with *less data*: indeed, the red bars in the leftmost graph are always higher than those in the other graphs (a similar phenomenon as the one in Fig. 7). In particular, we observe that BD is the most resilient detector when using limited data, but the weakest (aside from ED-v) when more data is available for training.

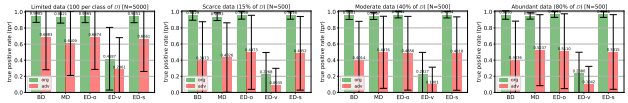

Fig. 8: Robustness to adversarial attacks.

6.2.4. Runtime. We report the operational runtime (as measured on the high-end platform) for all our detectors (ED includes both ED-o and ED-v) in Fig. 9. Each plot (related to a specific data availability setting) reports the time (in seconds) for training (blue bars) and testing (brown bars) the respective detectors. We can see that, on limited data, training is computationally less expensive than testing. Moreover, we also observe that training the ensembles is much more resource intensive than the simple MD and BD (the latter being the 'cheapest' to train). More details are in App.F.3).

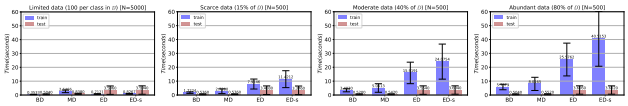

Fig. 9: Runtime (on the *high-end* platform).

6.3. Practical Considerations

By inspecting all our results (App.F), we conclude that the "no free lunch" statement [148] is, once again, correct. For instance, methods based on HGB can have a slightly superior performance than, e.g., RF (cf. the fpr and tpr of BD using the complete feature set with limited training data on GTCS in Table 9a). However, the HGB is worse in "open world" scenarios (Table 10a)—but it has a lower runtime (cf. Tables 23 and 28). In this cases, it is up to practitioners to decide which method to deploy in their NIDS. Our pragmatic assessment, which reports both the effectiveness (i.e., tpr, fpr, or Acc) and expenses (i.e., all

the requirements as well as the operational runtime) enables practitioners to make informed decisions. Let us use our experiments to draw some practical considerations.

The power of statistical comparisons. By performing a massive amount of trials, it is possible to carry out statistical tests that can be used to infer which ML method truly 'outperforms' other competitors.[21] For instance, by looking at the multi-classification tables (App. F.2), we can see that the *Acc* of BMD and MD tend to be close (e.g., on GTCS using HGB with Limited data, BMD has 0.994 while MD 0.982). With a Welch's t-test, we find that the resulting p-value is less than 0.0001 (i.e., below the usual target $\alpha = 0.05$), which statistically proves that BMD is better than MD (accuracy-wise). Despite being powerful, such verifications are underused in NID literature (§5).

Hardware can determine the winner. Let us recall our motivational example (§1). By looking at the detection results in the closed world scenario using the complete feature set on CICIDS17 (Table 15a), we can see that—with the exception of LR methods—almost all detectors (BD, MD, ED-s and ED-o) achieve near-perfect performance in the abundant data availability setting, with tpr close to 1, and fpr close to 0; even a statistical test cannot determine the best method. In these cases, the 'winner' can be determined by looking at the runtime in Table 26. We can see that the fastest method to train is the BD using HGB, which requires 9s. However, HGB uses *all 36 cores* of the high-end platform: in contrast, training the BD using DT requires 25s but by using only one core. From a CPU utilization perspective, the HGB is 13 times slower than DT. Hence, our takeaway is that the 'best' ML method for NID on CICIDS17 uses DT as ML algorithm.[22]

Small or Big Data? Some intriguing results have been obtained by ML models trained with limited amount of labeled data (see the leftmost plots in Figs. 6 to 8). Some of our ML models exhibited similar tpr as those trained with a considerably higher budget, but they had a higher tpr against both unknown and adversarial attacks (but at the expense of a slightly higher fpr against 'known' attacks). This finding is noteworthy, as it may help in demystifying the necessity of having training datasets that count millions of samples. To quote a recent statement by Andrew Ng: "*Collecting more data often helps, but if you try to collect more data for everything, that can be a very expensive activity*" [149]. We hence endorse development of ML methods that require smaller training datasets.

Concrete use-case. Suppose an organization wants to deploy an ML method in their NIDS for identifying malicious NetFlows. The organization can compare their own network environment with those captured by our five considered datasets, and see whether there exist any similarities between our testbed and their real network. Suppose that the organization finds some similarities with the networks contained in NB15 and UF-NB15: at this point, the organization can determine whether the NetFlow tool used by their NIDS is compatible to those used in NB15 and UF-NB15 (potentially by also considering the Essential and Complete feature set considered in our experiments). For example, if the organization has already a NetFlow tool using nProbe, then such an organization can almost directly transfer our ML methods trained on UF-NB15 onto their NIDS. Otherwise, the organization needs to manually deploy nProbe into their NIDS first (which requires some expenses). In the (likely) chance that the organization finds no similarities with their own network and those captured by our chosen datasets, such an organization can choose to develop the solution that best fits their necessities, e.g., by choosing the one that provides the best performance while requiring the least amount of labels.

21. In real scenarios, even a 0.0001% can be significant: a single false negative can compromise a system, whereas the fpr must be close to 0.

22. We reached out to some of the respondents of our survey (after they filled the questionnaire), and told them about such a finding: this made them change their mind on the importance of hardware.

7. Discussion and Related Work

We present some intrinsic difficulties of pragmatic assessments, perform some reflective exercises on our findings, and compare our paper with related literature.

7.1. Challenges of Pragmatic Assessments

A research paper that fulfills each criteria in DEF. 3 would be appreciated by practitioners. However, while some conditions are easy to meet, others are more difficult. Let us discuss some (current and future) challenges, so as to clarify the function of pragmatic assessments.

Statistical Significance. Obtaining results that are devoid of bias requires to perform multiple randomized trials. The experiments carried out in this paper required *weeks* of computations—some of which are performed on expensive hardware. Furthermore, some ML methods are rooted on the existence of temporal patterns among data (e.g., [53]): in these cases, performing many trials for statistically significant comparisons requires to either split the original $\mathbb{D}$ into different subsets or use completely different $\mathbb{D}$. Therefore, we acknowledge that pragmatic assessments are not simple—which explains the situation portrayed in Table 2. They are, however, doable: for instance, Liang et al. [139] performed more than 50 trials for some of their experiments. Nonetheless, in some cases (i.e., if the results are 'notably' different) only few trials are sufficient: the crux is reporting *how many* trials have been performed. Finally, we encourage future works to rely on statistical tests when claiming that a given ML method "outperforms the state-of-the-art."

Shortage of Public Data. A well-known problem in NID is the lack of datasets usable for research purposes [75]. Such a lack makes it impossible for scientific papers to *exactly* replicate the (real) network environment in which the proposed ML method can be deployed. Therefore, a pragmatic assessment is meant to "allow practitioners[23] to estimate the real value of an ML method for NID," and not to "ensure that every ML method for NID is deployed in practice." Indeed, the latter requires researches to evaluate their ML-NIDS in every possible network environment, which is clearly unfeasible. Nonetheless, future endeavours should attempt to evaluate their ML methods on diverse datasets—which is important to practitioners (§5.3). We outline the opinion of practitioners on NID datasets in App. C.2.

Concept Drift and Explainability. A pragmatic assessment should not aim at investigating the robustness

23. We stress that such an "estimate" is outside the scope of a research paper, since it can only be done by the developers of real products.

of an ML method to the concept drift problem.[24] Indeed, robustness to concept drift can be realistically assessed only *after* the deployment of an ML model; in contrast, the goal of a pragmatic assessment is to guide decision making *before* such deployment. Nevertheless, we acknowledge that some ML methods can better deal with concept drift [150], such as lifelong learning (e.g., [117]), or those methods that present a high explainability [34], [151]. In particular, we mention that the participants of our survey commented that providers of security solutions should favor methods that are "explainable to their clients". Unfortunately, it is well-known that the decisions of ML models are difficult to interpret [152]. Hence, we cannot put the "explainability" into our proposed factors, as it would be unfeasible to fulfill by research.

7.2. Reflections and Recommendations

Feasibility and Sweet Spots. We provide some recommendations that can maximize the pragmatic value of research without requiring extensive effort. We focus on those aspects that apply to "any" paper on ML-NIDS.

