Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey

Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar

***Abstract*—Network-based Intrusion Detection System (NIDS) forms the frontline defence against network attacks that compromise the security of the data, systems, and networks. In recent years, Deep Neural Networks (DNNs) have been increasingly used in NIDS to detect malicious traffic due to their high detection accuracy. However, DNNs are vulnerable to adversarial attacks that modify an input example with imperceivable perturbation, which causes a misclassification by the DNN. In security-sensitive domains, such as NIDS, adversarial attacks pose a severe threat to network security. However, existing studies in adversarial learning against NIDS directly implement adversarial attacks designed for Computer Vision (CV) tasks, ignoring the fundamental differences in the detection pipeline and feature spaces between CV and NIDS. It remains a major research challenge to launch and detect adversarial attacks against NIDS. This article surveys the recent literature on NIDS, adversarial attacks, and network defences since 2015 to examine the differences in adversarial learning against deep neural networks in CV and NIDS. It provides the reader with a thorough understanding of DL-based NIDS, adversarial attacks and defences, and research trends in this field. We first present a taxonomy of DL-based NIDS and discuss the impact of taxonomy on adversarial learning. Next, we review existing white-box and black-box adversarial attacks on DNNs and their applicability in the NIDS domain. Finally, we review existing defence mechanisms against adversarial examples and their characteristics.**

***Index Terms*—NIDS, adversarial attacks, deep learning, cybersecurity.**

I. Introduction

COMPUTER networks have progressed drastically over the past few decades. Researchers have developed novel network topologies and protocols for better communication across devices under various scenarios, ranging from complex enterprise networks to relatively small networks such as the Internet of Things (IoT) [1]. With the increasing number of network devices, securing the network from potential network attacks is crucial as networks are under various security threats. Network attacks can compromise the confidentiality, integrity, and availability of various assets, including the data, systems, and networks [2]. Typical examples of network attacks include: password cracking (*e.g.,* John the Ripper [3]) and IoT malware (*e.g.,* Mirai Botnet [4]) which compromise confidentiality; man-in-the-middle attacks (*e.g.,* ARP spoofing [5]) which violate both confidentiality and integrity; Denial of Service (DoS) attacks (*e.g.,* HTTP flooding [6]) that compromise availability.

One of the security solutions against such network attacks is Intrusion Detection System (IDS). Depending on deployment location, IDS is classified as either Network-based IDS (NIDS) or Host-based IDS (HIDS). A HIDS resides on a single device in the network and monitors the device's state to find any suspicious activity. However, the deployment of HIDS impacts the device's resources, and protection is limited to a single device. Therefore, HIDS is not an efficient security solution for large-scale networks [7].

NIDS monitors the entire network and identifies potential threats to network devices. A typical NIDS consists of three phases: monitoring, detection, and response [8]. The monitoring phase measures the network state by collecting statistical features, such as the number of packets sent by a device or connections made. These features are used in the classification phase, where a Machine Learning (ML) algorithm determines whether the monitored features exhibit properties of a potential network attack. According to the classification phase result, the system responds with appropriate actions to defend against the attack in the response phase.

Another classification of IDS concerns the attack detection paradigm used in intrusion detection, either signature-based or anomaly-based. Signature-based NIDS (*e.g.,* Snort [9] and Suricata [10]) extracts sequences of bytes found in network traffic payload known as a "signature" in the measurement phase. The classification phase applies pattern/string matching algorithms to match the extracted signature with a predefined database of known malicious sequences. Signature-based NIDS has an extremely low error rate. However, its rigid pattern matching scheme is vulnerable to zero-day attacks that are newly discovered attacks without any fixes or patches available. Moreover, obfuscating byte signatures can effectively bypass signature-based NIDS [11].

In contrast to signature-based NIDS, anomaly-based NIDS (*e.g.,* Zeek/Bro [12]) extracts various features of benign traffic

Manuscript received 17 February 2022; revised 21 June 2022, 5 September 2022, and 7 November 2022; accepted 20 December 2022. Date of publication 3 January 2023; date of current version 24 February 2023. *(Corresponding author: Ke He.)*

Ke He is with the School of Computer Science, The University of Auckland, Auckland 1142, New Zealand (e-mail: khe429@aucklanduni.ac.nz).

Dan Dongseong Kim is with the Faculty of Engineering, Architecture and Information Technology, University of Queensland, Brisbane, QLD 4067, Australia (e-mail: dan.kim@uq.edu.au).

Muhammad Rizwan Asghar is with the School of Computer Science, The University of Auckland, Auckland 1142, New Zealand, and also with the Department of Computer Science, University of Surrey, GU2 7XH Guildford, U.K. (e-mail: r.asghar@auckland.ac.nz).

Digital Object Identifier 10.1109/COMST.2022.3233793

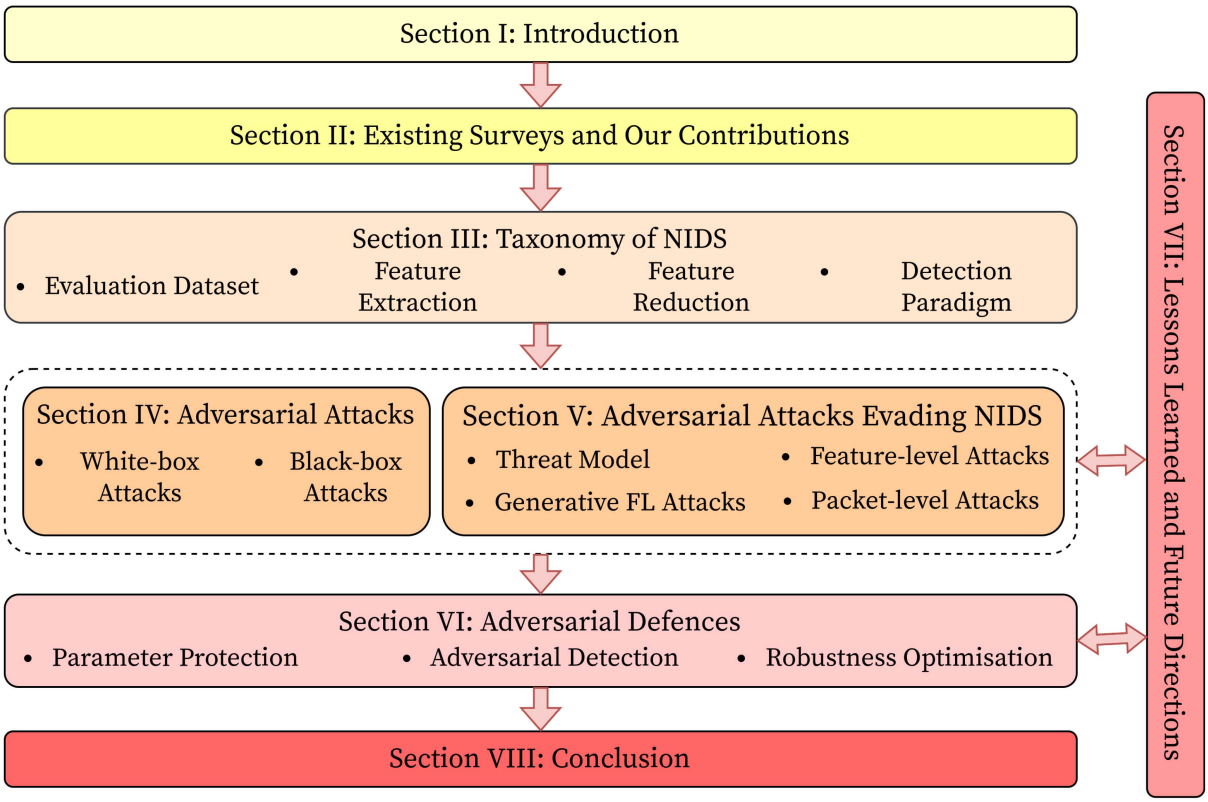


Fig. 1. Layout and structure of this survey.

data (*e.g.,* byte frequency of the payload) in the measurement phase to form a benign profile and train ML algorithms with the measured benign traffic data to detect the occurrence of network attacks. One advantage of anomaly-based NIDS is its capability to defend against zero-day attacks and signature obfuscation since they do not depend on fixed signatures for detection. However, the precise notion of normality is difficult to model or learn due to the diversity of network traffic [13], which often results in a high false-positive rate observed for anomaly-based NIDS. Additionally, anomaly-based NIDS are prone to mimicry attacks where the attacker modifies the attack features to make them similar to benign features. Examples of mimicry attacks include polymorphic blending attacks [14] and gp-based mimicry [15].

The recent trend in anomaly-based NIDS design favours adopting Deep Learning (DL) algorithms to detect potential network attacks [16]. With the exponential growth of traffic volume, shallow ML algorithms (*e.g.,* Decision Trees (DT), Support Vector Machines (SVMs), and Bayesian Networks (BN)) have become increasingly inefficient to train and test. DL offers accurate detection of network traffic by leveraging large quantities of data. The success of DL algorithms is primarily due to their ability to fully utilise large volumes of data to learn extremely abstract and non-linear representations [17] that allow for highly accurate classification. However, DL algorithms are vulnerable to adversarial examples created by adversarial attacks.[1]

Adversarial examples are small, imperceivable perturbations of the input that causes the DL algorithm to misclassify the example [18] and have exposed a new attack surface for DL-based applications. The emergence of adversarial attacks has motivated researchers to develop countermeasures to defend against adversarial attacks, especially in security-critical domains such as NIDS [19]. However, existing adversarial attacks and defences in the NIDS domain have directly adopted their formulations from CV and ignored the drastic differences in the detection pipeline and the feature space between NIDS and CV.

This article aims to bridge the gap between adversarial learning in NIDS and CV by surveying the literature (since 2015) on DL-based NIDS,[2] adversarial attacks, and adversarial defences to present a comprehensive overview of adversarial learning in DL-based NIDS. Moreover, we highlight fundamental differences between NIDS and CV that adversarial attacks and defences must consider during their formulation. Fig. 1 shows the layout of this article. We begin with a review of the previous surveys on related topics in Section II. Next, we present a taxonomy of NIDS in Section III, dissecting the NIDS into four components: evaluation datasets, feature extraction, feature reduction, and detection algorithms. Then, we describe various general adversarial attacks against DL models in Section IV, followed by NIDS-specific adversarial attacks in Section V. In Section VI, we investigate various adversarial defence mechanisms to

[1]We refer to exclusively "adversarial **evasion** attacks". Other types of adversarial attacks, such as poisoning, model inversion, and backdoor, are outside the scope of this survey.

[2]For brevity and simplicity, "DL-based NIDS" refers to anomaly-based NIDS with DL algorithms as the detection engine.

TABLE I
COMPARISON OF RECENT SURVEYS (SINCE 2015), ADVERSARIAL LEARNING, AND ADVERSARIAL LEARNING IN NIDS

Reference	Year	NIDS Taxonomy	ML in NIDS	Adversarial Attacks	Adversarial Defences	Adversarial Learning in NIDS	Challenges in Adversarial Learning in NIDS	Remarks
Buczak *et al.* [23]	2015	High	Medium	None	None	None	None	NIDS taxonomy and ML techniques used in NIDS are explained. Some of them raised the issue of adversarial attacks in NIDS. However, discussion on adversarial attacks and defences is limited.
Chaabouni *et al.* [24]	2019	High	High	None	None	None	None	
Zhang *et al.* [25]	2019	High	High	Low	None	None	None	
Hussain *et al.* [22]	2020	High	High	Low	None	None	None	
Li *et al.* [26]	2018	None	None	High	High	None	None	Adversarial attacks and defences are thoroughly explained with formulations. However, the attacks and defences are targeted at security-insensitive domains, such as CV and NLP.
Chakraborty *et al.* [27]	2018	None	None	High	High	None	None	
Wang *et al.* [28]	2019	None	None	Medium	High	None	None	
Bhambri *et al.* [29]	2019	None	None	Medium	High	None	None	
Silva *et al.* [30]	2020	None	None	Medium	High	None	None	
Serban *et al.* [31]	2020	None	None	High	High	None	None	
Zhang *et al.* [32]	2020	None	None	High	Low	None	None	
Ibitoye *et al.* [19]	2019	Low	High	High	High	High	Medium	These surveys discuss adversarial attacks and defences for NIDS, but fail to address the challenges of adversarial learning in NIDS that arise from differences between CV and NIDS.
Olowononi *et al.* [33]	2020	None	None	High	High	None	None	
Martins *et al.* [34]	2020	Low	Low	High	High	Low	None	
Alatwi *et al.* [35]	2021	None	Low	Medium	Medium	High	Medium	
Our work	2022	High	High	High	High	High	High	We present a comprehensive survey of NIDS taxonomy and discuss its impact on adversarial attacks and defences in general against NIDS. Moreover, we present exciting challenges and future directions for adversarial learning in NIDS.

protect the DL algorithms against adversarial attacks and their applicability in the NIDS domain. Section VII discusses the lessons learned from previous studies, identifies existing challenges and open problems facing adversarial machine learning in NIDS and provides research directions for future work. Finally, we conclude this article in Section VIII. In the Appendix, we present a table of acronyms used throughout the survey.