- *Experimental details.* Providing all details (§4.1) of the testbed (including hardware) is straightforward. The only issue are page limitations: in these cases, researchers can provide a link to supplementary files (but we also endorse editors and organizers to accept longer papers during the peer-review).
- *Performance.* As recommended by Arp et al. [8], at least two 'classification' performance metrics should be computed (we used tpr and fpr), which is trivial to accomplish. Moreover, measuring the runtime (for both training and testing) is also straightfoward and requires just few lines of code (plus, it helps in devising a sound and efficient experimental workflow).
- *Testbed variety.* Typically, a research paper on ML-NIDS requires to evaluate (i) the proposed method, and (ii) a suitable baseline for comparison—both of which should be assessed in the same settings.[25] However, we endorse papers that assume ML-NIDS requiring large $\mathbb{T}$ to also assess cases entailing a 'very small' $\mathbb{T}$ (some real product require months of data collection before they can be deployed [75]). Doing this is feasible since the training time is shorter, and the 'smaller' $\mathbb{T}$ can be generated as a subset of the 'larger' $\mathbb{T}$ (but in both cases, $\mathbb{E}$ should be the same).[26]
- *Repetitions (supervised ML).* As can be seen from our evaluation, when using 'large' $\mathbb{T}$ the performance does not change substantially (see the distribution of fpr and tpr in §6.2 for the Scarce, Moderate and Abundant data); hence, for these cases, we recommend at least 3x3 repetitions (i.e., changing $\mathbb{E}$ and $\mathbb{T}$ three times each). However, when considering small $\mathbb{T}$ (see the results for the limited data in §6.2) the performance can greatly vary; hence, for these cases, we recommend at least 10x10 repetitions. We stress that the training time for the Limited data was significantly inferior than for all the other cases (refer to Fig. 9), hence such a higher amount of repetitions should be feasible to perform.[27]

Simply put, meeting the requirements for our pragmatic assessment is well within the reach of most researchers.

The role of our factors. We discuss the relevance of our factors (Eq. 1) by using our experiments (§6):

- $\mathcal{P}$ can be observed by comparing the results on `NB15` and `UF-NB15` (e.g., Table 12a and Table 14a), because these datasets contain the exact same raw data, but the NetFlow tool (i.e., the preprocessing) is different. E.g., the MD using HGB with scarce data is robust against our adversarial attacks on `NB15` (0.96 tpr), but the same method on `UF-NB15` is very weak (0.47 tpr).
- $\mathcal{D}$ can be observed from any table (e.g., Table 9) as the performance clearly changes as $\mathbb{T}$ increases.
- $\mathcal{S}$ can be observed by comparing the multi-classification results of any table (e.g., Table 18), as the performance of MD and BMD differs due to different pipelines (cf. our remark on statistical significance); but also by comparing the results of different algorithms in any table.
- $\mathcal{H}$ is shown by Table 32, as the runtime changes up to 400% under diverse hardware settings.
- $\mathcal{U}$ is highlighted by the great variance of results achieved across our entire evaluation, which confirms the role played by randomness.

The unpredictability $\mathcal{U}$ is also implicit: we cannot foresee what is going to happen *after* any ML model is deployed.

User-study: Limitations. Our questionnaire (see `App.B`) resembles that of *structured interviews* (used also, e.g., by [66]), thereby allowing to derive *quantitative* results, while protecting our participants against possible NDA violations [153]. Such a design choice was chosen because our goal is to *validate the importance of our proposed factors* (§3.3), and to get the opinion of practitioners on the *current state-of-research* (§5). Although our closed-questions could introduce some form of bias, we remark that (i) each question had a 'negative' answer; and (ii) in some cases, the viewpoint of our population went against our theses. We acknowledge that our questionnaire could have been formulated in an 'open question' format; however, such a design choice could also be affected by bias, since we ultimately had to interpret the (unstructured) answers we received and map them to our proposed factors. We therefore acknowledge that some practitioners may have some priorities that are complementary to our factors. To account for such a limitation, we invited our respondents to give us some feedback *after* they filled their questionnaire, thereby allowing us to derive additional insight (discussed in `App.B.3`).

7.3. Related Work

Let us compare our paper with prior literature. We stress that our focus is on ML for NID, and we do not claim

24. This requires the researcher to know—in advance—whether a given dataset contains instances of such drift, which may not be the case. Simultaneously, concept drift is unpredictable and it is not known a priori whether it will occur or not. Hence, results derived from 'synthetic' testbeds are questionable due to such unpredictability ($\mathcal{U}$ in Eq. 1).

25. Hence, we reiterate that it is not necessary to consider hundreds of combinations (as we did in our demonstration).

26. Even if the performance with the 'small' $\mathbb{T}$ is subpar, it would not subtract to the paper's contribution (as long as it is sensible to assume that the proposed ML-NIDS requires large $\mathbb{T}$).

27. We believe our proposed "repetition sweet-spots" to be feasible to integrate in any ML-NIDS paper; however, a paper can provide a valid scientific contribution even without following our recommendations.

generality over different domains. Nonetheless, we discuss how our pragmatic assessment can be extended to other security applications of ML in App.C.4.

Technical papers. Taken individually, most papers on ML-NIDS have different goals than ours. The authors of [154] aim to 'outperform the state-of-the-art'; those of [34] focus on concept drift, which is unpredictable hence impossible to detect *before* the deployment of a ML solution (as explained in §3.2). Pendlebury et al. [117] aim to eliminate experimental bias, but ultimately consider a different security problem (i.e., malware analysis). An intriguing research area focuses on *privacy* of ML (e.g., [155]), which is complementary to our goal. Finally, a significant number of papers perform evaluations on outdated datasets (e.g., the NSL-KDD [156]), which makes the corresponding results of questionable value for modern and realistic deployments. Others present uncertainties due to overlooking some factors that real developers must take into account (cf. §5). In contrast, our testbed involves recent datasets (including their 'fixed' version [33]), increasing the realistic fidelity of our experiments. Due to (i) the broad combination of use-cases, (ii) the hundreds of trials to remove bias, and (iii) the consideration of many likely deployment scenarios, our evaluation enables a fair and statistically validated benchmark of existing ML methods for NID—benefiting both research and practice.

Reviews and Surveys. Some reviews tackle the entire cybersecurity domain (e.g., [157]) and do not delve into the specificity of ML; or focus on trustworthy ML development, but not from the perspective of NID (e.g., [158]). Some papers focus on 'deployment' challenges of ML, such as: [159] and [90], which are both very generic and do not focus on networked systems; [73], which does not have any form of practitioner validation, nor systematically explains how research can fulfill their needs; and [74], which is on network applications, but not specific of cybersecurity and thus do not consider the presence of malicious entities—which are intrinsic of NID. A recent paper [11] interviewed 21 SOC analysts but neither proposes nor empirically evaluates any solution that can meet practitioners' needs from the researcher perspective; and do not focus on ML (only 10% of their population uses ML!). We also mention [160] and [161], which propose 'certification' of ML *models*—which is not relevant for NID research, whose focus is on the ML *method* (due to the impossibility of reliably transferring ML models across network environments [28]). More related works provide a broad overview (e.g. [75]) or highlight the issues (e.g., [56]) of ML for generic cybersecurity tasks; others may focus on a single aspect of ML for NID, such as the architecture of an ML-NIDS (e.g., [162]), the role of features (e.g., [82]), the impact of unlabelled data [79], or the weakness to adaptive "adversarial" attacks (e.g., [72]). Our paper *extends* all such works by providing original takeaways—some of which are overlooked, or even contrast those by past work. We provide in App.C.3 an in-depth comparison of this SoK with the (closest) related work by Arp et al. [8] (presented at USENIX Security'22).

Summary. To the best of our knowledge, no paper: (i) elucidates the factors contributing to the real value of ML for NID, and (ii) explains how research can account for such factors; and then (iii) demonstrates how to do this in practice through a statistically-validated re-assessment of hundreds of diverse ML-NIDS; and (iv) performs a user study with practitioners to validate its major claims.

8. Conclusion

The integration of ML methods proposed in research into operational NIDS is progressing at a slow pace, due to the (justified) skepticism of developers towards the results reported in scientific literature. Our SoK paper aims to rectify this problem by changing the existing evaluation methodology adopted in this research domain. We do this by proposing the notion of *pragmatic assessments*, whose objective is allowing practitioners to estimate the operational effectiveness and required expenses related to the entire lifecycle of a ML method for NID. After presenting irrefutable evidence that prior research does not allow to estimate the real value of ML for NID, we perform the first pragmatic assessment. Our massive evaluation represents a benchmark for future research, but is also useful for practitioners who can ascertain the real value of existing ML methods.

One may ask: "**Must any future research paper perform a pragmatic assessment to be considered a significant contribution?**" Our answer is a clear "**no**": a paper that does not meet all requirements of a pragmatic assessment can still be useful *for research*. Indeed, we acknowledge that pragmatic assessments are tough to carry out. However, as we showed, they *can be done*. Hence, we endorse future work to improve their evaluations by embracing our guidelines and using our resources.

ETHICAL STATEMENT. Our institutions do not require a formal IRB approval for carrying out the research presented in this paper. During our efforts, we always adhered to the Menlo report [163]. Our experiments do not raise any ethical concern (they are a re-assessment of prior work). Our survey with practitioners was done so as to preserve the anonymity of our respondents—which is why we cannot disclose any further information about our population. All our participants were informed that their responses would have been used for research. Furthermore, all our participants know the identity and the contact details of the authors of this paper, which they can use to explicitly request their responses to be deleted.

ACKNOWLEDGEMENTS. We would like to thank: the Program Committee of EuroS&P'23 and NDSS'23 for the constructive comments that improved this paper immensely; the practitioners who contributed to our user-study; and the Hilti Corporation for funding.

References

[1] A. Esteva et al. Deep learning-enabled medical computer vision. *Nature Digital Medicine*, 2021.

[2] C.-J. Wu et al. Machine learning at Facebook: Understanding inference at the edge. In *IEEE Int. Symp. High-Perf. Comp. Arch.*, 2019.

[3] D. W. Otter et al. A survey of the usages of deep learning for natural language processing. *IEEE T. Neural Netw. Learning Syst.*, 2020.

[4] D. Amodei et al. Deep Speech 2: End-to-end speech recognition in English and Mandarin. In *ICML*, 2016.

[5] G. Litjens et al. A survey on deep learning in medical image analysis. *Elsevier Medical Image Analysis*, 2017.

[6] D. Ucci et al. Survey of machine learning techniques for malware analysis. *Elsevier Comp. Secur.*, 2019.

[7] T. Gangavarapu et al. Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artif. Intell. Review*, 2020.

[8] D. Arp et al. Dos and don'ts of machine learning in computer security. In *USENIX Security Symp.*, 2022.

[9] M. De Shon. Information Security Analysis as Data Fusion. In *IEEE Int. Conf. Inf. Fusion*, 2019.

[10] A SANS 2021 Survey: Security Operations Center (SOC). Technical report, SANS, 2021.

[11] B. A. Alahmadi et al. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In *USENIX Security Symp.*, 2022.

[12] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *ACM Conf. Comput. Commun. Secur.*, 2003.

[13] K. Wang et al. Anagram: A content anomaly detector resistant to mimicry attack. In *RAID*, 2006.

[14] K. Rieck and P. Laskov. Language models for detection of unknown attacks in network traffic. *J. Comp. Virology*, 2007.

[15] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symp. Secur. Privacy*, 2010.

[16] J. Deng et al. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comp. Vis. Pattern Recog.*, 2009.

[17] Y. You et al. ImageNet training in minutes. In *Int. Conf. Parallel Proces.*, 2018.

[18] K. He et al. Deep residual learning for image recognition. In *IEEE Conf. Comp. Vis. Pattern Recogn.*, 2016.

[19] S. I. Mirzadeh et al. Improved knowledge distillation via teacher assistant. In *AAAI Conf. Artif. Intell.*, 2020.

[20] D. Arp et al. Drebin: Effective and explainable detection of android malware in your pocket. In *Netw. Distrib. Syst. Secur. Symp.*, 2014.

[21] A. Demontis et al. Yes, machine learning can be more secure! A case study on android malware detection. *IEEE Trans. Depend. Sec. Comput.*, 2017.

[22] D. Li et al. Can we leverage predictive uncertainty to detect dataset shift and adversarial examples in android malware detection? In *Ann. Comp. Secur. Appl. Conf.*, 2021.

[23] R. Gálvez et al. Less is more: A privacy-respecting android malware classifier using federated learning. *Proceedings on Privacy Enhancing Technologies*, 4:96–116, 2021.

[24] P. Irolla and A. Dey. The duplication issue within the DREBIN dataset. *J. Comp. Virology Hacking Tech.*, 2018.

[25] G. Suarez-Tangil and G. Stringhini. Eight years of rider measurement in the android malware ecosystem. *IEEE Trans. Depend. Sec. Comput.*, 2020.

[26] N. Daoudi et al. A Deep Dive inside DREBIN: An Explorative Analysis beyond Android Malware Detection Scores. *ACM Trans. Privacy Secur.*, 2021.

[27] P. Mishra et al. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Comm. Surv. Tut.*, 2018.

[28] G. Apruzzese et al. The Cross-evaluation of Machine Learning-based Network Intrusion Detection Systems. *IEEE T. Netw. Serv. Manag.*, 2022.

[29] M. Sarhan et al. Netflow datasets for machine learning-based network intrusion detection systems. In *EAI Int. Conf. Big Data Tech.*, 2021.

[30] I. Sharafaldin et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Int. Conf. Inf. Syst. Secur. Privacy*, 2018.

[31] R. Vinayakumar et al. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 2019.

[32] C. Pontes et al. A new method for flow-based network intrusion detection using the inverse potts model. *IEEE T. Netw. Serv. Manag.*, 2021.

[33] G. Engelen et al. Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *IEEE Symp. Secur. Privacy Workshop*, 2021.

[34] G. Andresini et al. INSOMNIA: Towards Concept-drift Robustness in Network Intrusion Detection. In *ACM Workshop Artif. Intell. Secur.*, 2021.

[35] Source-code of this paper (GitHub). https://github.com/hihey54/pragmaticAssessment, 2023.

[36] H. Yang et al. Security in mobile ad hoc networks: challenges and solutions. *IEEE Wireless Comm.*, 2004.

[37] H.-J. Liao et al. Intrusion detection system: A comprehensive review. *J. Netw. Comp. Appl.*, 2013.

[38] N. Tsikoudis et al. LEoNIDS: A low-latency and energy-efficient network-level intrusion detection system. *IEEE Trans. Emerg. Topics Comp.*, 2014.

[39] R. Shirey. Internet security glossary, version 2. Technical report, 2007.

[40] B. Mukherjee et al. Network intrusion detection. *IEEE Network*, 1994.

[41] D. Chou and M. Jiang. A survey on data-driven network intrusion detection. *ACM Comp. Surv.*, 2021.

[42] A. Nadeem et al. Enabling visual analytics via alert-driven attack graphs. In *ACM Conf. Comp. Commun. Secur.*, 2021.

[43] N. Chaabouni et al. Network Intrusion Detection for IoT security based on learning techniques. *IEEE Comm. Surv. Tut.*, 2019.

[44] P. Radoglou-Grammatikis et al. Spear SIEM: A security information and event management system for the smart grid. *Elsevier Comp. Netw.*, 2021.

[45] B. D. Bryant and H. Saiedian. Improving SIEM alert metadata aggregation with a novel kill-chain based classification model. *Elsevier Comp. Secur.*, 2020.

[46] A. Oprea et al. Made: Security analytics for enterprise threat detection. In *Ann. Comp. Secur. Appl. Conf.*, 2018.

[47] C. Feng et al. A user-centric machine learning framework for cyber security operations center. In *IEEE Int. Conf. Intell. Secur. Inf.*, 2017.

[48] A. Khraisat et al. Survey of intrusion detection systems: techniques, datasets and challenges. *Springer Cybersecurity*, 2019.

[49] A. Yehezkel et al. Network Anomaly Detection Using Transfer Learning Based on Auto-Encoders Loss Normalization. In *Proc. ACM Workshop Artif. Intell. Secur.*, 2021.

[50] D. Ucci et al. Near-real-time Anomaly Detection in Encrypted Traffic using Machine Learning Techniques. In *IEEE Symp. Series Comp. Intell.*, 2021.

[51] Cisco IOS NetFlow. https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/, Accessed in April 2021.

[52] G. Vormayr et al. Why are my flows different? a tutorial on flow exporters. *IEEE Comm. Surv. Tut.*, 2020.

[53] A. Corsini et al. On the evaluation of sequential machine learning for network intrusion detection. In *Int. Conf. Availability, Reliability, Secur.*, 2021.

[54] R. J. Joyce et al. A Framework for Cluster and Classifier Evaluation in the Absence of Reference Labels. In *ACM Workshop Artif. Intell. Secur.*, 2021.

[55] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surveys Tuts.*, 2016.

[56] G. Apruzzese et al. On the effectiveness of machine and deep learning for cybersecurity. In *IEEE Int. Conf. Cyber Conflicts*, 2018.

[57] H. Liu and B. Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 2019.

[58] M. Ring et al. A survey of network-based intrusion detection data sets. *Elsevier Comp. Secur.*, 2019.

[59] G. Apruzzese et al. Identifying malicious hosts involved in periodic communications. In *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, pp. 1–8, Oct. 2017.

[60] Y. Mirsky et al. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Netw. Distrib. Syst. Secur. Symp.*, 2018.

[61] T. Van Ede et al. Deepcase: Semi-supervised contextual analysis of security events. In *IEEE Symp. Secur. Privacy*, 2022.

[62] A. Oliver et al. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *NeurIPS*, 2018.

[63] C. Vishik et al. Key concepts in cyber security: Towards a common policy and technology context for cyber security norms. *NATO CCD COE Publications*, 2016.

[64] K. S. Wilson and M. A. Kiy. Some fundamental cybersecurity concepts. *IEEE Access*, 2014.

[65] N. Kshetri. Economics of artificial intelligence in cybersecurity. *IEEE IT Professional*, 2021.

[66] S. Fischer-Hübner et al. Stakeholder perspectives and requirements on cybersecurity in Europe. *J. Inf. Secur. Appl.*, 2021.

[67] T. Nguyen et al. A security monitoring plane for named data networking deployment. *IEEE Comm. Magazine*, 2018.

[68] Darktrace. Machine Learning in the Age of Cyber AI. Technical report, 2020.

[69] Lastline. Using AI to detect and contain Cyberthreats. Technical report, 2019.

[70] D. M. J. Tax. One-class classification: Concept learning in the absence of counter-examples. *TU Delft – PhD Dissertation*, 2002.

[71] T. Moore. The economics of cybersecurity: Principles and policy options. *Elsevier Int. J. Critical Infrastructure Protection*, 2010.

[72] G. Apruzzese et al. Modeling realistic adversarial attacks against network intrusion detection systems. *ACM Digital Threats: Research and Practice*, 2021.