II. EXISTING SURVEYS AND OUR CONTRIBUTIONS

The success of DL algorithms was first observed in CV [20] and NLP [21] domains. Consequently, most of the research on adversarial attacks and defences was centred on these fields. During this time, NIDS still uses shallow ML methods as the detection algorithm, for example, Decision Tree (DT), Support Vector Machine (SVM), and Bayesian Networks (BN). Researchers have recently begun designing DL-based NIDS [22], where adversarial attacks are limited to naively applying existing adversarial attacks against CV models to NIDS. The threat of adversarial attacks against DL-based NIDS remains unexplored, and this article aims to bridge the gap in adversarial learning for NIDS.

We classify the existing surveys into three categories based on their covered scope, shown in Table I:

- The first category is a set of surveys on *NIDS*. In particular, these surveys provide a taxonomy of NIDS with an explanation of typical ML and DL algorithms used and the characteristics of NIDS input space [22], [23], [24], [25]. However, surveys on NIDS fail to acknowledge the effect of adversarial attacks on DL-based NIDS. In our survey, we extend the views of the previous works by discussing the impact of adversarial attacks and defences in the NIDS domain.
- The second category is a set of surveys that investigated *adversarial attacks and defences* in prevalent but security-insensitive domains such as CV and Natural Language Processing (NLP) [26], [27], [29], [30], [31], [32], [36]. They present an intuitive overview of adversarial attacks and defences against DL models. However, the discussion centred around DL algorithms in CV and NLP, which have multiple fundamental differences compared to NIDS. Our

survey expands the discussion of adversarial learning on NIDS and investigates the unique challenges faced by adversarial learning under a practical NIDS environment.

- The last category is a set of surveys that review *adversarial learning in security-sensitive domains,* such as NIDS and Cyber-Physical Systems (CPS) [19], [33], [34], [35], [37]. These surveys provide a broad yet shallow view of adversarial attacks against a wide range of security-sensitive applications and did not consider the practicality of applying adversarial attacks and defences from CV to NIDS. In contrast, our survey focuses explicitly on adversarial attacks and defences against NIDS. It provides a confined yet insightful analysis of the principles that adversarial attacks and defences against NIDS must follow.

A. NIDS Surveys

Researchers have developed various traffic classification and outlier detection algorithms to efficiently identify malicious traffic from benign traffic based on ML. The emergence of DL algorithms has triggered broad interest in the NIDS domain due to their ability to leverage large volumes of data to extract extremely abstract features that allow highly accurate detection. Previous surveys have provided a comprehensive picture of ML/DL algorithms used in anomaly-based NIDS. Buczak and Guven [23] present a comprehensive taxonomy of NIDS and existing ML and DL algorithms deployed in NIDS. Chaabouni et al. [24] examined ML algorithms used in NIDS explicitly designed for IoT devices and the unique challenges faced by NIDS in IoT environments. Hussain et al. [22] went beyond the NIDS setting and evaluated ML and DL techniques to address a wide range of security issues in IoT, such as malware analysis, authentication and access control, and intrusion detection. Zhang et al. [25] evaluated ML and DL algorithms used for various security/analytical tasks for wireless and mobile networks. Furthermore, the authors looked at tailoring DL algorithms to suit different types of mobile networks. These surveys present a comprehensive analyse of ML/DL algorithms in their respective domain; however, they fail to cover adversarial attacks and defences targeted at the DL algorithm.

B. Adversarial Attacks and Defences

Deep Neural Networks (DNNs) are prone to adversarial attacks [18], [38] constructed by adding imperceivable perturbation to an input example that causes the neural network to misclassify the input. Previous surveys in adversarial attacks and defences mainly target image classifiers [26], [27], [29], [30], [31]. Wang et al. [28] provide a taxonomy of attacks and defences against the ML models categorised by the stage of the training process that the attack targets. Furthermore, they discussed the importance of data privacy protection during training. Li et al. [26] present a mathematical review of the most notable WB and BB adversarial attacks, followed by defences that strengthen the training process or remove adversarial perturbation at test time. Chakraborty et al. [27] use a broader definition of adversarial attacks. Their analysis includes exploratory attacks that extract the model's training data, poisoning attacks that contaminate the training data to cause misclassification, and evasion attacks that alter the input to bypass detection. Following that, they describe ways to defend against adversarial attacks. Silva and Najafirad [30] examine existing WB and BB attacks, with defence focusing on methods that increase the robustness of the network. Serban et al. [31] analyse existing adversarial attacks and defences specifically in the scope of CV tasks. They have raised some controversial points on whether defending against adversarial samples is necessary for CV. Bhambri et al. [29] solely examine attacks under the BB threat model for CV tasks along with defence mechanisms. Zhang et al. [32] investigate existing adversarial attacks and defences in the image domain and seeks to tailor these attacks to textual inputs for NLP tasks. They have presented some open issues of adversarial attacks in the NLP domain that are also valid for traffic data, for example, the notion of "perceivablity" in textural inputs and the transferability of adversarial textual inputs.

C. Adversarial Learning in Cybersecurity

Adversarial attacks targeted at DL algorithms deployed in cybersecurity domains are still a largely unexplored topic. Olowononi et al. [33] provide a taxonomy of ML/DL algorithms with applications in CPS, followed by existing adversarial attacks targeted at CPS and defences that increase the resilience of ML systems against adversarial attacks. Martins et al. [34] present notable WB and BB attacks against ML/DL algorithms and their applicability in NIDS and malware detection. However, they fail to acknowledge the validity and maliciousness of the generated adversarial samples. Ibitoye et al. [19] present a comprehensive summary of ML and DL algorithms used for network security tasks and numerous adversarial attacks and defences techniques. Alatwi and Morisset [35] investigate the effect of adversarial examples on ML-based NIDS and existing attacks and defences applied in NIDS. Unfortunately, previous works have primarily listed techniques used for adversarial attacks and defences but fail to provide an intuitive and abstract view on the topic. Moreover, previous works have also failed to discuss the practicality of adversarial attacks and defences by examining the drastic difference between network traffic features and image features that cause adversarial attacks designed for images to be impractical for the NIDS domain. We have found a significant research gap in adversarial attacks and defences against NIDS and aim to bridge this gap with this article.

D. Contributions

This article focuses on adversarial attacks and defences in the NIDS domain and investigates the necessary changes in the formulation of adversarial attacks and defences to make them applicable in the NIDS domain. The main contributions of this survey are summarised as follows:

1) We present a taxonomy of DL-based NIDS to identify common design choices when developing DL-based NIDS. Furthermore, we critically analyse the design choices of DL-based NIDS under practical scenarios

TABLE II
A SUMMARY OF DL-BASED NIDS ARCHITECTURES FROM 2017 TO 2020

Solution	Year	Target Network	Evaluation Datasets	Feature Extraction	Feature Reduction	Detection Paradigms	Detection Algorithms	Application	Main Metrics
Lopez *et al.* [39]	2017	IoT	NSL-KDD	Flow-based	None	Classification	CVAE	Offline	Accuracy, F1
Yousefi *et al.* [40]	2017	General	NSL-KDD	Flow-based	AE	Classification	SVM XGBoost	Offline	Accuracy
Yin *et al.* [41]	2017	General	NSL-KDD	Flow-based	None	Classification	RNN	Offline	Accuracy, FPR, Time
Thing [42]	2017	Wireless	Own Dataset	Packet-based	AE	Outlier Detection Classification	Softmax Regression	Real-time	Accuracy
Jia *et al.* [43]	2018	General	NSL-KDD KDD Cup'99	Flow-based	None	Classification	DNN	Offline	F1, FPR, FNR
Shone *et al.* [44]	2018	General	NSL-KDD KDD Cup'99	Flow-based	SNDAE	Classification	Random Forest	Offline	Accuracy, F1, FPR, Time
Yan *et al.* [45]	2018	General	NSL-KDD	Flow-based	SSAE	Classification	SVM	Offline	Accuracy, FPR, TPR, Time
Naseer *et al.* [46]	2018	General	NSL-KDD	Flow-based	None	Classification	CNN RNN AE	Offline	Accuracy, AUC, Time
Mirsky *et al.* [47]	2018	IoT	Kitsune	Packet-based	None	Outlier Detection	AE Ensemble	Real-time	AUC, EER, Time
Diro *et al.* [48]	2018	IoT	NSL-KDD	Flow-based	None	Classification	DNN	Offline	Accuracy, F1, TPR, FPR
Otoum *et al.* [49]	2019	IoT	NSL-KDD	Flow-based	SMO	Classification	SDPN	Offline	Accuracy, F1
Xiao *et al.* [50]	2019	General	KDD Cup'99	Flow-based	PCA AE	Classification	CNN	Offline	Accuracy, TPR, FPR, Time
Zhang *et al.* [51]	2019	General	UNSW-NB15 CIC-IDS2017	Packet-based	None	Classification	CNN & caXGBoost	Real-time	Accuracy, TPR, FPR, Time
Vinayakumar *et al.* [52]	2019	General	NSL-KDD KDD Cup'99 UNSW-NB15 CIC-IDS2017 WSN-DS	Flow-based	None	Classification	DNN	Offline	Accuracy, F1
Roopak *et al.* [53]	2019	IoT	CIC-IDS2017	Flow-based	None	Classification	CNN & LSTM	Offline	Precision, Recall, Accuracy
Nguyen *et al.* [54]	2019	General	UGR'16	Flow-based	None	Outlier Detection	VAE	Offline	AUC
Sun *et al.* [55]	2020	General	CIC-IDS2017	Flow-based	None	Classification	CNN & LSTM	Offline	Accuracy, F1

and how it affects adversarial attacks and defences (Section III).

2) We provide a comprehensive review of existing adversarial attacks and defence mechanisms and investigate their applicability in the NIDS domain (Sections IV–VI).
3) We discuss major challenges faced by adversarial attacks and defences in NIDS and provide possible future research directions to overcome these challenges (Section VII).

III. TAXONOMY OF NIDS

We begin with a taxonomy of DL-based NIDS and the underlying DL algorithms to give the readers a better understanding of DL's role in NIDS. In particular, we break down the structure of NIDS into four main components: 1) Evaluation Datasets; 2) Feature Extraction; 3) Feature Reduction; and 4) Detection Paradigms and Algorithms. Table II shows a summary of the NIDS surveyed in this section.

A. Evaluation Datasets

Benchmark network datasets are essential for evaluating and comparing different NIDS approaches as well as adversarial attacks and defences. Over the last two decades, only a handful of datasets have been released. The datasets are either labelled or unlabeled. Labelled datasets contain class information of each feature, labelled as benign or a specific attack. In contrast, unlabelled datasets contain no class information, and features are collectively labelled as benign or attack (*i.e.,* the features are captured during the attack period, and specific features responsible for the attack are unknown). Typical attacks used in network datasets include, but are not limited to, scanning/probing (*e.g.,* port scanning), DoS (*e.g.,* HTTP flooding), user to root (*e.g.,* heart-bleed), remote to local (*e.g.,*

brute-force password cracking), and man-in-the-middle (*e.g.,* ARP spoofing attacks).

In the following, we present an overview of datasets used in NIDS evaluation and adversarial learning, categorised by the target network as either *general networks* or *IoT networks*.

General Network Datasets: Benchmark datasets for general networks aim to simulate network activities generated by an organisation that consists of a wide range of devices such as desktops, laptops, and printers. Devices in a general network are multi-purpose (*e.g.,* laptops can stream video, visit websites, or play online games), thus exhibiting drastically diverse traffic patterns. Commonly used datasets for general networks include NSL-KDD [56], Kyoto 2006+ [57], ISCXIDS2012 [58], CTU-13 [59], UNSW-NB15 [60], NGIDS-DS [61], TRAbID [62], and CIC-IDS2017/2018 [63].

IoT Network Datasets: IoT is an emerging network topology that connects physical objects ("Things") via software or sensors. Some of its applications include smart homes and sensor networks. IoT networks exhibit different traffic patterns than general networks since IoT devices often have limited, single-purpose functionality (*e.g.,* security cameras is only capable of streaming video footage). As a result, NIDS developed explicitly for IoT environments are evaluated on a different set of datasets, such as AWID [64], Kitsune [47], Bot-IoT [65], and TON_IoT [66].

Discussion: Benchmark datasets are essential to comprehensively evaluate the performance of NIDS and its performance under adversarial influences. However, we have found that several recently proposed NIDS systems are evaluated with inappropriate datasets. For example, NIDS designed for IoT networks are evaluated with General network datasets [39], [48], [49], [53], and a large majority of recently proposed NIDS [39], [40], [41], [43], [46], [48], [49], [50] are evaluated with NSL-KDD and KDD Cup'99 only, which is over 20 years old and are not representative of network traffic today. As a result, multiple adversarial attacks have been inappropriately evaluated on these outdated datasets [67], [68], [69]. The lack of benchmark datasets reduces the validity of NIDS, adversarial attacks, and defences. We extend the discussion on the challenges in NIDS datasets in Section VII-A1.

B. Feature Extraction

Direct detection of intrusion based on raw network packets is challenging because of the sheer volume of network data (*e.g.,* terabytes per day [23]) and limited processing capabilities. Thus, for fast network traffic detection, a common practice in NIDS is to extract features from raw packet headers and train the detection engine on the extracted features. Some NIDS also release the source code of the feature extractors publicly (*e.g.,* CIC Flowmeter [63] and AfterImage [47]).