[73] A. Paleyes et al. Challenges in deploying machine learning: a survey of case studies. *ACM Comp. Surv.*, 2022.

[74] F. Pacheco et al. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Comm. Surv. Tut.*, 2018.

[75] G. Apruzzese et al. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, 2022.

[76] R. Jordaney et al. Transcend: Detecting concept drift in malware classification models. In *USENIX Security Symp.*, 2017.

[77] I. Corona et al. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Elsevier Inf. Sci.*, 2013.

[78] A. Vogelsang and M. Borg. Requirements engineering for machine learning: Perspectives from data scientists. In *Int. Req. Eng. Conf. Workshops*, 2019.

[79] G. Apruzzese et al. SoK: The Impact of Unlabelled Data in Cyberthreat Detection. In *IEEE Eur. Symp. Secur. Privacy*, 2022.

[80] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE T. Neural Netw. Learn. Syst.*, 2013.

[81] MITRE CALDERA. https://caldera.mitre.org/, Feb. 2023.

[82] S. Das et al. Network Intrusion Detection and Comparative Analysis using Ensemble Machine Learning and Feature Selection. *IEEE T. Netw. Serv. Manag.*, 2021.

[83] B. Biggio et al. One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time. In *Int. Workshop Multiple Classifier Syst.*, 2015.

[84] Y. Yu et al. An efficient cascaded method for network intrusion detection based on extreme learning machines. *J. Supercomput.*, 2018.

[85] K. A. da Costa et al. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Elsevier Comp. Netw.*, 2019.

[86] A. Kim et al. AI-IDS: Application of deep learning to real-time Web intrusion detection. *IEEE Access*, 2020.

[87] B. Miller et al. Reviewer integration and performance measurement for malware detection. In *Int. Conf. DIMVA*, 2016.

[88] D. Liu et al. FP-ELM: An online sequential learning algorithm for dealing with Concept Drift. *Elsevier Neurocomputing*, 2016.

[89] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Elsevier Pattern Recogn.*, 2018.

[90] E. Jenn et al. Identifying challenges to the certification of machine learning for safety critical systems. In *Eur. Cong. Embedded Real Time Syst.*, 2020.

[91] V. Rimmer et al. Open-world network intrusion detection. In *Security and Artificial Intelligence*, pp. 254–283. Springer, 2022.

[92] P. M. Winter et al. Trusted Artificial Intelligence: Towards Certification of Machine Learning Applications. *arXiv:2103.16910*, 2021.

[93] On Artificial Intelligence—A European approach to excellence and trust. Technical report, European Commission, 2020.

[94] M. Jalili et al. Cost-efficient overclocking in immersion-cooled datacenters. In *ACM/IEEE Ann. Int. Symp. Comp. Arch.*, 2021.

[95] M. Pereira et al. Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic. In *RAID*, 2018.

[96] PassMark – CPU Benchmarks. https://www.cpubenchmark.net/cpu_list.php, 2022.

[97] M. Lindauer and F. Hutter. Best practices for scientific research on neural architecture search. *J. Machin. Learn. Res.*, 2020.

[98] B. Bowman et al. Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *RAID*, 2020.

[99] M. Dundar et al. Learning classifiers when the training data is not IID. In *IJCAI*, volume 2007, pp. 756–61, 2007.

[100] N. Šrndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *IEEE Symp. Secur. Privacy*, 2014.

[101] G. Apruzzese et al. Addressing adversarial attacks against security systems based on machine learning. In *IEEE Int. Conf. Cyber Conflicts*, 2019.

[102] L. Tong et al. Improving robustness of ml classifiers against realizable evasion attacks using conserved features. In *USENIX Security Symp.*, 2019.

[103] F. Pierazzi et al. Intriguing properties of adversarial ml attacks in the problem space. In *IEEE Symp. Secur. Privacy*, 2020.

[104] G. Apruzzese et al. SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning. In *Ann. Comp. Secur. Appl. Conf.*, 2022.

[105] J. Su et al. One pixel attack for fooling deep neural networks. *IEEE T. Evol. Comput.*, 2019.

[106] N. Carlini et al. On evaluating adversarial robustness. *arXiv:1902.06705*, 2019.

[107] G. Apruzzese et al. Position:"real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. In *IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE, 2023.

[108] M. Happ et al. Optimal sample size planning for the wilcoxon-mann-whitney test. *Statistics in Medicine*, 2019.

[109] D. W. Zimmerman and B. D. Zumbo. Rank transformations and the power of the Student t test and Welch's t test for non-normal populations with unequal variances. *Canad. J. Exp. Psych.*, 1993.

[110] A. Mahfouz et al. Ensemble classifiers for network intrusion detection using a novel network attack dataset. *Future Internet*, 2020.

[111] T. Liao et al. Are we learning yet? a meta review of evaluation failures across machine learning. In *NeurIPS*, 2021.

[112] A. Mudgerikar et al. E-spion: A system-level intrusion detection system for iot devices. In *ACM Asia Conf. Comp. Commun. Secur.*, 2019.

[113] R. Bortolameotti et al. Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In *Ann. Comp. Secur. Appl. Conf.*, 2017.

[114] S. M. Milajerdi et al. Holmes: real-time apt detection through correlation of suspicious information flows. In *IEEE Symp. Secur. Privacy*, 2019.

[115] J. Wang et al. Crafting Adversarial Example to Bypass Flow-&ML-based Botnet Detector via RL. In *RAID*, 2021.

[116] A. Erba et al. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In *Ann. Comp. Secur. Appl. Conf.*, 2020.

[117] F. Pendlebury et al. TESSERACT: Eliminating experimental bias in malware classification across space and time. In *USENIX Security Symp.*, 2019.

[118] A. Fass et al. Hidenoseek: Camouflaging malicious javascript in benign ASTs. In *ACM Conf. Comp. Commun. Secur.*, 2019.

[119] H. Aghakhani et al. When malware is packin'heat; limits of machine learning classifiers based on static analysis features. In *Netw. Distrib. Syst. Symp.*, 2020.

[120] G. Ho et al. Detecting credential spearphishing in enterprise settings. In *USENIX Security Symp.*, 2017.

[121] K.-T. Cho and K. G. Shin. Viden: Attacker identification on in-vehicle networks. In *ACM Conf. Comp. Commun. Secur.*, 2017.

[122] H. Siadati and N. Memon. Detecting structurally anomalous logins within enterprise networks. In *ACM Conf. Comp. Commun. Secur.*, 2017.

[123] M. Kneib and C. Huth. Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks. In *ACM Conf. Comp. Commun. Secur.*, 2018.

[124] F. Araujo et al. Improving intrusion detectors by crook-sourcing. In *Ann. Comp. Secur. Appl. Conf.*, 2019.

[125] C. Feng et al. A systematic framework to generate invariants for anomaly detection in industrial control systems. In *Netw. Distrib. Syst. Secur. Symp.*, 2019.

[126] F. Liu et al. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *ACM Conf. Comp. Commun. Secur.*, 2019.

[127] M. Du et al. Lifelong anomaly detection through unlearning. In *ACM Conf. Comp. Commun. Secur.*, 2019.

[128] L. Leichtnam et al. Sec2graph: Network attack detection based on novelty detection on graph structured data. In *DIMVA*, 2020.

[129] A. Singla et al. Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. In *Proc. ACM Asia Conference on Computer and Communications Security*, 2020.

[130] X. Han et al. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *Netw. Distrib. Syst. Secur. Symp.*, 2020.

[131] S. T. Jan et al. Throwing darts in the dark? detecting bots with limited data using neural data augmentation. In *IEEE Symposium on Security and Privacy*, 2020.

[132] M. Ghorbani et al. Distappgaurd: Distributed application behaviour profiling in cloud-based environment. In *Ann. Comp. Secur. Appl. Conf.*, 2021.

[133] M. Nabeel et al. CADUE: Content-Agnostic Detection of Unwanted Emails for Enterprise Security. In *RAID*, 2021.

[134] M. Piskozub et al. Malphase: Fine-grained malware detection using network flow data. In *ACM Asia Conf. Comp. Commun. Secur.*, 2021.

[135] L.-P. Yuan et al. Recompose event sequences vs. predict next events: A novel anomaly detection approach for discrete event logs. In *ACM Asia Conf. Comp. Commun. Secur.*, 2021.

[136] L. Yang et al. CADE: Detecting and explaining concept drift samples for security applications. In *USENIX Security Symp.*, 2021.

[137] D. Barradas et al. Flowlens: Enabling efficient flow classification for ml-based network security applications. In *Netw. Distrib. Syst. Secur. Symp.*, 2021.

[138] D. Han et al. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. In *ACM Conf. Comp. Commun. Secur.*, 2021.

[139] J. Liang et al. FARE: Enabling fine-grained attack categorization under low-quality labeled data. In *Netw. Distrib. Syst. Secur. Symp.*, 2021.

[140] C. Fu et al. Realtime robust malicious traffic detection via frequency domain analysis. In *ACM Conf. Comp. Commun. Secur.*, 2021.

[141] G. Apruzzese and M. Colajanni. Evading botnet detectors based on flows and random forest with adversarial samples. In *IEEE Int. Symp. Netw. Comput. Appl.*, 2018.