The feature extraction process makes adversarial attacks against NIDS fundamentally different from attacks against models in CV. NIDS classifies packets based on the extracted features, yet the functional behaviour of network traffic lies within the network packets. Therefore, naively applying adversarial attacks that perturb features will not generate practical network traffic, and converting a set of adversarial features back to packets is computationally expensive [70]. Moreover, feature extraction introduces unknown relationships between the features, making the perturbation of realistic traffic features difficult. We refer to this as the *Practicality of Feature Manipulation* problem and discuss it in Section VII-B1.

The feature extraction process can be categorised via the *basic unit* of each feature, which is either *packet-based* or *flow-based*. With each type of feature extraction, we summarise the common types of features extracted and the advantages and disadvantages of the respective feature extraction.

1) Flow-Based Feature Extraction: The most common form of feature extraction is flow-based feature extraction. It extracts aggregated information of related packets within the same flow/connection. Each feature represents a connection between two devices. Flow-based features are mostly calculated from the packet header and are categorised as summary statistics, aggregate information, and statistical information.

- *Summary statistics* provides overall information about the entire connection, such as the duration of the connection and protocol/service used.
- *Aggregate information* calculates a count of the discrete features of the packets in the connection, such as the number of packets sent, the number of bytes sent, and flag counts.
- *Statistical information* measures statistics of the packets' features in the connection, for example, the mean and standard deviation of packet size and interarrival time.

In addition to statistical features from the packet header, information can be gathered from external software (*e.g.,* malware detection engine) and system API calls (*e.g.,* number of shell prompts) [56], [57]. These features profile the device's internal behaviour to provide a more comprehensive view of the network devices. However, the system API calls are different across devices. Therefore, the result of external software can be challenging to set up and replicate. Thus, external features are only applicable in specific network configurations and not for general networks.

2) Packet-Based Feature Extraction: Another feature extraction method is to produce a feature for each packet rather than flow/connection, known as packet-based feature extraction. Typical packet-based features are aggregate and statistical information similar to flow-based features with additional correlation measures of packet size and inter-arrival time between two devices. Features are calculated over various time windows to capture the packets' sequential information and reduce network traffic diversity. Packet-based feature extractors have been used in various NIDS [42], [47], [51].

Most packet-based feature extractors only examine the header information to allow efficient feature extraction. However, Zhang et al. [51] propose a novel way of representing each packet as greyscale images without feature extraction. The packet payload's hexadecimal representation is padded with 0s to make each packet 1518 bytes long. Next, the p-zigzag encoding scheme is used to create a 64×64 greyscale image. Finally, the image undergoes Inverse Discrete Cosine

Transform (IDCT) to create a pattern texture image used for classification. Unfortunately, the authors did not report on the time used for applying this transformation. It is questionable whether this method is fast enough for real-time packet extraction.

Discussion: Network traffic is diverse and heterogeneous, which hinders NIDS' detection performance [13]. Flow-based features reduce this variability by aggregating network features by connection. Hence, flow-based features exhibit consistent network behaviour that forms a stable profile of normal network activities, leading to highly accurate detection results. However, a significant limitation of flow-based feature extractors is that they must wait for the connection to finish to produce a feature. Therefore, attacks that establish extremely long connections on purpose (*e.g.,* the Slowloris attack) may not be detected in real-time. Due to the inherent delay in flow-based feature extractors and their consistent network behaviour, we believe they are more applicable for *offline* network traffic analysis than real-time intrusion detection.

In contrast to flow-based features, packet-based features are extracted for each packet on arrival. Therefore, they reflect the network's state in real-time to detect network anomalies as soon as possible and are more applicable for *real-time* detection of network attacks. However, a major weakness of packet-based feature extraction is labelling. Not all packets in an attack are responsible for malicious behaviour. Some packets can be present in both malicious and benign traffic (*e.g.,* TCP three-way handshake). Moreover, the sheer volume of packets makes manual labelling an impossible task. Due to the lack of labels available, packet-based features are often used in conjunction with outlier detection algorithms rather than classification algorithms, and we discuss this further in Section III-D.

C. Feature Reduction

An optional stage in the NIDS detection pipeline is reducing the extracted features' dimensionality. When designing a NIDS against a specific type of attack or deployed in a particular network configuration, not all extracted features are useful for achieving desired high accuracy. Removing unnecessary features typically leads to more accurate detection [71]. We refer to this process as feature reduction.

In early NIDS, feature reduction is performed manually by security experts using filter methods, where each feature is treated as an independent variable, and only the discriminative features are used in classification [72]. With the advances in computational power and DL algorithms, various NIDS systems have used DL algorithms and heuristics search algorithms to reduce input features' dimensionality. We categorise common methods for feature reduction as either *Selection* or *Projection*.

1) Selection: Feature selection refers to finding a subset of the input features that maximises the decision model's performance. The three main strategies used for feature selection are:

- *Filter methods* utilise statistical methods to score each feature as an independent variable, where features with low scores are removed. Standard filter methods include univariate statistical tests (*e.g.,* Chi-Squared and ANOVA), mutual information (*e.g.,* information gain), and correlation coefficient [73], [74], [75].
- *Wrapper methods* consider selecting relevant features as a combinatorial optimisation problem that finds the optimal subset of features to produce a model with the highest accuracy. Combinatorial optimisation problems are often solved using heuristics search algorithms that mimic animals finding prey. Common heuristics search algorithms used in NIDS include Firefly Algorithm (FA) [76], Spider Monkey Optimisation (SMO) [49], Grey Wolf Optimisation (GWO), and Particle Swarm Optimisation (PSO) [77].
- *Embedded methods* incorporate feature selection within the training stage. The decision model learns important features during training using weight regularisation that forces weights of unimportant features to be close to zero. Common regularisation methods include LASSO [78] and Elastic Net [79].

2) Projection: Feature projection refers to mapping high-dimensional features into a low-dimensional representation. Intuitively, feature projection generates compact and abstract representations of the original high-dimensional features. The projected features can be linear or non-linear combinations of the original features. Linear projections include using Principal Component Analysis (PCA) [50], [74], and non-linear projections often use AutoEncoders (AE) [40], [50] or its variants, such as Stacked Sparse Autoencoder (SSAE) [45] and Stacked Non-symmetrical Deep Autoencoder (SNDAE) [44].

Discussion: An interesting NIDS design choice is that authors often use shallow ML algorithms (*e.g.,* Random Forest (RF), SVM and DT) for detection after feature reduction. On the other hand, NIDS design that uses DL for detection often does not have a feature reduction stage. DL algorithms are designed to handle high-dimensional inputs by utilising the layering structure in DNN to extract abstract features automatically. In contrast, shallow machine learning algorithms often perform poorly in speed and accuracy when the inputs are high-dimensional. DL-based NIDS does not perform feature reduction; hence, it does not affect adversarial attack formulations. However, adversarial defences may benefit from feature reduction by removing adversarial perturbations from the input or introducing extra complexities in the model to increase the cost of generating a successful attack.

D. Detection Paradigms and Algorithms

The detection algorithm deployed in DL-based NIDS aims to detect known and unknown attacks accurately, given the provided features. Two common paradigms used in NIDS detection are traffic classification and anomaly/outlier detection. To avoid confusion with anomaly-based NIDS, we will use the term outlier detection instead of anomaly detection in the rest of the survey. In what follows, we introduce DL algorithms used in *traffic classification* and *outlier detection*, along with nuances associated with each approach.

1) Traffic Classification: Traffic classification aims to train the NIDS to detect and identify attacks when they occur.

Conventional DL algorithms used in traffic classification include Deep Neural Network (DNN), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN).

DNN is the most basic neural network architecture. DNNs are feed-forward, multi-layer neural networks and can be modelled as a function $F(x) = y$, where $x \in \mathbb{R}^n$ is the input of size n and $y \in \mathbb{R}^m$ is the output of size m, corresponding to the number of classes in classification. $F(x)$ consists of several layers. Within each layer, numerous neurons take input from the previous layer and produce an output for the next layer. The objective of a DNN is to learn the underlying mapping from x to y with $F(\cdot)$, and optimisation of DNN is made using some loss function that minimises the difference between the ground-truth label and the predicted label y. DNNs have been frequently used in NIDS with different parameters and structures proposed [43], [49], [52], [68].

RNN is a class of neural networks specialised for tasks involving sequential data, such as NLP. The philosophy of RNN is to maintain a "memory" of previously processed information, which can aid detection. The "memory" mechanism is accomplished by maintaining a hidden state within each neuron that is updated based on the previous hidden state and the current input. Unfortunately, training RNN has proven to be problematic [80]. Memorising the system's long-term behaviour is difficult because the gradients either grow or shrink at each training step, causing the gradient to explode or vanish over time. This phenomenon is primarily due to the gradient passing through numerous multiplicative stages during training and causes the gradient to be continuously compounded to either vanish or explode [81].

Two types of memory units are designed to combat vanishing and exploding gradients: Long Short-Term Memory (LSTM) units [82] and Gated Recurrent Unit (GRU) [83], where GRU is a simplification of LSTM. These memory units use a unique additive structure to compute the gradient instead of the multiplicative structure used in standard RNNs. To further enhance the capability of RNNs, an attention mechanism [84] is designed to help RNNs remember multiple past keywords. Researchers quickly discovered that attention alone is powerful enough for NLP tasks [85], which gives birth to transformer models. Transformers are novel neural network architectures designed purely with the attention mechanism and did not rely on any RNN models, allowing inputs to be processed in parallel to train large datasets efficiently.

Similar to NLP, network traffic data also contains sequential information that are helpful for detection. However, despite the RNN-based advances in the NLP domain, NIDS designs with RNN as the detection algorithm still use LSTM models or plain RNN for detection [41], [53], [55]. We encourage future research to incorporate advanced NLP techniques, such as attention mechanisms and transformers, into RNN-based NIDS.

CNN specialises in tasks involving locally related data, such as object detection and image classification. The architecture of CNN consists of convolutional layers, pooling layers, and fully connected layers. The convolutional layers capture local features in the input with multiple convolutional filters. The pooling layers abstract these features with max pooling. Once the abstract features are extracted, they are classified by the fully connected layers with a similar structure as DNN. Intuitively, the convolutional layers extract simple features such as edges and corners in the image, followed by the pooling layers that abstract the edges and corners into simple objects such as squares and circles. This feature extraction and abstraction process are repeated several times to create increasingly complex and abstract features, thus acting as an automatic feature extractor on the image. Finally, the extracted features are classified with fully connected layers to produce a label.

CNN is most frequently used in image classification/video object detection tasks as it captures local spatial information with convolutional filters. Network traffic features do not contain any local spatial relations. Thus, it is typically used in hybrid systems with RNN, where CNN acts as a feature extractor. The output of CNN is classified with RNN/LSTM networks [53], [55].

Interestingly, a few designs of NIDS have used pure CNN as the classification algorithm. Xiao et al. [50] used dimensionality reduction techniques and mapped the reduced features to a 2D matrix to make the input feature suitable for CNN. The classification is done with LeNet-5. Zhang et al. [51] transformed raw packets into images via p-zigzag encoding and IDCT (Section III-B), and use GoogLenet to classify the images. Unfortunately, the authors failed to analyse the transformation time, and whether it is practical for online detection of malicious traffic is unknown.

2) Outlier Detection: The outlier detection approach aims to establish a normal network traffic profile and report an alarm when abnormalities are detected. Most outlier detection-based NIDS uses statistical measures such as K-Nearest-Neighbour (KNN), One-Class SVM (OC-SVM), and the Markov chain model [86]. DL-based outlier detection NIDS is an emerging topic, and existing solutions often use a variation of autoencoders [47].

AE is a feed-forward multi-layer neural network that learns identity mapping from input to output. The identity mapping constructed via a pair of encoders $(E(\cdot))$ and decoders $(D(\cdot))$:

$$\vec{x'} = D(E(\vec{x}))$$

The objective of AE is to reconstruct the input, and optimisation of AE is performed by minimising the distance between the input and reconstructed output. A common distance measure is the ℓ_2 distance, known as the Euclidean distance (see Section IV-A for more distance measures). The magnitude of the distance between the original and reconstructed input is known as the reconstruction error.

Learning the identity function seems to be a redundant task, but the beauty of AE is that the hidden layer's dimensions are much lower than the input. Hence, the hidden layers are forced to learn a low-dimensional and non-linear representation of the input, effectively reducing input dimensionality. Indeed, AEs are commonly used for dimensionality reduction tasks. Multiple DL-based NIDS have used AEs in the feature reduction step to produce a compact representation of traffic data for highly accurate detection of attacks, as shown in Table II [44], [45], [46].

When used as an outlier detection algorithm, the encoder and decoder are trained on only benign data so that the reconstruction of input is optimised solely for benign data. A threshold value, h, is determined based on the benign data's reconstruction error. Any reconstruction error above the threshold is considered abnormal. When an abnormal sample occurs during the execution stage, $D(\cdot)$ and $E(\cdot)$ will fail to produce the identity mapping for the abnormal sample, leading to a high reconstruction error significantly above h, and the sample will be classified as abnormal. Multiple DL-based NIDS have been proposed based on AEs [39], [47], [54].