[142] S. Garcia et al. An empirical comparison of botnet detection methods. *Elsevier Comput. Secur.*, 45:100–123, 2014.

[143] Argus NetFlow. https://qosient.com/argus/argusnetflow.shtml, Feb. 2022.

[144] N. Moustafa and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems. In *Military Commun. Inf. Syst. Conf.*, 2015.

[145] ZeekIDS. https://bricata.com/blog/bro-ids-renames-zeek-ids/, Accessed in Feb. 2022.

[146] nProbe. https://www.ntop.org/guides/nprobe/, Feb. 2023.

[147] A. H. Lashkari et al. Characterization of TOR Traffic using Time based Features. In *Int. Conf. Inf. Syst. Secur. Privacy*, 2017.

[148] D. H. Wolpert et al. No free lunch theorems for search. Technical report, 1995.

[149] Andrew Ng: Unbiggen AI. Technical report, IEEE Spectrum, 2022.

[150] J. Lu et al. Learning under concept drift: A review. *IEEE Trans. Knowledge Data Eng.*, 2018.

[151] C. Meske et al. Explainable artificial intelligence: objectives, stakeholders, and future research opportunities. *Inf. Syst. Manag.*, 2022.

[152] U. Bhatt et al. Explainable machine learning in deployment. In *ACM Conf. Fairness, Accountability, Transparency*, 2020.

[153] R. Opdenakker et al. Advantages and disadvantages of four interview techniques in qualitative research. In *Forum: Qualitative Social Research*, 2006.

[154] A. Binbusayyis and T. Vaiyapuri. Identifying and benchmarking key features for cyber intrusion detection: an ensemble approach. *IEEE Access*, 2019.

[155] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *USENIX Security Symp.*, 2019.

[156] C. Iwendi et al. The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems. *Sensors*, 2020.

[157] R. Leszczyna. Review of cybersecurity assessment methods: Applicability perspective. *Elsevier Comp. Secur.*, 2021.

[158] P. Xiong et al. Towards a robust and trustworthy machine learning system development: An engineering perspective. *Elsevier J. Inf. Secur. Appl.*, 2022.

[159] L. Baier et al. Challenges in the deployment and operation of machine learning in practice. In *Eur. Conf. Inf. Syst.*, 2019.

[160] E. Damiani and C. A. Ardagna. Certified machine-learning models. In *Int. Conf. Current Trends Theory Practice Inf.*, 2020.

[161] H. Jiang et al. To trust or not to trust a classifier. *NeurIPS*, 2018.

[162] G. Giacinto et al. Network intrusion detection by combining one-class classifiers. In *Int. Conf. Image Anal. Proc.*, 2005.

[163] M. Bailey et al. The menlo report. *IEEE Secur. Privacy*, 2012.

[164] X. Deng and J. Mirkovic. Commoner privacy and a study on network traces. In *ACM Ann. Comp. Secur. Appl. Conf.*, 2017.

[165] J. Kim et al. P2dpi: practical and privacy-preserving deep packet inspection. In *ACM Asia Conf. Comp. Commun. Secur.*, 2021.

[166] G. Apruzzese et al. Detection and threat prioritization of pivoting attacks in large networks. *IEEE Trans. Emerg. Topics Comput.*, 2017.

[167] E. Chuah et al. Challenges in Identifying Network Attacks Using Netflow Data. In *IEEE Int. Symp. Netw. Comp. Appl.*, 2021.

[168] R. Hofstede et al. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Commun. Surv. Tut.*, 2014.

[169] K. Wolsing et al. IPAL: breaking up silos of protocol-dependent and domain-specific industrial intrusion detection systems. In *RAID*, 2022.

[170] P. Dodia et al. Exposing the Rat in the Tunnel: Using Traffic Analysis for Tor-based Malware Detection. In *ACM Conf. Comp. Commun. Secur.*, 2022.

[171] Zeek conn.log. https://docs.zeek.org/en/master/logs/conn.html, Feb. 2023.

[172] J. Ma et al. Supervised anomaly detection in uncertain pseudoperiodic data streams. *ACM T. on Int. Tech.*, 2016.

[173] A. Nappa et al. Cyberprobe: Towards internet-scale active detection of malicious servers. In *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014.

[174] I. Syarif et al. Unsupervised clustering approach for network anomaly detection. In *Int. Conf. Netw. Digit. Tech.*, 2012.

[175] G. Apruzzese et al. Deep reinforcement adversarial learning against botnet evasion attacks. *IEEE T. Netw. Serv. Manag.*, 2020.

[176] A. Demontis et al. Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In *USENIX Security Symp.*, 2019.

[177] A. Erba and N. O. Tippenhauer. Assessing Model-free Anomaly Detection in Industrial Control Systems Against Generic Concealment Attacks. In *Ann. Comp. Secur. Appl. Conf.*, 2022.

[178] A. S. Jacobs et al. AI/ML for Network Security: The Emperor has no Clothes. In *ACM Conf. Comp. Commun. Secur.*, 2022.

[179] L. D'hooge et al. Establishing the contaminating effect of metadata feature inclusion in machine-learned network intrusion detection models. In *DIMVA*, 2022.

[180] Y. Feng et al. CJ-Sniffer: Measurement and Content-Agnostic Detection of Cryptojacking Traffic. In *RAID*, 2022.

[181] Z. Fu et al. Encrypted Malware Traffic Detection via Graph-based Network Analysis. In *RAID*, 2022.

[182] I. J. King and H. H. Huang. Euler: Detecting network lateral movement via scalable temporal link prediction. In *Netw. Distrib. Syst. Secur. Symp.*, 2022.

[183] M. Landen et al. DRAGON: Deep Reinforcement Learning for Autonomous Grid Operation and Attack Detection. In *Ann. Comp. Secur. Appl. Conf.*, 2022.

[184] R. A. Sharma et al. Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment. In *USENIX Security Symp.*, 2022.

[185] E. Tekiner et al. A lightweight IoT cryptojacking detection mechanism in heterogeneous smart home networks. In *Netw. Distrib. Syst. Secur. Symp.*, 2022.

[186] X. Wang. ENIDrift: A Fast and Adaptive Ensemble System for Network Intrusion Detection under Real-world Drift. In *Ann. Comp. Secur. Appl. Conf.*, 2022.

[187] X. Wang et al. MADDC: Multi-Scale Anomaly Detection, Diagnosis and Correction for Discrete Event Logs. In *Ann. Comp. Secur. Appl. Conf.*, 2022.

[188] A. Kenyon et al. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Comp. Secur.*, 2020.

[189] M. Zipperle et al. Provenance-based intrusion detection systems: A survey. *ACM Comp. Surv.*, 2022.

[190] M. Conti et al. A survey on industrial control system testbeds and datasets for security research. *IEEE Comm. Surv. Tut.*, 2021.

[191] IBM. What is data labeling? https://www.ibm.com/topics/data-labeling, 2023.

[192] "Real Attackers Don't Compute Gradients", a fireside chat with the co-authors on adversarial ML. https://www.robustintelligence.com/resource-center/adversarial-ml-fireside-chat, Feb. 2023.

[193] L. Liu et al. Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In *IEEE Conf. Commun. Netw. Secur.*, 2022.

[194] Sophos intercept x endpoint features. https://www.sophos.com/en-us/products/endpoint-antivirus, 2022.

[195] G. Pellegrino et al. Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics. In *USENIX Workshop Offensive Tech.*, 2015.

[196] A. Kumar et al. Improving detection of false data injection attacks using machine learning with feature selection and oversampling. *Energies*, 2021.

[197] R. Bapat et al. Identifying malicious botnet traffic using logistic regression. In *IEEE Syst. Inf. Eng. Design Symp.*, 2018.

[198] G. Ke et al. Lightgbm: A highly efficient gradient boosting decision tree. *NeurIPS*, 2017.

Appendix A. Additional Background and Use-cases

We provide some supplementary descriptions and examples to facilitate the understanding of our paper.

A.1. NetFlow-based analyses (and tools)

Problem. Analyzing the *all* raw-data generated by modern networks is problematic [52], due to the sheer size of full packet captures (PCAP). Indeed, performing deep packet inspection (DPI) is computationally demanding (in terms of processing and storaging), besides also raising [164] privacy concerns[28].

Solution. To make automated analyses of network data more feasible, a convenient alternative is to analyze high-level summaries of the communications between two endpoints, commonly referred to as *NetFlows*. A NetFlow can be roughly expressed as the following tuple:

$$\text{NetFlow} = (srcIP,\ dstIP,\ srcPort,\ dstPort,\ proto,\ startTime\ endTime,\ ...), \quad (2)$$

where the last three dots can include any element that relates to the other fields (e.g., the amount of bytes transferred in the 'flow'). Compared to traditional PCAP, NetFlows present several advantages. For instance, the PCAP version of the `CICIDS17` dataset is of 50GB, whereas its NetFlow version requires just 1GB [33]. Such low requirements makes NetFlow viable for real time analyses,

28. Encryption may solve the issue, but makes DPI challenging [165].

and the implicit lack of privacy issues is appreciated in commercial products as well as for research—including (e.g., [29], [31], [32]), but not limited to (e.g., [52], [166], [167]), ML-specific proposals.

Variants. Initially introduced by CISCO in 1996 [168], the concept of NetFlow has evolved substantially over the years. For instance, besides being available by default on most CISCO routers, it is possible to generate NetFlows via open-source software tools, such as Argus [143], nProbe [146], or CIC-FlowMeter [147] (including its fixed version by Engelen et al. [33]). We even mention that Zeek [145] (formerly BroIDS), among the leading tools for network monitoring[29], has implemented its own variant of NetFlows, named "connection logs" [171]. In our evaluation, we will consider all these variants.

Disclaimer: NetFlow represent a cost-effective solution to perform data-driven analyses for NID. However, despite being used in both research and practice, NetFlows are not a panacea [98], [167], and we invite future work to explore also other data-types (or create new ones!).

A.2. Supervised vs Unsupervised ML

It is common to distinguish between *supervised* and *unsupervised* ML algorithms [15]. Supervised algorithms require the training data to be provided with *labels* that denote the ground truth of each sample, and are hence suited to 'specific' tasks (e.g., distinguishing benign from malicious samples). In contrast, unsupervised algorithms do not have such requirement, making them applicable only to more 'generic' tasks (e.g., grouping similar data): indeed, without ground truth labels it is not possible to 'supervise' what the ML model is actually learning [54].

At a high-level, both supervised and unsupervised ML algorithms are applicable to either misuse or anomaly detection approaches. For instance, labelled data can serve as a guide to produce the signatures (for misuse detection, e.g., [32]) or to establish the notion of normality (for anomaly detection, e.g., [172]); at the same time, the signatures can be determined by extracting some rules after clustering (e.g. [173]), while the normality can be determined from the clusters with most data points (e.g., [174]).

A.3. An use-case of (supervised) ML in NID

Consider a NIDS that includes a ML model analyzing NetFlows. Such ML-NIDS will receive the raw network traffic from the gateway (Fig. 1). Such data is in PCAP format, and not usable by the ML model: hence, the PCAP is preprocessed into NetFlows (e.g., by using Argus [143]), and the resulting NetFlows are sent as input to the ML model. The output of such ML model can be further utilized, e.g., by an additional ML model. For instance, it is possible to create a cascade of a binary and multiclass classifier (depicted in Fig. 10): the first ML model determines whether a NetFlow is benign or malicious, and the second ML model analyzes only the malicious outputs—according to the M 'known' classes.

The initial PCAP can also be analyzed via traditional signature-based approaches (but in separate pipelines).

29. Zeek also provides additional logging tools (e.g., [98]), and is frequently mentioned in research papers (e.g., [61], [128], [169], [170]).

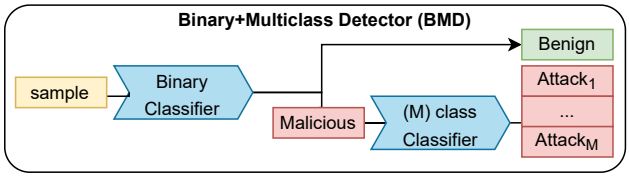

Fig. 10: An ML pipeline representing a detector by cascading a 2- and M-class classifier. The Binary classifier first analyzes a sample, predicting whether such sample is benign or malicious. If the sample is malicious, then it is forwarded to a M-class classifier that determines the specific malicious class (out of M possible classes).

A.4. A 'practically redundant' ML-NIDS

We present a case-study of a 'redundant' ML-NIDS adopted in a recent paper in a high-quality journal, [175]. Our objective is showcasing the immaturity of related research from the perspective of operational deployment.[30]

In [175], an ML-NIDS is first developed, and then assessed in adversarial scenarios. The evaluation is based on the `CTU13` dataset, which contains malicious samples belonging to 5 different botnet families. The adversarial attacks are carried out by applying small perturbations to the malicious samples of each family: the ML-NIDS is then tested on such adversarial samples. From a 'research' perspective, such methodology is *correct*, because the goal in [175] was the assessment of adversarial attacks. However, from a 'practical' perspective, the ML-NIDS considered in [175] is *redundant* due to a questionable architectural design (schematically depicted in Fig. 11). Indeed, such ML-NIDS consists in an ensemble of ML models, with the logic that each model is dedicated to a specific family; however, each model of the ensemble is tested *only* on the samples of its specific family. Hence, the ML models of [175] can only be viable if the NIDS knows—in advance!—which 'attacks' should be forwarded to the ML model(s), therefore defeating the entire purpose of using ML to detect an attack.

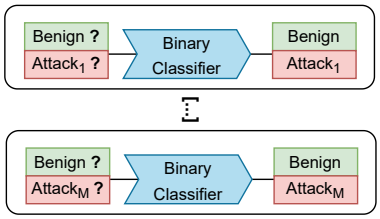

Fig. 11: Exemplary design of a 'redundant' ensemble of ML models for NID (used in [175]). Each classifier is trained on a specific attack (out of M); however, each classifier receives only samples that are either benign, or belong to the specific attack that the classifier can recognize.

Furthermore, the ML-NIDS analyzes samples that are either benign, or correspond to one among 5 botnet families, i.e., a "closed world" setting. For instance, how would the ML models in [175] behave on samples that are generated via different malicious activities (e.g., a brute-force attack)? Finally, all the experiments in [175] are performed by attacking only a single ML model (based on RF). Hence, the effectiveness of the resulting attacks is questionable: what if the attacked ML model was slightly

30. To avoid 'pointing-the-finger', we observe that some of the authors of [175] are shared with those of this SoK; nonetheless, we note that the methodology adopted in [175] derives from others peer-reviewed papers.

different from the one considered in the evaluation? We acknowledge that some adversarial attacks can be "transferred" [176], but such operations do not guarantee the same degree of effectiveness. By performing many trials, all such uncertainties could be removed.

In the pragmatic assessment carried out in this SoK, we do not make any of such 'redundancies', and hence our results have a higher practical value.

Appendix B. Survey with Practitioners

A crucial contribution of our paper is the survey we carried out with real practitioners. Our aim was to substantiate two of our major claims: (i) whether our factors were truly relevant for practitioners; and (ii) whether practitioners truly see research proposals of ML for NID with skepticism—and, if yes, what are some possible reasons.

Let us explain how we performed our survey, which is rooted in *fairness* and *transparency* to minimize bias.

B.1. Selection of Participants

Eligibility Criteria. Our goal was collecting the opinion of 'practitioners' in the context of Machine Learning-based Network Intrusion Detection. For our survey, such "practitioners" entail people that have (had) first-hand experience with such technologies in the industrial sector. In-line with what we described in §3.1, we hence focused on people who either work, or have worked for, companies that either: (a) provide cybersecurity to third-parties, e.g., by monitoring the networks of their clients via ML-NIDS; or (b) manage their own cybersecurity, e.g., they have a section entirely devoted to developing ML-NIDS that protect the network of the entire organization. Furthermore, since we were also interested in collecting meaningful opinions on the current state-of-research, our participants had to have some connection with the research domain (most co-authored peer-reviewed publications).

Population. Overall, we reached out and found agreements with a total of 12 'practitioners'. To prevent bias, the companies for which our practitioners work (or have worked) are all different. To provide comprehensive and diverse opinions, we did not set ourselves any boundary to either the *location* of the company (some are from the USA, some are based in the EU), or in its *size* (some have dozens of employees, some are world-leaders in cybersecurity). Although our population may appear small, we stress that the corresponding companies have clients distributed everywhere in the World. To ensure fairness, all our interviewees were unaware of the *specific* research we were carrying out; and none of the authors of this SoK had ever asked the opinion of the respondents of our survey *beforehand*.[31] Finally, also for fairness, we reached out to our population by sending a generic email, stating that "we want to collect the opinion of practitioners on ML-NIDS about the current state of research and practical deployment of such technologies."

31. In other words, we did not 'cherry pick' people that we knew would confirm our claims (some responses go against some claims!)

B.2. Survey Design

We carry out our survey through an *online questionnaire* having 13 questions with fixed answers.[32]

B.2.1. Organization. The 13 questions were distributed into four 'pages' (P), each with a specific purpose:

- P1) Introducing the questionnaire to the participant, and determining their suitability for our questions.
- P2) Collecting the opinion on our proposed *factors*.
- P3) Collecting the opinion on the situation of Table 2.
- P4) Collecting the opinion on the state of research.

Aside from P1, all the questions in the other pages had three possible answers, which can be summarized as: "yes"; "yes, but"; and "no".

To prevent 'snooping bias' [8], the questionnaire was designed so that participants could not see the questions of a given page until they answered the previous ones. We gave the possibility of participants of not answering questions, because some participants may not have had the expertise to answer all of them. Once they submitted their answers, their response was recorded and no changes could be made. There was no time limit for any question.

We distributed the questionnaire to our participants (after reaching an agreement) via email, which included the link to our questionnaire. We asked each participant to provide us some form of confirmation that they submitted their answers—this was necessary to avoid cases in which a participant filled the questionnaire more than once.