Discussion: Existing DL-based NIDS solutions have primarily used classification algorithms to classify the network traffic because it shows better performance with more interpretable output than outlier detection. However, classification algorithms are challenging to train under a realistic scenario as gathering correctly labelled network datasets is troublesome. Moreover, the evaluation of classification-based NIDS has largely ignored zero-day attacks, where the attack traffic is generated with network attacks that the NIDS has not seen before. Hence, outlier detection algorithms are more practical in the NIDS domain as they do not require labelled data and are inherently robust to zero-day attacks. However, the diverse network traffic pattern makes it extremely challenging for the anomaly detection system to learn a stable notion of normality. The choice of detection paradigm also affects the evaluation of adversarial attacks and defences. Existing adversarial attacks and defences are mostly evaluated against classification models, where the strength of attacks/defence can be easily measured with AUC or F1 measures. However, it is difficult to accurately evaluate the attack/defence strength for anomaly detection models since the ground truth labels are unknown. We elaborate on this further in Section VII-A2.

IV. Adversarial Attacks

The term "adversarial examples" is first coined by Szegedy et al. [18]. They have found that applying an imperceivable perturbation to input can cause the classifier to misclassify the input as an arbitrary class. This intriguing property of neural networks has attracted considerable attention, but the exact cause of adversarial samples remains an open question today. Szegedy et al. [18] claim that the mapping from the neural network's input to output is discontinuous to a significant extent. These are discontinuities that cause DNN to misclassify the adversarial sample. In contrast, Goodfellow et al. [38] claim adversarial samples are not caused by highly non-linear decision functions but due to the locally linear nature of neural networks. Summing small perturbations linearly in high-dimensional space pushes the sample in a direction that will cause misclassification. The lack of understanding of adversarial examples requires more fundamental research across all domains.

In the following, we present a high-level overview of the *fundamental theory* behind adversarial attacks, followed by explaining *white-box* (WB) and *black-box* (BB) threat models and attack methods associated with a respective threat model.

A. Foundations of Adversarial Attacks

An adversarial attack modifies the original example, $\vec{x_0} \in \mathbb{R}^d$, classified as some class, C_o, to an adversarial example, $\vec{x_0'}$, classified as another class, C_t [87]. In some scenarios, the goal is not to perturb $\vec{x_0}$ to a specific target class but to push it away from its original class C_o, known as an untargeted attack. Similarly, when a specific target class is identified, it is known as a targeted attack. Adversarial attacks can be formulated as finding the mapping $A : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that $\vec{x_0'} = A(\vec{x_0})$ is misclassified as C_t for targeted attacks or not C_o for untargeted attacks.

Adversarial attacks often define A as an additive mapping, $\vec{x_0'} = \vec{x_0} + \vec{r}$ for $\vec{r} \in \mathbb{R}^d$ where $\vec{r}$ is the perturbation vector. Finding $\vec{r}$ can be formulated as a constrained optimisation problem involving two constraints: similarity and confidence.

Confidence constraint: Consider a multi-class classifier where the classes are defined as $C_1, C_2, \ldots, C_k$ and the classifier's confidence for each class is modelled by the function $g_i(\cdot)$ for $i = 1, 2, \ldots, k$. If we want the classifier to classify $\vec{x_0'}$ as C_t, then the confidence functions have to satisfy $g_t(\vec{x_0'}) \geq g_j(\vec{x_0'}) \forall j \neq t$. In other words, the target class has the highest confidence across all classes. The inequality can be reformulated as the constraint:

$$\max_{j \neq t} \left\{ g_j\left(\vec{x_0'}\right) \right\} - g_t\left(\vec{x_0'}\right) \leq 0 \tag{1}$$

For untargeted attacks, the confidence function has to satisfy $g_i(\vec{x_0'}) \leq g_j(\vec{x_0'}) \forall j \neq i$. In other words, the original class has the lowest confidence across the classes. The inequality can be reformulated as the constraint:

$$g_i\left(\vec{x_0'}\right) - \min_{j \neq i} \left\{ g_j\left(\vec{x_0'}\right) \right\} \leq 0$$

The confidence constraint for targeted and untargeted attacks is interchangeable depending on the attacker's goal. For simplicity, we assume all the attacks we describe in this section are targeted attacks and use the targeted version of the confidence constraint (Equation 1). Fig. 2 illustrates the confidence constraint for targeted and untargeted attacks.

Similarity constraint: The generated adversarial example has to be similar to the original example. For image data, a common measure of perturbation is the ℓ^p norm ($\|\cdot\|_p$) of $\vec{r}$, defined as

$$\|\vec{r}\|_p = \left(\sum_{i=1}^{n} \|r_i\|^p \right)^{\frac{1}{p}}$$

Common choices for p are 0, 1, 2, and ∞. The ℓ^0 distance measures the number of dimensions that have different values between two points. The ℓ^1 distance is the sum of the absolute value of each dimension, commonly known as the Manhattan distance. The ℓ^2 distance is the root mean squared distance between two points, commonly known as the Euclidean distance, and is the most frequently used distance measure. The ℓ^∞ distance measures the maximum absolute difference among all dimensions. With the measure of distance, we define the similarity constraint as

$$\left\| \vec{x_0'} - \vec{x_0} \right\| < \epsilon \tag{2}$$

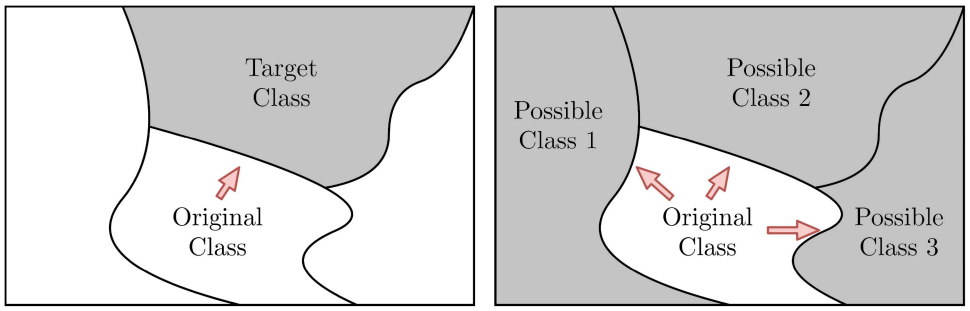
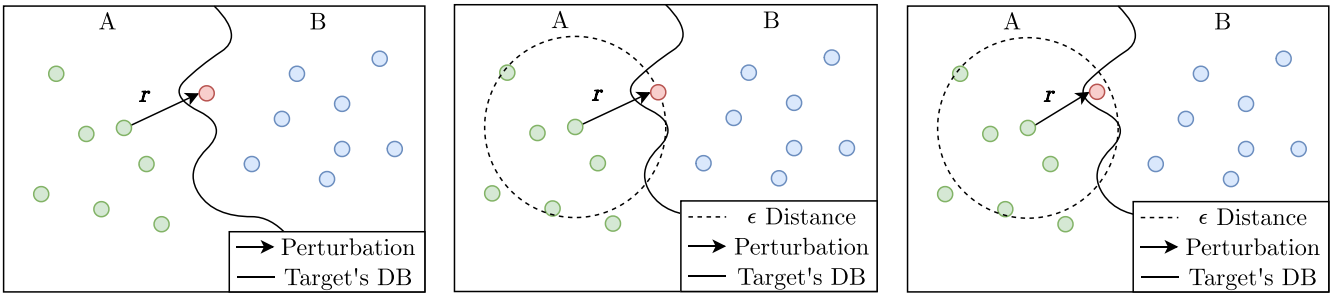


Fig. 2. Visualisation of the confidence constraint for targeted and untargeted attacks. Shaded regions indicate that the confidence constraint is satisfied. Left: targeted attack from the original class to the target class. Right: untargeted attack to anywhere but the original class.



(a) Minimum Norm Attack: Minimise the perturbation $\boldsymbol{r}$ while the adversarial example is being classified as the target class, B.

(b) Maximum Allowable Attack: Maximise the confidence of the adversarial example being classified as the target class, B, while constraining the perturbation $\boldsymbol{r}$ within ϵ.

(c) Regularisation-Based Attack: Adversarial example is generated by balancing the similarity and confidence constraints.

Fig. 3. Three WB attack types based on similarity and confidence constraints: The victim DL model has learned the Decision Boundary (DB) between two classes, A (green) and B (blue). The adversarial attack tries to generate an adversarial example (red) classified as class B with imperceivable perturbation from the original example classified as class A.

where ϵ is the maximum magnitude of perturbation and $\|\cdot\|$ is any of the ℓ^p distance metric.

B. White-Box Attacks

WB attacks are often used to analyse and gain insight into adversarial examples. Thus, it assumes to have complete knowledge of the classifier. Under a WB threat model, the attacker has the following goal, knowledge, and capability:

1) *Goal:* The attacker's goal is to generate an adversarial example that satisfies both confidence and similarity constraints.
2) *Knowledge:* The attacker has perfect knowledge of the model's parameters.
3) *Capability:* The attacker can manipulate every feature of the input.

In the following, we categorise and present WB attacks based on the constraint being emphasised.

Minimum norm attack: The minimum norm attack prioritises the similarity constraint. As shown in Equation (3), it emphasises the perturbation introduced in the adversarial sample as long as the sample is classified as the target class.

$$\underset{x}{\text{minimise}} \quad \left\|\vec{x_0'} - \vec{x_0}\right\| \tag{3a}$$

$$\text{subject to} \quad \max_{j \neq t}\left\{g_j\left(\vec{x_0'}\right)\right\} - g_t\left(\vec{x_0'}\right) \leq 0 \tag{3b}$$

Geometrically, the adversarial sample lies near the point on the decision closest to the original sample, shown in Fig. 3(a). A notable adversarial attack based on minimum norm attack formulation is DeepFool [88].

Maximum allowable attack: In contrast to the minimum norm attack, the maximum allowable attack prioritises the confidence constraints. It aims to maximise the model confidence in the target class within some perturbation level, as shown in Equation (4).

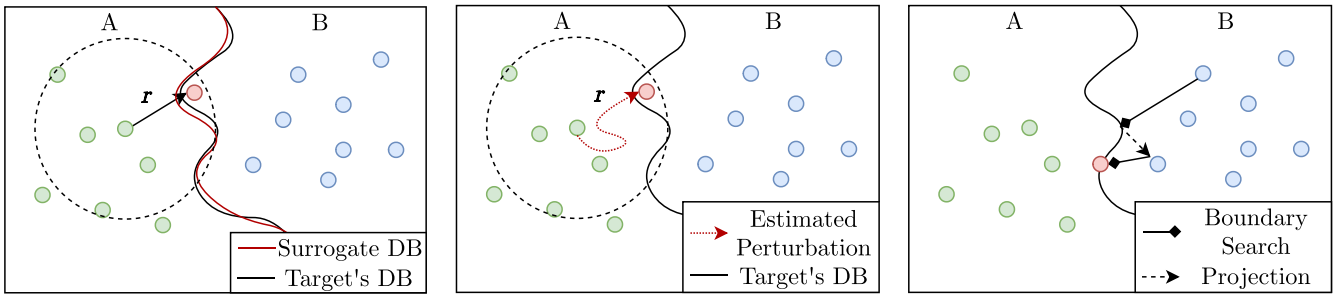
$$\underset{x}{\text{minimise}} \quad \max_{j \neq t}\left\{g_j\left(\vec{x_0'}\right)\right\} - g_t\left(\vec{x_0'}\right) \tag{4a}$$

$$\text{subject to} \quad \left\|\vec{x_0'} - \vec{x_0}\right\| \leq \epsilon \tag{4b}$$

Geometrically, the adversarial sample lies near the point of highest confidence of C_t bounded by ϵ, shown in Fig. 3(b). Notable adversarial attacks based on maximum allowable attack include Fase Gradient Sign Method (FGSM) [38], Jacobian Saliency Map Attack (JSMA) [89], and Basic Iterative Method (BIM) [90].

Regularisation-based attack: The third type of attack is the regularisation-based attack, which carefully balances the similarity and confidence constraints to allow a more finely crafted adversarial example. Equation (5) presents the formulation of a regularisation-based attack.

$$\underset{x}{\text{minimise}} \quad \|\vec{x_0'} - \vec{x_0}\| + \lambda\left(\max_{j \neq t}\left\{g_j\left(\vec{x_0'}\right)\right\} - g_t\left(\vec{x_0'}\right)\right) \tag{5}$$



(a) Transfer-based attacks [94], [97]: WB attacks are conducted on a substitute model with similar decision boundaries to the victim model. The adversarial example generated with the surrogate model is also adversarial for the target model.

(b) Score-based attacks [96], [98], [99]: These attacks use estimated gradients (dashed line) in WB formulation to find the adversarial example. Since the gradients are estimated, it will take longer to find the adversarial example.

(c) Decision-based attacks [95], [100]: These attacks find the boundary between the original and random example in the target class (blue) and traverse the decision boundary between the two classes to produce the adversarial sample.

Fig. 4. Three main categories of BB attacks: transfer-based, score-based and decision-based. The victim DL model has learned the decision boundary (black) between two classes, A (green) and B (blue) and the BB attack tries to generate an adversarial example (red) based on the original example without knowing the true decision boundary.

where λ is the regularisation parameter that balances the relative weight between the two constraints. Geometrically, regularisation-based attacks find adversarial examples between maximum allowable and minimum norm, shown in Fig. 3(c). Notable regularisation-based attacks include the C & W attack [91] and EAD [92].

Under the WB threat model, the solution to the optimisation problems is found by applying gradient descent algorithms [93] since the gradient of the model can be easily obtained under the white-box threat model.

C. Black-Box Attacks

BB attacks simulate a more realistic attack scenario in which the attacker has zero knowledge of the classifier. Thus, the gradient of the decision function is unknown to the attacker, and the adversarial example cannot be simply obtained via gradient descent. To overcome limited classifier knowledge, researchers have developed various BB attacks [94], [95], [96].

Under a BB threat model, the attacker's goal, knowledge, and capability are defined as follows:

1) *Goal:* The goal of the attacker is the same as the WB counterpart. To generate an adversarial example that satisfies both similarity and confidence constraints.
2) *Knowledge:* The classifier is treated as an oracle, O, and the attacker knows nothing about its parameters. However, an educated guess can be made to determine the general task the neural network is trying to solve (*e.g.,* object detection or image classification).
3) *Capability:* The attacker can manipulate every feature of the input. Furthermore, the attacker can make a limited amount of queries to O and obtain the output. The output can be either:
   - The model's confidence score, which forms the basis of *score-based attacks*.
   - The predicted label, which forms the basis of *decision-based attacks*.

Transfer-based Attacks: Transfer-based attacks create a substitute model with similar decision boundaries as the target model and apply white-box attacks on the substitute model to generate adversarial examples. Transfer-based attacks leverage the adversarial transferability property of adversarial examples across neural network models [94], where adversarial examples generated for a model are also adversarial for another model with a completely different network structure. Therefore, by leveraging transferability, the attacker can create adversarial examples on the substitute model that is also adversarial to the target model. The general outline of transfer-based attacks is provided below [97], and an illustration of transfer-based attacks is presented in Fig. 4(a).

1) Guess the task of the target model, *e.g.,* image classification or sentiment analysis.
2) Develop a deep learning model suitable for the task, which we call the substitute model.
3) Train the substitute model on the dataset with the oracle's output label/score.
4) Apply WB attacks (see Section IV-B) on the substitute model.
5) Adversarial examples generated for the substitute model are transferable to the target model.

The performance of transfer-based attacks depends mainly on how well the substitute model mimics the decision boundary of the target model. However, the attacker only has access to a limited number of queries to the oracle. Hence, the research direction for transfer-based attacks focuses on reducing the number of queries made to the oracle while improving the substitute and target model's similarity. Papernot et al. [94] proposed the Jacobian-based Dataset Augmentation (JDA) method to train the substitute model. JDA aims to obtain a concise and representative dataset by augmenting the initial subset of samples towards the substitute model's decision boundary and querying the target model for the labels of the augmented samples.

Score-based Attacks: Score-based attacks directly apply the white-box attack formulations (Section IV-B) and estimate the gradient of the decision functions $g_i(x)$ by querying the target model with various points near x. As a result, score-based attacks do not rely on the quality of the substitute model and

potentially outperform transfer-based attacks in terms of attack success rate and distortion since it does not introduce any performance loss in transferability [96]. Fig. 4(b) illustrates score-based attacks.

Accurate gradient estimation is crucial for the performance of score-based attacks, and multiple gradient estimation methods have been proposed. Narodytska and Kasiviswanathan [99] apply local-search algorithms [101] to measure the influence of a subset of pixels on the output and implicitly compute an approximation to the gradient. Chen et al. [96] propose a Zeroth Order Optimisation (ZOO) [102] attack that estimates the gradient via the difference between the output scores of the inputs near x. Ilyas et al. [98] use Natural Evolution Strategy [103] to optimise the attack formulation's expected value under the search function.

Decision-based Attacks: Decision-based attacks are a more robust form of score-based attacks requiring only the output label rather than the scores. Intuitively, decision-based attacks can be thought of as traversing the boundary between the target and original class to find an adversarial sample closest to the original sample. In order to do so, decision-based attacks begin with any sample in the target class and the original sample. Next, the sample in the target class is perturbed in the direction of the original sample by calculating the difference between the two samples. Once the target sample reaches the boundary between the original and target classes, it is projected along the boundary. Finally, adversarial examples with minimum perturbation can be found by traversing the boundary. Fig. 4(c) illustrates the process of decision-based attacks.

A crucial component of decision-based attacks is finding the direction to project $\vec{x}$. In the original implementation of the Boundary Attack by Brendel et al. [100], the direction is drawn randomly from a proposal distribution. Later, the direction of projection is improved by Chen et al. in the HopSkipJump attack [95], where the direction is more accurately estimated by sampling random points near the boundary and Monte Carlo estimate.

V. Adversarial Attacks Evading NIDS

In this section, we review existing adversarial attacks designed to evade NIDS. We first present common threat models used in NIDS-specific attacks and their distinctions from threat models defined in CV, followed by an overview of NIDS-specific attacks categorised by the perturbation method. Table III provides an overview of NIDS-specific attacks described in this section.

A. Threat Model

Goal: The attacker's goal is to generate an adversarial example that satisfies both confidence and similarity constraints. For image inputs, the similarity constraint ensures the image preserves the original semantics and can be easily bounded by limiting the perturbation, say less than some threshold in the ℓ_p norm. However, defining a unified notion of similarity is an open problem for network traffic data. Existing NIDS adversarial attacks often place the similarity constraint between the adversarial and original features. In particular, there are three main methods to place the similarity constraint:

- Bounded by ℓ_p *ball*, where the network features are freely perturbed within an ℓ_p norm ball.
- Bounded by the *generative model*, where the network features are constrained by generative models such as GAN to be similar to realistic network features.
- Bounded by the *packet obfuscation*, where packet-level perturbations are bounded by pre-defined mutation operations.

Placing the similarity constraint purely in the feature space is insufficient since the network features are highly complex and rarely examined by humans. Therefore, a more appropriate approach is to place the similarity constraint on the semantic behaviour of the adversarial attack compared to the original, which we elaborate on more in Section VII-B3.

Knowledge: Threat models in adversarial attacks against CV primarily consider the attacker's knowledge of the classifier. Although CV models often apply preprocessing functions to the input, they are simple, reversible functions that transform the input image into another image. However, in NIDS, the preprocessing function (*i.e.,* feature extraction and reduction) is irreversible and transforms input packets into features. Moreover, the exact type of features extracted is up to the design of NIDS and is not standard. Therefore, knowing the feature extraction and reduction stages can significantly help the attacker produce realistic features. Based on the knowledge of the attacker, we classify NIDS-specific attacks as

- *Black-box*, if the attacker has zero knowledge of the classifier and its auxiliary components.
- *Grey-box*, if the attacker has zero knowledge of the classifier but has some knowledge of the preprocessing functions.
- *White-box*, if the attacker has complete knowledge of the classifier and auxiliary components.

Capability: Adversarial attacks against CV often assume the attacker does not have access to the training data. However, it is reasonable for the attacker to be able to access the training network data. For example, if the target network is connected wirelessly, it is reasonable for the attacker to use WiFi sniffers to sniff benign packets silently. Therefore, we categorise the capability of the attacker based on its access to training data as:

- *None*, if the attacker cannot obtain the training dataset, or it is not necessary (*e.g.,* the attacker has WB knowledge)
- *Passive*, if the attacker can capture both benign and malicious traffic.

B. Feature-Level Attacks (FLA)

The first category of attacks is a set of feature-level attacks that directly perturb the input network traffic features. These attacks follow a very similar pattern. The authors assume a WB threat model and conduct various vanilla WB attacks (*e.g.,* FGSM [67], [68], [106], [108], JSMA [67], [68], [106], [115], DeepFool [68], C & W attack [68], [106], BIM [108], PGD [108], and EAD [106]) to some target NIDS (*e.g.,* shallow ML algorithms [67], MLP [68], [115], [116], LSTM [115], CNN [115], DNN [108], SNN [108], and Kitsune [106]) on

TABLE III
COMPARISON OF ADVERSARIAL ATTACKS DESIGNED TO BYPASS NIDS

Author	Year	Type	Constraints	Knowledge	Capability	Target Model	Datasets	Metrics	Main Limitations
Rigaki *et al.* [67]	2017	FLA	ℓ_p ball	WB	None	Shallow ML	NSL-KDD	Accuracy, F1, AUC	Outdated dataset
Homoliak *et al.* [104]	2018	PLA	Packet obfuscation	GB	None	Shallow ML	ASNM-NPBO	F1	Vigorous Trial and Error
Lin *et al.* [105]	2018	GFLA	Generative model	GB	Passive	Shallow ML	NSL-KDD	Accuracy	Outdated dataset
Wang [68]	2018	FLA	ℓ_p ball	WB	None	MLP	NSL-KDD	Accuracy, F1, ROC	Outdated dataset
Yang *et al.* [69]	2018	FLA	ℓ_p ball	GB	Passive	DNN	NSL-KDD	F1	Outdated dataset
Clements *et al.* [106]	2019	FLA	ℓ_p ball	WB	None	Kitsune	Kitsune	Accuracy, ℓ_p distance	Unrealistic adversarial features
Hashemi *et al.* [107]	2019	PLA	Packet obfuscation	WB	None	Kitsune, DAGMM, BiGAN	CIC-IDS2018	TPR, FPR	Vigorous Trial and Error
Ibitoye *et al.* [108]	2019	FLA	ℓ_p ball	WB	None	FNN, SNN	BoT_IoT	Accuracy, F1, Cohen's Kappa, MCC	Unrealistic adversarial features
Kuppa *et al.* [109]	2019	PLA	MAA+Packet obfuscation	GB	Passive	AGMM, AE, AnoGAN, ALAD, DSVDD, shallow ML	CIC-IDS2018	F1	Lack of maliciousness evaluation
Alhajjar *et al.* [110]	2020	GFLA	Generative model	GB	None	Shallow ML	NSL-KDD, UNSW-NB15	Accuracy, FNR	Lack of replayability
Han *et al.* [70]	2020	PLA	Generative model + Packet obfuscation	BB/GB	Passive	Kitsune	Kitsune	FNR, ℓ_2 Distance	Lack of maliciousness evaluation
Chen *et al.* [111]	2021	GFLA	Generative model + Domain constraints	BB	Passive	Shallow ML	CTU-13	MAPE, FNR	Lack of replayability
Sharon *et al.* [112]	2021	PLA	LSTM	BB	Passive	Kitsune, AE, IF	Kitsune, CIC-IDS2017	TPR	Only considers time delay
Sheatsley *et al.* [113]	2022	FLA	Extracted constraints	WB	None	Shallow ML	NSL-KDD, UNSW-NB15	Accuracy	Lack of replayability
Zolbayar *et al.* [114]	2022	GFLA	Generative model + Domain constraints	WB/GB/BB	Passive	DNN, MLP	NSL-KDD, CIC-IDS2018	FNR	Lack of replayability

various benchmark datasets (*e.g.,* NSL-KDD [67], [68], [116], SDN [115], BoT_IoT [108], and Kitsune [106]). The similarity constraint is satisfied by restricting the degree of perturbation to be within an ℓ_p norm ball, in a similar manner to attacks in CV.

Yang et al. [69] assume a GB threat model where the attacker knows the feature extractor but not the classifier and conducts three attacks on a DNN model trained on NSL-KDD: 1) a transfer-based attack that leverages adversarial transferability [117] for training a surrogate model of a different structure with NSL-KDD; 2) a score-based attack that estimates the gradient with symmetric difference quotient called ZOO [96]; 3) an attack based on Wasserstein GAN [118]. However, the NSL-KDD dataset is extremely outdated, and the output features cannot be used to conduct the actual attack.

To generate realistic features and take into account the constraints posed by network packets, Sheatsley et al. [113] analysed the network traffic to extract constraints on the features. Based on these constraints, they propose Augmented JSMA (AJSMA), which checks if any predefined constraints are violated in each iteration of JSMA. However, these perturbations are still at the feature level and do not generate malicious flows/packets that can be used to conduct the actual attack.

Discussion: Although these studies show WB and BB attacks with feature-level perturbations are effective, they have a common limitation: lack of practicality. The generated examples are constrained within an ℓ_p norm ball with additional constraints on discrete values, which cannot fully capture the inherent structures within network features. In contrast to unstructured image data, network traffic data are structured, and features are inherently related. Consider a trivial feature extractor that extracts aggregate information such as the number of packets sent and the number of bytes sent. Naturally, these two features are positively correlated. It is nearly impossible to have a high amount of bytes sent and a low number of packets sent. These inherent relationships make the network traffic space low dimensional, incomplete,

unstable, and irregular (see Section VII-B for a detailed discussion of these characteristics). Therefore, simple perturbations of network features within an ℓ_p norm ball are likely to result in unrealistic features that cannot be replicated in real life. To generate more realistic features, Generative Feature-level Attacks are proposed.

C. Generative Feature-Level Attacks (GFLA)

Generative Feature-level Attacks utilize generative algorithms such as Generative Adversarial Networks (GAN) to learn the inherent structure within network traffic features and enforce the features to look realistic. The attack formulation of GFLA consists of a Generator that generates synthetic network features and a Discriminator that tries to distinguish whether the input is benign and realistic. To train GAN, the Generator first generates a random input, which the Discriminator easily detects as not benign and realistic. The Discriminator then passes its loss back to the Generator, and the Generator generates a new set of inputs that reduces the loss. This process is repeated until the Discriminator cannot distinguish the input, which indicates the Generator has generated realistic adversarial features.