B.2.2. Questions. Only one question was asked in P1 and P4: in P1, we asked whether the company of the participant had a connection with ML and NID; such a question acted as a form of verification (a 'negative' answer would terminate the survey); in P4, we asked the simple question reported (verbatim) at the end of §5.3.

In contrast, P2 and P3 had 5 and 6 questions, respectively. In P2, we: considered each of our proposed five factors (§3.3); provided a brief explanation of such factor; and then asked "how important" such factor was for the respondent. In P3, we first displayed an anonimysed (author names were hidden) version of Table 2, and briefly explained what each column represented. Then, for each of the six columns in table, we asked "how problematic" it was that such a column had a certain amount of ✗.

B.3. Analysis and Feedback

After filling the questionnaire, some of our respondents gave us some **feedback**, which we now summarize.

- *"It depends!"* Many respondents commented that they felt the urge to answer all questions with "it depends". We were expecting this, which is why we did not include such a possibility in our questions: all participants would have chosen that option.
- *"I did not expect that!"* Some respondents stated that hardware is often not a concern in operational environments, because computational resources are abundant. We responded to them by showing some of our results, and they changed their mind: apparently,

32. We created a copy of our questionnaire for reviewing purposes, accessible at this link: https://forms.gle/TxfwmAqG7zi5WCsZ9

they did not expect that some ML methods may exhibit similar detection performance, while requiring substantially different time to train or test.
- *"It must be explainable!"* Some respondents commented that their clients always ask for "reasons why something (bad) happened," thereby inducing security providers to favor 'explainable' ML methods. We were aware of the importance of this 'factor' (as also evidenced in [11]) but we could not include it in our list because it would be unfeasible for *any* research to provide an exhaustive answer for practical purposes—at least today [152] (even [11] argues that explanations are client-specific!).

Let us provide some **additional information**.

- *Timeline.* The first response was registered at the end of Jun. 2022, and the last at the start of Oct. 2022.
- *Missing answers.* One of our respondents did not answer any of the questions in P2, whereas two respondents skipped the "Stat. Sign." question in P3.
- *Length.* Filling the questionnaire required ∼20 mins.

Our repository [35] also includes some code-snippets providing a breakdown of the answers received.

Appendix C. Complementary Analyses

We provide in this Appendix some additional considerations that further enrich the contributions of this SoK.

C.1. State of Research (in 2022)

In §5 we presented the state-of-research from 2017 until 2021. This was because we carried out our survey with practitioners in Summer'22 (i.e., 2022 was still ongoing). However, at the time of writing, all venues considered in our analysis have been held also in 2022: we find instructive to analyze also this year to see if there are any 'improvements' w.r.t. the situation portrayed in §5.

Methodology. We perform the exact same analysis described in §5, but by considering the proceedings of 2022. We repeated our analysis twice, between Feb. and March 2023. We identified 16 papers, reported in Table 5. Altogether, these papers have various goals related to ML-NIDS (e.g., evaluating novel attacks [177], or proposing explainability methods for ML [178]). Similarly to §5, all these papers consider a single preprocessing mechanism (the only exceptions are: [79], [179], which consider data generate via different NetFlow tools); and consider open-world settings (aside from [79], wherein the evaluation represents a closed-world setting).

Improvements. By looking at Table 5, we observe an improvement w.r.t. Table 2 Notably, we appreciate the utilization of more public datasets (which we believe stems from the increased release of open NID datasets[33]) and the consideration of diverse data availability scenario. The hardware also appears to be reported more often w.r.t. Table 2. However, we believe that the most significant improvement (which is not captured in these tables) is an *increased release of source-code*. Indeed, out of the 30 papers in Table 2, only 10 publicly disclosed their source code (i.e., [60], [113], [116], [124], [130], [136]–[140]), which is a mere 33%. Such a low percentage dramatically increased to 75% in 2022: 12 out of 16 papers in Table 5 published their code (i.e., [61], [79], [170], [177]–[180], [182], [184]–[187]). Such a positive trend is encouraging for both research and practice, since it facilitates reproducibility and can also allow practitioners to directly assess research proposals in production environments.

33. Papers using NetFlows: [8], [79], [170], [178]–[182], [186], [187].

TABLE 5: State-of-the-Art (2022): papers published in top cybersecurity conferences that consider applications of ML linked with NID.

Paper	Year	Hardware	Runtime	Adaptive	Stat. Sign.	Avail.	Pub. Data
Apruzzese [79]	2022	✓	𝕋	✗	✓	✓	✓ (3)
Arp [8]	2022	✗	✗	◐	✗	✗	✓ (1)
D'hooge [179]	2022	✗	✗	✗	✗	✓	✓ (8)
Dodia [170]	2022	✗	✗	✗	✓	✗	✓ (1)
Erba [177]	2022	✗	✗	✓	✗	✗	✓ (1)
Feng [180]	2022	✓	✓	◐	✗	✓	✓ (1)
Fu [181]	2022	✓	𝔼	◐	✗	✗	✓ (2)
Jacobs [178]	2022	✗	✗	✗	✗	✗	✓ (6)
King [182]	2022	✓	✓	✗	✗	✓	✓ (3)
Landen [183]	2022	✗	𝕋	✓	✗	✓	✗ (1)
Sharma [184]	2022	◐	✗	◐	✗	✗	✗ (1)
Tekiner [185]	2022	✓	𝔼	✓	✓	✓	✓ (3)
Van Ede [61]	2022	✓	✓	✓	✗	✓	✓ (1)
Wang [186]	2022	✓	✓	✓	✗	✓	✓ (1)
Wang [187]	2022	✗	✗	✗	✗	✗	✓ (3)
Wolsing [169]	2022	✗	✗	✗	✗	✗	✓ (3)

C.2. NID datasets: practitioners' opinion

The real-world utility of public NID datasets has been scrutinized by many works—the most relevant being the paper by Kenyon et al. [188]. In what follows, we extend the main takeaways of [188] by providing some original observations based on our interactions with practitioners.

Context. The performance of any ML method depends on its training data (§2.3). Due to the increasing popularity of ML, the research community on ML-NIDS can now benefit from dozens of publicly available datasets. We refer the reader to some surveys of recent datasets for diverse domains related to ML-NIDS: [29], [58], [179], [189], [190]. Despite the usefulness of such datasets in research, from the operational perspective the sheer concept of a dataset has intrinsic limitations. Let us explain.

Problem. We (informally) asked practitioners about the practical relevance of publicly available datasets for ML-NIDS. The general consensus is that all datasets they are aware of are inappropriate to derive sound conclusions on the applicability of a given ML method. The reasons are diverse, but can be summarized as:

- *Unrealistic assumptions.* Many datasets have samples generated via 'simulations' (e.g., `NB15` [144]), and the labelling may be done either too rigorously or too loosely (e.g., [33]). For instance, assuming that the ground truth is known for *each* sample is overly optimistic (practitioners use coarse labelling strategies [61], [191], [192]).
- *Fixed point in time and space.* In order to serve as a "benchmark" for research purposes, a ML-NIDS dataset must be immutable. As a result, even if the data comes from real networks and corresponds to true attacks (e.g., `CTU13` [142]), its practical value quickly deteriorates as the state-of-the-art advances (e.g., new network services may replace previous standards, and the threat landscape evolves). For instance, showing that a ML-NIDS can detect botnet samples that were 'problematic' 10 years before is not very relevant today (from a practical perspective).

Practitioners also remarked that these problems do not undermine the scientific contribution of research papers.

Mitigation. We asked practitioners if they had any recommendations to mitigate the problems affecting public datasets for ML-NIDS. Accordingly, existing datasets could be *enhanced by generating 'new'* datasets that capture recent trends—e.g., by using CALDERA [81]. For instance, the IDS17 [30] was updated with a more comprehensive version.[34] Doing this, however, may be tough for researchers: creating a new dataset makes the result of prior works not comparable, thereby requiring the researchers to assess previous methods on the new dataset (which is necessary for a meaningful comparison [8]). Unfortunately, performing such re-assessments is not simple due to the lack of source code,[35] preventing a simple (and bias-free) implementation of prior baselines [107], [169]. For this reason, practitioners endorse researchers to *be as open as possible with their implementation*: alongside being helpful for future research, a 'plug-and-play' artifact enables practitioners to assess the proposed ML-NIDS on their own environments—provided, of course, that the corresponding paper allows practitioners to estimate whether it would work in the first place.

C.3. Comparison with a closely related work

We compare our SoK with the work by Arp et al. [8].

Different goals. Arp et al. [8] aim to provide recommendations that improve the soundness of ML assessments *for future research*; in contrast, our recommendations aim to *reduce the practitioners' skepticism on the practical value* of research papers. For this reason, some of our recommendations focus on aspects that are orthogonal to research, and contrast those of [8]. For instance, [8] claim that "an evaluation of adversarial aspects is [...] a mandatory component in security research", and suggest "focusing on white-box attacks where possible"; in contrast, we argue that attackers with full-knowledge of the ML-NIDS is an extreme assumption in real environments (as also mentioned in [107]), and our experiments focus on adaptive attackers with partial knowledge (which are more likely in reality).