Lin et al. [105] assume a GB threat model, where the attacker knows the features being extracted but treats the classifier as an oracle. The attack is launched against seven shallow ML classifiers trained on NSL-KDD: SVM, NB, MLP, Logistic Regression (LR), DT, RF, and KNN. A WGAN is used to attack these classifiers. The generator generates adversarial features with malicious features and some noise. Next, the BB target classifier classifies the adversarial and benign features to obtain the predicted labels. The discriminator then classifies the labels. The resulting adversarial features will closely mimic real network traffic by minimising the loss of the discriminator and generator.

Alhajjar et al. [110] extended the work of Lin et al. [105] in several aspects. First, four target classifiers were added to cover a wide range of shallow ML algorithms: gradient boosting, linear discriminant analysis, quadratic discriminant analysis, and bagging. Second, two evolutionary algorithms were proposed to generate the adversarial data: Genetic Algorithm and Particle Swarm Optimisation. Evolutionary algorithms iteratively update their population based on a fitness function (in this case, the classifier's output) to find the most optimal candidate. The input features are split into mutable and immutable parts to ensure network semantics are preserved, and the evolutionary algorithm only modifies the mutable features. Finally, the adversarial examples are evaluated on UNSW-NB15 as well as NSL-KDD.

Cheng et al. [111] propose Attack-GAN that generates the adversarial attack by considering byte-level modifications and sequential information within the bytes. In addition to the constraints learned and imposed by the Generator, Attack-GAN introduces predefined network feature constraints. For example, the TTL field of the packets can only be 0x20 or 0x80, and the functional bytes of the packet is fixed.

Zolbayar et al. [114] propose NIDSGAN that contains predefined domain constraints. They introduce additional terms in the loss function for more evasive attacks. In addition to the standard loss function that makes the discriminator unable to distinguish between the adversarial and benign traffic, the loss function also minimises the distance between the adversarial and original features.

Discussion: The primary role of generative algorithms in generative feature-level attacks is to capture the hidden interdependencies between network features and constrain the feature-level perturbations to be more realistic. However, despite realistic adversarial network features, there is still a significant gap in transforming the network features into replayable packets. Network features are abstract representations of network packets and are used solely to detect intrusion. They do not contain practical significance and cannot be transmitted between devices to conduct network activities. Thus, for any feature-level attacks to be practical, one must find the inverse function of the feature extraction and feature reduction process that generates a set of packets for generating adversarial features. Unfortunately, this process is computationally hard to find [70]. Therefore, practical adversarial attacks should be conducted at packet-level rather than feature-level.

D. Packet-Level Attacks (PLA)

To ensure the practicality of the generated adversarial example, researchers have created adversarial attacks that manipulate the network packets directly rather than network features. Homoliak et al. [104] consider a GB threat model and attack five shallow ML classifiers (NB, DT, SVM, LR, and NB with kernel density estimation) on a self-gathered dataset named ASNM-NPBO. The attacks are conducted by randomly applying non-payload based obfuscations to the packets and checking if it evades the classifier. The obfuscations include packet time delay, packet loss, packet damage, packet duplication, packet reordering, and payload fragmentation.

Hashemi et al. [107] assume a WB threat model and attack three NIDS: Kitsune [47], DAGMM [119], and BiGAN [120] on CIC-IDS2018. The attack takes a similar approach as Homoliak et al. [104], but with only three mutation operations: delay a packet, split a packet, or inject new packets. The adversarial examples are generated by iteratively trying and testing one of the mutation operations and checking whether the target NIDS's anomaly score is lower than the threshold after the mutation.

Kuppa et al. [109] propose a GB attack and attack seven different anomaly detectors: AE, DAGMM, AnoGAN, ALAD, DSVDD, OC-SVM, and isolation forest. The attack was conducted on the CIC-IDS2018 dataset. Unlike previous packet-level attacks that modify packets directly, Kuppa et al. first find data distributions in the labelled data with Manifold Approximation Algorithm (MAA) that transforms the decision space into piece-wise local spherical subspaces. Based on the subspaces, the attack generates an adversarial example with the smallest spherical distance to the original example. The adversarial features are realised back into packets by modifying time-based features in the original pcap file.

Han et al. [70] developed a novel BB/GB adversarial traffic generation framework called Traffic Manipulator and evaluated

TABLE IV
COMPARISON OF VARIOUS DEFENCE TECHNIQUES AGAINST ADVERSARIAL ATTACKS AND THEIR APPLICABILITY IN THE NIDS DOMAIN

Type	Methods	Brief Description	Defence Strength	Applicability	Section
Parameter Protection	Gradient Masking	Hides the gradient of the model from attackers to prevent WB attacks.	Weak defence. BB attacks do not require gradient and can easily bypass gradient masking.	Applicable since gradient masking is domain agnostic.	VI-A
Adversarial Detection	Secondary Classifier	Uses another neural network to classify clean and adversarial examples	Weak defence. The attack can be formulated to simultaneously bypass both detectors under the limited-knowledge and perfect-knowledge threat model.	Not applicable. Adversarial examples are not available at training time.	VI-B1
	Projection-Based	Projects input into low-dimensional space to detect adversarial examples.	Weak defence. Characteristics in low-dimensional space are specific to datasets.	Applicable but not generalisable.	VI-B2
	Statistics-Based	Finds distributional differences between clean and adversarial examples.	Weak defence. Statistical characteristics are specific to datasets.	Applicable but not generalisable.	VI-B3
	Mutation-Based	Mutates the decision boundary randomly to detect adversarial example.	Strong defence. The adversarial attack has to be universally adversarial to bypass this defence.	Applicable.	VI-B4
Robustness Optimisation	Input Preprocessing	Filter or transform the input to remove adversarial perturbation.	Weak defence. The transformations can be modelled and bypassed with EOT.	May be applicable, but the transformations may remove malicious traits.	VI-C1
	Adversarial Training	Trains the model with a correctly labelled adversarial example.	Strong defence. Adversarial training finetunes the decision boundary of the classifier.	Not applicable. Adversarial examples are not available at training.	VI-C2
	Post-Training	Modifies the structure of the model after training.	Weak defence. Multiple stronger attacks can bypass post-training defences.	Applicable but weak defence.	VI-C3

it against Kitsune [47]. Traffic Manipulator formulates the attack as a bi-level optimisation problem. The first level of optimisation is to find adversarial features closest to the original feature via GAN (similar to generative feature-level attacks described in Section V-C). The second level of optimisation finds a set of packet manipulation similar to the ones defined by Hashemi et al. [107] that can be applied to the original packets, which minimises the Euclidean distance between the actual feature and adversarial feature.

Sharon et al. [112] propose TANTRA, which reshapes the traffic by adjusting the arrival time of the original network attack. The attack first utilises an LSTM to learn and predict the benign traffic's inter-arrival times. Then, the LSTM reshapes the attack traffic to have the same distribution as the learned benign traffic. Although TANTRA has a very high evasion rate, modifying the time delays may reduce the malicious behaviour of the attack. However, the authors did not report on the maliciousness of the adversarial traffic.

Discussion: Packet-level attacks mark a significant improvement in practical adversarial attacks evading NIDS, as they can generate replayable packets in the network. However, existing adversarial attacks lack a comprehensive evaluation of the maliciousness of the adversarial examples. Current evaluations on adversarial attacks are mostly focused on evaluating their evasiveness (*i.e.,* measuring the accuracy or FNR of the NIDS on adversarial traffic), with very little evaluation of its maliciousness (*i.e.,* how much of the original malicious behaviour is preserved). The attacker's ultimate goal is to attack a device without being detected. Thus evasiveness and maliciousness of the adversarial packets are equally important. Future research should comprehensively evaluate the adversarial packets' maliciousness and evasiveness rather than solely focusing on their evasiveness. Moreover, it is known that the packet mutations can produce side effect features [121] (*e.g.,* injecting new packets into the traffic or fragmenting packets can cause new features to be extracted), but existing packet-level adversarial attacks have largely ignored the impact of side effect features. In Section VII-B3, we provide some recommendations on evaluation.

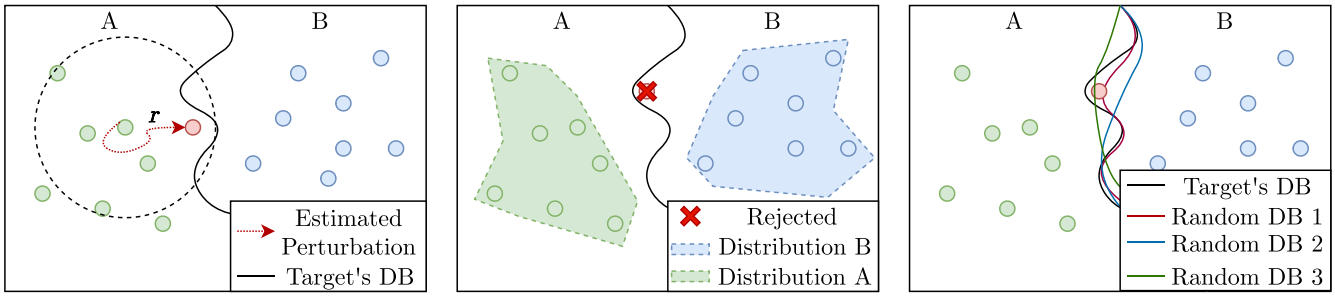
VI. ADVERSARIAL DEFENCES AND ITS APPLICABILITY IN NIDS

Due to the abundance of adversarial attacks and the potential damage they cause, researchers have developed various defensive mechanisms to detect or mitigate adversarial examples. However, most of the adversarial defence is evaluated with respect to CV, and adversarial defence methods intended for NIDS are largely unexplored.

This section reviews existing defences against adversarial attacks in general and categorises existing adversarial defences into three categories: parameter protection, adversarial detection, and robustness optimisation. For each category, we discuss its theoretical applicability in the NIDS domain. An overview of the defences described in this section is shown in Table IV. The defence mechanisms are illustrated in Figures 5 and 6.

A. Parameters Protection

Parameters protection is a class of defence mechanisms that intends to hide the neural network's internal weights, parameters and training data from the outside world. Parameter protection is commonly deployed to defend against model inversion attacks [122] that learn sensitive information about an individual in the training dataset. In defence against adversarial examples, parameter protection takes the form of

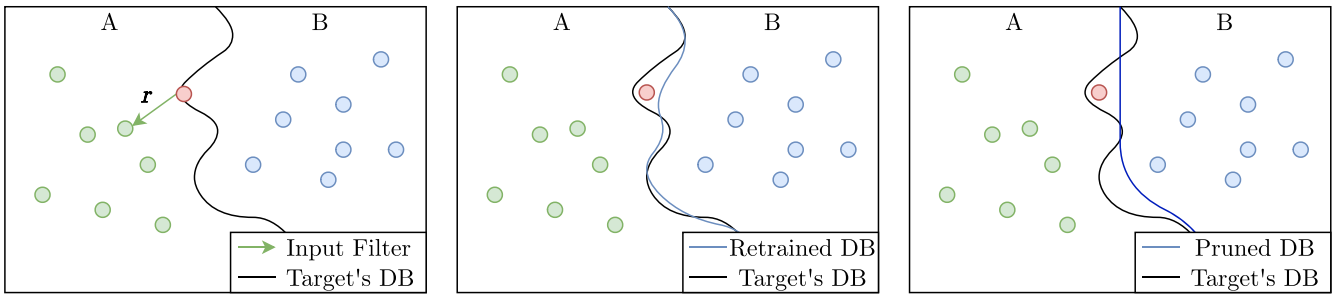


(a) Gradient masking: The gradient is protected from the attack so that the adversarial example is perturbated in the wrong direction and the resulting example (red) is not adversarial.

(b) Distributional adversarial detection: The adversarial example have inherent characteristics compared to clean examples (blue and green) and can be detected and rejected.

(c) Mutation-based adversarial detection: The model's decision boundary is randomly altered, and the output label of adversarial examples will be sensitive to this change.

Fig. 5. Gradient masking and adversarial detection defences: The DL model under protection has learned the decision boundary between two classes, A (green) and B (blue) and the defence mechanism tries to defend against an adversarial example (red) that is similar to A but classified as B.



(a) Input pre-processing: The adversarial example is projected back to the original class (green) by eliminating the adversarial perturbations.

(b) Adversarial training: The model is trained with a correctly labelled adversarial example so that the new decision boundary (blue) classifies it correctly.

(c) Post-training modifications: The original decision boundary is smoothed (blue) so that the model can generalise to adversarial examples.

Fig. 6. Robustness optimisation defences: input pre-processing, adversarial training, and post-training modification. The DL model under protection has learned the Decision Boundary (DB) between two classes, A (green) and B (blue). The defence mechanism tries to defend against the adversarial example (red) similar to A but classified as B.

gradient masking. The core idea is to prevent the actual gradient from being exposed by the network so that WB attacks cannot obtain a meaningful gradient.[3] Gradient masking can be accomplished by Shattered Gradients, Stochastic Gradients, and Vanishing & Exploding Gradients [123]. Fig. 5(a) illustrates the effect of gradient masking on adversarial attacks.

1) Shattered Gradients: Shattered gradients mask the gradient by making it non-differentiable, numerically unstable, non-existent, or incorrect. Defences that cause shattered gradients include Thermometer Encoding [124], Input Transformation [125] and Local Intrinsic Dimensionality [126].

Counterattacks: Shattered gradients can be bypassed with Backward Pass Differentiable Approximation (BPDA) [123]. BPDA approximates the non-differentiable layer in the neural network with a differentiable substitute function. Next, BPDA generates adversarial examples with WB methods (Section IV-B) using the approximate gradient from the substitute function.

2) Stochastic Gradients: Stochastic gradients hide the gradient by introducing randomness in the network or the input sample. As a result, the gradient is randomised, and the attacker cannot obtain the true gradient easily. Defences that utilise stochastic gradients are Stochastic Activation Pruning [127], and Input Randomisation [128].

Counterattacks: Expectation Over Transformation (EOT) [129] can bypass stochastic gradient-based defences under the WB setting. Rather than optimising on a single input example in a typical attack formulation, EOT introduces a set of transformations T and optimises the expectation over T of the input example. In essence, EOT finds the adversarial example that remains adversarial under all transformations in T.

3) Vanishing & Exploding Gradients: Vanishing & exploding gradients are caused by defences that utilise another deep generative model to project potentially adversarial examples back onto the data manifold, which "purifies" the adversarial example. The deep generative models cause the gradient to vanish to 0 or explode towards infinity. Defences that cause vanishing & exploding gradients include Pixel Defend [130] and Defense-GAN [131].

Counterattacks: Vanishing & exploding gradients can be bypassed with reparametrisation [123]. Consider a classifier $F(g(x))$ where $g(\cdot)$ is the projection function of the deep generative model. Although differentiating through $g(\cdot)$ is achievable, it results in vanishing or exploding gradients. The reparametrisation finds a differentiable function $h(\cdot)$ such

[3]Some defences cause gradient masking unintentionally, thus defences reviewed in this section may also appear in other categories.

that $g(h(z)) \approx h(z)$ for all z. This allows the gradient to be calculated via $F(h(z))$.

Discussion: Overall, gradient masking alone does not provide a complete defence against adversarial examples [94], [132]. All three types of gradient masking can be bypassed if the attacker knows the defence by approximating the masked gradients with a substitute gradient using BPDA [123], EOT [129], or reparameterisation [123]. Although not exact, the approximated gradient still provides helpful directions for constructing adversarial samples. Furthermore, gradient masking is even less effective against attacks generated under the BB threat model since the gradient is unknown in the first place [94].

Parameter protection is also not a viable option to increase the robustness of NIDS against adversarial attacks. The packet-level adversarial attacks we surveyed used a BB/GB threat model that does not rely on the classifier's gradient. This is because the gradient of the classifier's output with respect to input packets is computationally hard to find due to the feature extraction being a one-way function [70]. Thus, packet-level attacks treat the NIDS as a black box, and gradient masking is ineffective.

B. Adversarial Detection

Adversarial detection aims to detect adversarial examples before the model classifies them. The design philosophy behind adversarial detection is that adversarial examples are generated by synthetically altering the input, making them possess unique characteristics that differ from clean examples. Common adversarial detection methods include finding distributional differences between clean and adversarial examples (Fig. 5(b)), such as using a secondary classifier, projection-based detection, and statistics-based detection. Alternatively, adversarial detection can be made by stochastically modifying the decision function of the model (Fig. 5(c)) with mutation-based detection.

Carlini and Wagner [133] have evaluated adversarial detection with C & W attack [91] under the following three settings:

- *Zero-Knowledge Attacks* are those where the attacker knows nothing about the defence or whether there is a defence in place. This attack is made with a strong attack (C & W attack).
- *Perfect-Knowledge Attacks* are those where the attacker has perfect knowledge of the defence. To conduct an attack under this setting, a new attack loss function is constructed that evades both the classifier and adversarial detector.
- *Limited-Knowledge Attacks* are those where the attacker knows the type of the defence detector but not the detector's internal parameters. To conduct attacks under this setting, WB attacks are conducted to a substitute model, similar to transfer-based attacks described in Section IV-C.

1) Secondary Classifier: As the name suggests, this type of adversarial detection involves training a secondary classifier to classify adversarial examples. Grosse et al. [134] propose a new classifier model with an additional category specifically for the adversarial examples and train the model with adversarial samples generated with various adversarial attacks such as FGSM [38] and JSMA [89]. Gong et al. [135] proposed a similar defence where a secondary binary classifier is trained to distinguish between the adversarial and clean examples. Besides classifying the input features, Metzen et al. [136] detect adversarial examples based on intermediate feature representations. The inputs to convolutional layers are fed to a binary classifier that determines whether the input is adversarial or normal.

Counterattacks: Intuitively, adding another classifier only imposes more constraints on the adversarial attacks, but it does not remove the discontinuities in the feature manifolds learned by the classifier. Therefore, defences with a secondary classifier can be bypassed under perfect-knowledge and limited-knowledge settings by introducing additional terms in the attack's optimisation function that simultaneously bypasses the primary classifier and the adversarial detector [133].

2) Projection-Based Detection: Projection-based detection transforms the input feature space into a low-dimensional space for detection. PCA is the most frequently used method due to its linear nature. Hendrycks and Gimpel [137] found that adversarially perturbed images have abnormally high coefficients on larger principal components compared to natural images. Thus, classification can be made by analysing the coefficients. Bhagoji et al. [138] use PCA to reduce the dimensionality of the input sample and train a classifier on the smaller input, which forces the attacker to modify only the first few principal components. As a result of dimensionality reduction, the authors have used an FC neural network for classification since PCA removes spatial locality. Li and Li [139] propose a cascading classifier that applies PCA to the output of each convolutional filter and assigns a binary classifier to each layer. The sample is accepted as clean if all binary classifiers accept the input.

Counterattacks: The characteristics of adversarial examples after PCA projections are unique to specific datasets and are not generalisable [133] (*i.e.,* defences evaluated on MNIST [140] are not transferable to ImageNet [141]). Moreover, these defences were evaluated with weak FGSM that introduce relatively large perturbations and have failed to detect more sophisticated C & W attacks even under zero-knowledge settings.

3) Statistics-Based Detection: Statistics-based detection aims to detect adversarial examples via statistical methods. The motivation behind using statistical methods is that adversarial and clean examples belong to different distributions due to the adversarial perturbations. Grosse et al. [134] use the Maximum Mean Discrepancy (MMD) test, a type of hypothesis test that determines whether two sets of data are drawn from the same underlying distribution. The authors used Fisher's permutation test with MMD test statistics. First, the MMD test statistic is calculated between two sets of input. Next, the two sets are shuffled into new sets, and the MMD test statistic is calculated again for the new sets. If the two test statistics are different, then the two sets are drawn from different distributions. This process is repeated several times to increase confidence.

Feinman et al. [142] model the output of the final hidden layer as a Gaussian Mixture Model and applies Kernel Density

Estimation (KDE) to the final layer of the neural network to check whether the input comes from the same distribution as the clean example. The adversarial detector sets a threshold value during the training, and any input with KDE greater than the threshold is classified as adversarial.

Counterattacks: Statistics-Based Detection can be thought of as a combination of Secondary and Projection-Based Detection. The features are first projected to statistical distributions, followed by the classification of distributions with statistical tests. Therefore, Statistics-Based Detection inherits the weaknesses of both Secondary and Projection-Based Detection. Namely, the statistical properties are not generalisable to different datasets. The defences are evaluated against weak FGSM attacks, and they cannot defend against C & W attacks under a zero-knowledge setting. The classification of statistical tests can be bypassed by adding another term in the attack formulation to minimise the output of statistical tests [91].

4) Mutation-Based Detection: Mutation-based detection randomly alters the decision boundary of the neural network and measures the input example's sensitivity to the mutations. Adversarial examples often lie close to the decision boundary of the neural network to satisfy the similarity constraint (Equation 2). Thus, randomly altering the decision boundary causes adversarial examples to be classified differently while clean samples have constant output. Feinman et al. [142] propose measuring the neural network's uncertainty on a particular input, called Bayesian Neural Network Uncertainty (BNNU). Specifically, the network is randomised by applying dropout, and the input image is fed through the network several times. Adversarial examples that lie close to the decision boundary will be more sensitive to dropout and produce inconsistent labels.

Counterattacks: Mutation-based detection is the most effective type of adversarial defence. Studies have found that bypassing BNNU [142] requires the generated adversarial examples to be transferable to multiple networks rather than a single target network, making the task five times more challenging for the CIFAR dataset [133] with the C & W attack [143].

Improving Mutation-based Defence: The success of Mutation-Based Detection has inspired several defence techniques that successfully defend against C & W attacks. Xu et al. [144] proposed a defence technique called feature squeezing. They have found that the input feature space is often unnecessarily large, which provides an opportunity for adversarial evasion attacks. Feature squeezing reduces the input feature space and limits the degree of perturbations available to the attacker. The general strategy is to compare the model's prediction between original and squeezed inputs. If there is a significant difference, the sample is likely adversarial. The original sample can be compared with multiple squeezing operations in a cascading manner for more accurate detection. Xu et al. evaluated two ways to squeeze the features for image data: reducing the colour depth and spatial smoothing using filters. The authors showed its effectiveness in defending against C & W attacks [91] under a limited-knowledge setting where the attacker does not know the exact squeezing methods used.

Wang et al. [36] utilise the idea of mutation testing in the software engineering domain. They propose to detect adversarial examples at runtime by measuring the sensitivity of the DNN using mutation testing. During runtime, the neural network's structure is randomly mutated. The sensitivity of an input example is measured based on the label change rate. The authors show that this method can detect C & W attacks [91] under the zero-knowledge setting.

Discussion: Secondary Classifiers, Projection-Based Detection, and Statistics-Based Detection all rely on a common assumption: adversarial examples are synthetically modified and cannot occur naturally in the input space (*e.g.,* the distribution of pixels in an adversarial image cannot occur naturally). Therefore, the adversarial examples exhibit specific distributional characteristics that clean examples do not possess. However, studies have found that these characteristics are distinct to specific datasets and are not universal [133]. Furthermore, practical adversarial attacks against NIDS are created by altering the packets directly to ensure they can occur naturally in the input space. Hence, they are unlikely to exhibit distributional differences compared to clean examples.

Mutation-Based Detection exploits the fact that the adversarial examples have to lie close to the decision boundary imposed by the adversarial attack formulation, which is the most promising direction for adversarial detection. Recently, researchers have started applying Mutation-Based Detection to NIDS, such as MANDA [145]. MANDA utilises manifold learning to learn the manifolds of benign and malicious data. An example is likely to be adversarial if a mismatch exists between the NIDS's decision and manifold classification. However, the adversarial attacks launched against MANDA are feature-level attacks with domain constraints rather than practical packet-level attacks. Moreover, the threat model of MANDA assumes the defender has access to malicious network data, which may not be realistic.

Adversarial detection based defence is a simple and cost-effective method to defend against adversarial attacks. However, researchers must be careful when applying them to the NIDS domain. In Section III, we have discussed that NIDSes have much more rigorous constraints on its processing time to ensure user experience and that even a slight increase in processing time can accumulate and significantly worsen the user experience. However, adversarial detection inevitably increases the processing time since an extra classifier has to classify the packets. Therefore, future studies in adversarial detection based defences for NIDS must carefully evaluate computational overheads, for which we provide guidelines in Section VII-C3.

C. Robustness Optimisation

Robustness optimisation aims to make the neural network inherently robust to adversarial examples so that the adversarial examples are classified correctly. The notion of adversarial robustness was first introduced by Madry et al. [146], where the robustness of a classifier is measured by the minimum amount of perturbation needed for any adversarial attack to succeed. In other words, all

regions within a certain distance around an input example are classified correctly.

This section presents existing methods to increase the robustness of models, namely, input pre-processing, adversarial training, and post-training modifications. Fig. 6 illustrates the three robustness optimisation techniques.

1) Input Pre-Processing: Input pre-processing aims to remove the adversarial perturbations in the sample, as shown in Fig. 6(a). Intuitively, the pre-processing removes adversarial perturbation around the original input and pushes the adversarial input back to the original input. Input pre-processing methods can be applied in a plug-and-play manner and require no modification to the existing model.

One of the easiest ways to remove adversarial perturbation is to apply an average filter of size three by three to the input [139]. Although overly simplistic, results show that the accuracy of the classifier went from almost 0 to 73%. The average filter effectively adds a three-by-three convolutional filter at the beginning of the model and can be easily bypassed by taking the filter into account [133].

A more complicated pre-processing defence mechanism is Mag-Net [147]. Mag-Net utilises a detector and reformer to mitigate the effect of adversarial perturbations. The detector is an Autoencoder trained on clean data that minimises the reconstruction error of clean data. During execution, an adversarially perturbed input will have a high reconstruction error since the autoencoder is not optimised to reconstruct adversarial input. If the detector classified an input example as adversarial, it is rejected. Next, examples that are not rejected by the detector will be given to the reformer, which is another autoencoder that reconstructs the input to remove any potentially adversarial perturbations. Finally, classification is performed on the reconstructed input. Mag-Net is effective in detecting C & W attacks under limited-knowledge attacks.

Discussion: Input pre-processing methods are also domain agnostic and can be applied to the NIDS domain. However, some pre-processing methods are not sensible for network traffic data. For example, while applying rotation to image data makes sense, it does not apply to network data. Moreover, the structured nature of NIDS features could cause an initially malicious input to exhibit benign characteristics after filtering or vice versa and decrease the NIDS accuracy. Another concern for input pre-processing methods is the tradeoff between performance and computational complexity. While Mag-Net [147] can effectively detect strong adversarial attacks, it utilises an autoencoder to pre-process the input features, which could significantly increase detection time. On the other hand, applying a simple three-by-three average filter is computationally cheaper, but it can only defend against weak adversarial attacks.