Different focus. Arp et al. [8] focus on generic applications of ML for security, and some recommendations have *poor relevance in the specific NID context* (which is our focus). For example, [8] emphasize the problem of "temporal snooping", which may be relevant when, e.g., analyzing malware samples, but not-so-much in when the analyses focuses on network activities over short timespans (as we explained in §4.2.2; even our results show that there is barely any difference, performance-wise).

Overlapping and Actionable recs. While some recommendations by [8] can be applied for our cases (e.g., the base-rate fallacy §4.3), some of our recommendations are *not elaborated in* [8]. For instance, although [8] recommend to "move away from a laboratory setting [e.g., for runtime] and approximate a real-world setting [e.g., for open-world]", there is no mentioning of how this could be done in NID: in contrast, we propose, e.g., 'leaving-out' some malicious samples (§4.2.1), and we perform an original experiment to showcase the importance of hardware on runtime (§4.4).

Literature Analysis and Validation. The main theses of [8] rely on an analysis of 30 papers over 10 years (from 2011 to 2021): in contrast, our SoK considers a higher number (46 in total) of more recent works (from 2017 until 2022). Finally, the contributions by [8] are exclusively based on prior literature and "laboratory findings", whereas our SoK has an additional validation phase supported by a user study with real practitioners.

C.4. Pragmatic Assessments for other IDS

The focus of this SoK is on Machine Learning applications for Network Intrusion Detection Systems. Let us explain how our pragmatic assessment can be applied to other types of Intrusion Detection Systems (IDS) [48].

Context. What sets a (ML-based) NIDS apart from other IDS is the presence of a 'network' element in its analysis (refer to §2). By removing such an element, the "Uniqueness of networks" deployment challenge (§3.2) disappears. From a practical perspective, this leads to *a narrowing-down of the problem*: if the IDS does not have to account for the underlying network complexity, then it is easier to define the boundaries of what represents an 'intrusion' or not. For example, detecting malware at the host level can be done a-priori, since "malware is malicious everywhere, everytime" (§1). Consequently, we argue that the results of similar researches are more directly applicable to reality. As a matter of fact, many commercial security products integrate state-of-the-art ML methods: e.g., *deep learning* is used by Sophos to *detect malware* [194]; and also by other companies to *detect phishing webpages* [107]. Therefore, we believe that there is a reduced necessity for pragmatic assessments in IDS that do not envision the underlying network complexity.

Extension. Research papers on other IDS can, however, still embrace our proposed pragmatic assessment notion: *all our recommendations can be broadly applied to ML-IDS*. Nonetheless, in these cases, we argue that *the role of hardware is even more important*. Indeed, while an organization (may) have the possibility of deploying the ML elements of a NIDS on diverse machines, in the case of host-IDS there is less room for doing this, since the analysis must be performed on the specific host[36]. For example, consider our original experiment in §4.4: the inference time can be substantially different (3x in our case) even for CPUs mounting "an intel i5". Hence, papers on host-IDS (including, e.g., commodity antiviruses) should put high emphasis on the size of $\mathbb{T}$ and $\mathbb{E}$, and on the runtime for both training and testing—while clearly specifying the hardware specifications. We also endorse future researchers to consider different hardware configurations: this can be done, e.g., by downclocking the CPU, or running the experiments on a virtual machine and regulating the allocated computational resources (as we did in our experiments).

34. Unfortunately, even this version was found to be flawed [193].

35. Among the 46 papers we analysed in this SoK, only 22 released their source-code at the time of acceptance (i.e., 48%, see App.C.1).

36. Of course, an organization can choose to deploy the ML element of a host-IDS on a powerful remote machine, but doing this for all machines of an organization may be impractical. Alternatively, if an organization used a centralized server that simultaneously analyzes all the low-level operations of the hosts, then this would resemble a NIDS.

Appendix D.
Experiments: Configuration Settings

We now provide an exhaustive description of our massive experimental campaign. Our experiments focus on supervised ML models for NID that analyze NetFlows. As explained in §6.1, such settings allow to witness the effects of all the 'factors' described in our paper—while guaranteeing reproducibility. Indeed, using NetFlows showcases the role of data preprocessing, supervised settings highlight the importance of labelling, the generic 'intrusion detection' epitomizes the distinction between open and closed worlds, and several public datasets are available.

We begin by presenting the considered datasets. Then, we thoroughly explain all the diverse configuration settings to meet all the conditions of a pragmatic assessment.

Remark: Due to page restrictions, all content beyond App.D.2 is omitted. Such content includes: the additional low-level details (from App.D.3 to App.D.6); the description of the considered operational scenarios (App.E); and the tables containing the complete benchmark results (App.F). However, we provide all these appendices in a dedicated document,[a] which is meant to complement our SoK paper (these materials were provided during the peer-review process of EuroS&P'23).

a. Available in our repository: https://github.com/hihey54/pragmaticAssessment/blob/main/supplementary.pdf

D.1. Public Datasets

To provide meaningful results, for our evaluation we consider five datasets that include recent traffic and attack patterns, and which span across large and small network segments. We focus on datasets that are publicly available and validated by the state-of-the-art. In particular, we consider the following five datasets: CTU13, NB15, UF-NB15, CICIDS17, GTCS. Let us explain our choice.

- CTU13 [142] is one of the largest publicly available datasets for NID. The data in CTU13 is generated in a *large network environment* ($\sim$ 300 hosts), and contains attacks generated by diverse *botnet* families.
- NB15 [144] is well-known [129], [135] and contains *many attacks*, from DoS [195] to shellcode injections.
- UF-NB15 [29] is generated from the *exact same* traffic of NB15, but the NetFlows derive from a different tool (i.e., UF-NB15 has different $\mathcal{P}$ than NB15).
- CICIDS17 [30] is among the most popular datasets (e.g., [32], [128]) for NID. Its original version was found to present labelling flaws [33], so we perform our experiments on the *fixed version* of CICIDS17.
- GTCS [110] is a very recent dataset. It includes similar attacks as those in CICIDS17, but the network is *smaller* (i.e., it has less than a dozen hosts).

An in-depth view of such datasets is provided by Table 6, showing the exact amount of samples per class. For CICIDS17 we merge some underrepresented families into a single class (i.e., *other*); whereas for NB15, UF-NB15 we exclude some families because they had significantly mismatching numbers (in terms of available samples) which—we believe—could be due to labelling issues.

TABLE 6: Distribution of samples for each Dataset.

Dataset	Class	Attack Family	Samples
CTU13	0	*Benign*	16 748 326
	1	*neris*	205 928
	2	*rbot*	143 918
	3	*nsis*	2 168
	4	*virut*	40 904
	5	*donbot*	4 630
	6	*murlo*	6 127
GTCS	0	*Benign*	139 186
	1	*ddos*	131 211
	2	*bot*	93 021
	3	*brute*	83 857
	4	*inf*	70 202
NB15	0	*Benign*	2 218 764
	1	*expl*	44 525
	2	*recon*	13 987
	3	*dos*	16 353
	4	*shell*	1 511
	5	*fuzz*	24 246
	6	*bdoor*	2 329
	7	*ana*	2 677
UF-NB15	0	*Benign*	2 295 222
	1	*expl*	31 551
	2	*recon*	12 779
	3	*dos*	5 794
	4	*shell*	1 427
	5	*fuzz*	22 310
	6	*bdoor*	2 169
	7	*ana*	2 299
CICIDS17	0	*Benign*	1 666 837
	1	*ddos*	95 123
	2	*geye*	7 567
	3	*hulk*	158 469
	4	*http*	1 742
	5	*loris*	4 001
	6	*ftp*	3 973
	7	*pscan*	159 151
	8	*ssh*	2 980
	9	*other*	971

In our experiments, we treat each dataset $\mathbb{D}$ as a separate environment, and we do not perform any mixing due to the intrinsic risks of such operations [28].

D.2. Hardware specifications

We carry out our evaluation on three different platforms each with different computational resources.

- *High-end* (default). A dedicated server for ML experiments, running an Intel Xeon W-2195@2.3GHz (36 cores), 256GB RAM. The OS is Ubuntu 20.04.
- *Desktop*: Intel Core i5-4670@3.2GHz (4 cores) and 8GB of RAM. The OS is Windows 10.
- *Laptop*: Intel Core i5-430M@2.5GHz (4 cores) and 8GB of RAM. The OS is Windows 10.
- *Workstation*: Intel Core-i7 10750HQ@2.6GHz (12 cores) with 32GB RAM. The OS is Windows 10.
- *Low-end*. A 'downclocked' variant of the workstation, running on a Virtual Machine (using Ubuntu 20.04) that is set up to use only 4 cores (using at most 40% of the frequency) and 8GB of RAM.
- *IoT*. A Raspberry Pi 4B with 2GB of RAM (4 cores).

We do not use GPU acceleration to ensure fairness.

We perform the majority of our experiments on the *high-end* platform. The reason (as explained in §3.3) is that hardware only affects[37] the runtime of an ML model. Hence, we use the other platforms to compare the *training* and *inference* runtime of each ML model. We do this only on the GTCS dataset, as runtime scales almost linearly with the size of the analyzed data.

37. We verified this manually: all our ML models we develop across all our platforms achieve ultimately comparable detection performance—despite being trained/tested on different platforms