2) Adversarial Training: Another way of defending against adversarial examples is to enhance the decision function's robustness during the training stage so that adversarial examples are classified correctly, as shown in Fig. 6(b). Adversarial examples lie in the training data's low probability regions and fool the classifier due to covariate shift [130]. Therefore, a simple, intuitive, and effective way of increasing the robustness of neural networks is adversarial training [38], [90], [146], where the neural network is trained with correctly labelled adversarial examples alongside clean examples. Intuitively, adversarial training trains the model with low probability regions of the decision function, leading to a more robust model. The success of adversarial training depends mainly on the quality of the generated adversarial examples, and multiple adversarial training frameworks have been proposed in CV:

1) *Projected Gradient Descent Adversarial Training [146]* finds adversarial examples via projected gradient descent and trains the model with the adversarial examples.
2) *Ensemble Adversarial Training [132]* generates adversarial examples from multiple pre-trained models and adversarial attacks. The target model is trained with the generated adversarial examples.
3) *Adversarial Logit Pairing [148]* is another technique where an additional regularisation function is added to the model's loss function to minimise the distance between the logits from two clean examples or between an adversarial example and a clean example.
4) *Misclassification Aware adveRsarial Training (MART) [149]* emphasises adversarial examples that are generated based on misclassified examples rather than correctly classified examples.
5) *Robust Local Features for Adversarial Training [150]* recognises the neural network's bias towards local features and transfers robust local features to the training of adversarial examples.
6) *TRadeoff-inspired Adversarial DEfence via Surrogate-loss Minimisation (TRADES) [151]* considers the region around the input and adversarially trains the target model's decision boundary to include the regions around the training examples. TRADES increases the robustness of the model at the cost of potentially losing some accuracy.

Discussion: Adversarial training is the most effective adversarial defence method in CV. When applied to the NIDS domain, the benefit of adversarial training is that it does not incur additional processing overheads during execution, which is unlike adversarial detection and input pre-processing defences. However, applying NIDS adversarial training in practice poses a significant challenge: it is difficult for the defender to obtain the network attack traffic and generate adversarial attacks against NIDS. Nevertheless, by relaxing the threat model of the defender to have access to malicious data, adversarial training has been successfully applied. The adversarial training examples are generated with existing feature-level attacks [152], [153] or with generative feature-level attacks [154], [155], [156] discussed in Sections V-B and V-C. Feature-level attacks are not practical, and future adversarial training defences should consider training the NIDS with packet-level attacks.

3) Post-Training Modification: Post-training modifications increase the classifier's robustness in a post-hoc manner after the model has been trained. The overarching goal of post-training modification is to smooth the model to generalise better for data outside of the training dataset, shown in Fig. 6(c).

Papernot et al. [117] propose defensive distillation to make trained neural networks more robust. Defensive distillation first

extracts the trained network's probability vectors and uses the probability vectors as the label to train another network with the same structure. The additional information from the probability vectors captures structural similarities between classes, preventing the overfitting of the model and increasing the robustness of the model against adversarial examples.

Stochastic Activation Pruning (SAP) [127] is another post-hoc defence against adversarial samples. In the forward-pass of the neural network, activations in each layer are randomly dropped inversely proportional to their magnitude so that large activations are more likely to be retained. The activations in subsequent layers are also scaled up to ensure the dynamic range of inputs. Intuitively, pruning small activations reduces the probability of accumulating adversarial perturbations in later layers, making the model more robust to adversarial examples.

Counterattacks: Defensive distillation was evaluated against relatively weak attacks, and it was shown to be ineffective against the C & W attack [91] and the Hop-Skip-Jump attack [95]. SAP can be bypassed via EOT [129] under perfect-knowledge settings [123].

Discussion: To the best of our knowledge, Post-training Modification has not yet been applied to NIDS adversarial defences, possibly because it is not an effective defence against CV models. Post-training Modification assumes the magnitude of feature perturbation is small and can be reduced to have no effect. However, practical adversarial attacks in NIDS only restrict the perturbation at the packet level rather than the feature level. Thus, the magnitude of perturbation at the feature level can be large and irreducible.

VII. Lessons Learned and Future Research Directions

This section discusses the lessons learned from existing research on NIDS and adversarial learning in NIDS and elaborates further on open problems present in designing practical adversarial attacks and adversarial defences in the NIDS domain, which we briefly highlighted in Sections V and VI. We take inspiration from other domains such as CV and NLP and provide potential future trends and recommendations. Fig. 7 shows the outline of this section.

A. Open Problems Facing NIDS

The unique nature of network traffic data and the taxonomy of NIDS have introduced many problems in the design and evaluation of NIDS. Consequently, evaluating adversarial attacks and defences on poorly evaluated NIDS can result in a false sense of strength. In the following, we identify common challenges facing the design and evaluation of NIDS, affecting the evaluation of adversarial attacks/defences.

1) Lack of Benchmark Dataset: Benchmark datasets are essential for evaluating any proposed NIDS and the respective adversarial attacks and defences. The true strengths and weaknesses of adversarial attacks and defences cannot be assessed without proper evaluation of the NIDS. However, existing benchmark datasets for NIDS face several unique challenges. Firstly, benchmark datasets rarely reflect the real distribution of network traffic due to privacy concerns. Real network traffic data contains sensitive information that could lead to a potential data breach, and it is hard for labs/organisations to release it publicly. Censoring/masking the payload is not a viable solution as it alters the distribution of the packet's payload, making it unrealistic. Similarly, datasets gathered in a simulated network environment cannot capture the diversity of realistic network traffic.

Second, network topologies and protocols are constantly evolving and growing. Consequently, network traffic characteristics and patterns are constantly changing as well. Therefore, benchmark datasets in NIDS are only valid for a short period. Using outdated datasets (*e.g.,* NSL-KDD [56] that is used by multiple recently proposed NIDS shown in Table II) may not capture representative network patterns for modern networks. Moreover, IoT networks exhibit drastically different levels of traffic diversity compared to large enterprise networks. Thus, NIDS evaluated with enterprise networks (*e.g.,* NSL-KDD [56], CIC-IDS-2017, and CSE-CIC-IDS2018 [63]) are unsuitable for IoT. However, our survey has found that some NIDS designed for IoT networks are evaluated with enterprise networks [39], [48], [49], [53], and urges researchers to evaluate the proposed system on suitable datasets.

Third, collecting network traffic data is a costly and labour-intensive process. The private and ever-changing nature of network data makes it challenging for researchers to gather network data by themselves. However, collecting realistic network data incurs a high economic and labour cost. One would have to set up a network with multiple devices and simulate benign network activity over time. Due to the high cost of setting up an enterprise network, self-gathered network data is commonly captured in IoT environments, where the network size is relatively small and individual devices are cheap [47], [157].

Future Directions: The main reason for the lack of benchmark datasets is the private and ever-changing nature of network traffic data, which is inevitable. Therefore, a potential future research direction is to apply Transfer Learning (TL) techniques [158] to reduce the cost of generating suitable network datasets for NIDS. TL aims to store the knowledge gained from solving a task and apply it to a different but related task. In the NIDS domain, researchers can leverage TL to learn general traffic patterns on general benchmark network datasets and utilise TL to apply learned patterns to specific network topologies.

2) Evaluation of Outlier Detection Is Challenging: ML algorithms are much better at finding similarities between examples (classification task) than identifying samples that do not belong (outlier detection) [13]. Therefore, training a NIDS with an outlier detection paradigm requires the training data to represent a near-perfect model of normality that is not improbable in the real world. Additionally, traffic classification directly outputs the specific type of attack, which is essential for a quick response to mitigate or counteract the attack. In contrast, outlier detection algorithms only report an attack without specifying the attack type. Due to better performance and output interpretability, most recently proposed DL-based NIDS systems are designed based on classification tasks rather than outlier detection. However, treating NIDS as a pure classification task has its limitations in practice: producing a correctly labelled

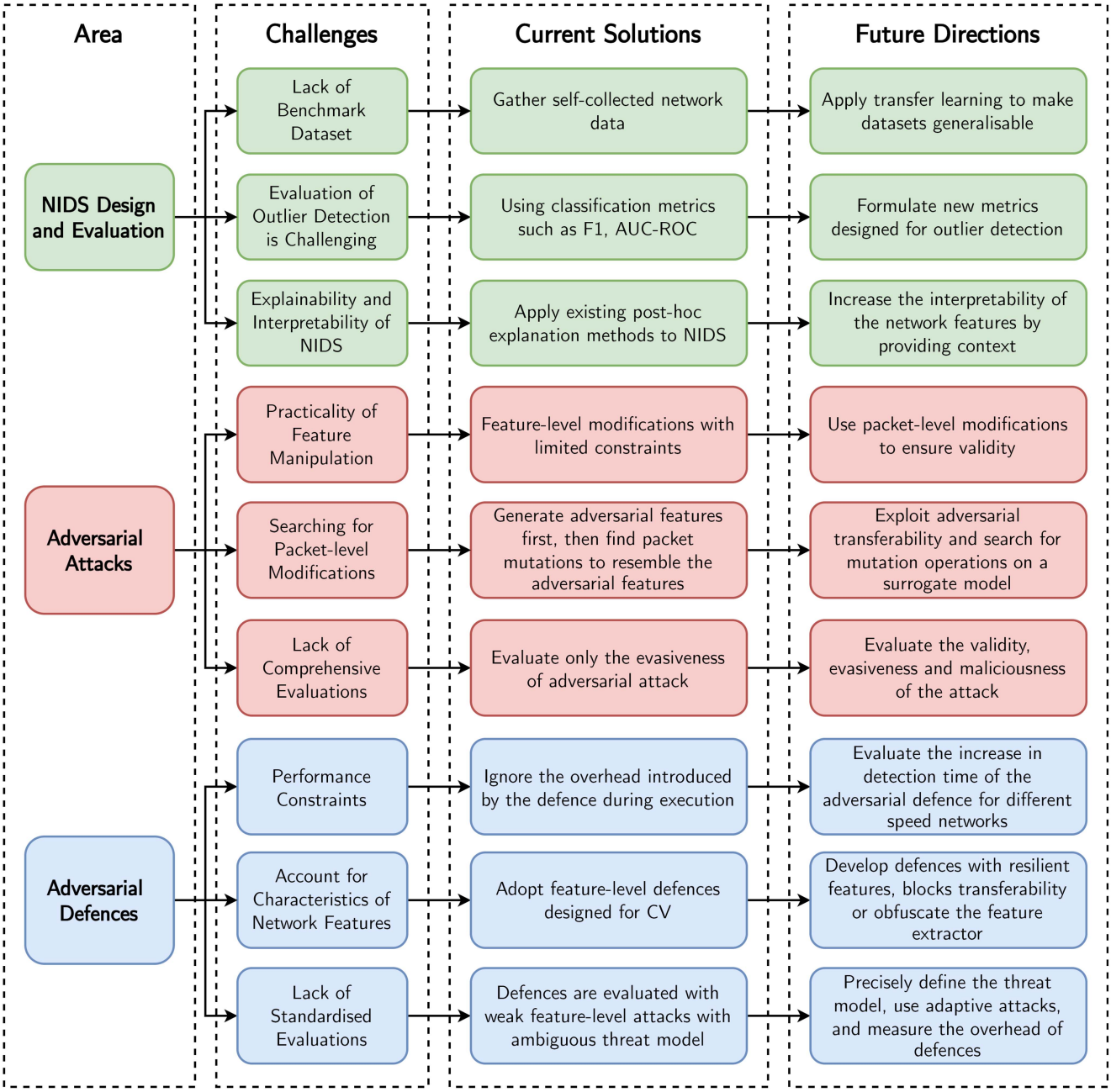


Fig. 7. Lessons learned and future directions of adversarial learning in NIDS.

dataset. Training an accurate classification model requires a balanced and labelled dataset, which is challenging to achieve for network data. Most network traffic will consist of benign traffic since it is unlikely that a network is constantly under network attack. Therefore, the dataset will be highly imbalanced and potentially results in biased decisions made by the trained model. Also, the sheer volume of network traffic (up to terabytes per day [23]) makes labelling all the traffic time-consuming. Further, it is also difficult to pinpoint packets/flows directly responsible for the traffic's malicious behaviour since malicious and benign traffic can have packets/flows in common (*e.g.,* TCP three-way handshake).

On the other hand, outlier detection algorithms do not have these limitations because training an outlier detection algorithm only requires benign traffic, which can be easily obtained by monitoring the network. Labelling is also unnecessary because there is only one class. Moreover, outlier detection algorithms in NIDS are more robust to zero-day attacks. Classification systems classify the input as one of the predefined classes based on the training dataset. However, NIDS is constantly faced with new, zero-day attacks that the system has not seen before. The classification-based NIDS performance in detecting zero-day attacks has not been comprehensively evaluated. Moreover, it is difficult to gather a comprehensive set of network attacks to train the classification model because it is unknown what type of attacks the network will be targeted at the training stage.

Future Directions: Due to the difficulties in obtaining a balanced, labelled dataset for traffic classification, we believe classification-based NIDS is more suitable for offline analyses of network attack characteristics in lab environments. On the

other hand, NIDS based on outlier detection algorithms is more suitable for practical, real-time detection of attacks. However, the majority of adversarial attacks and defences are evaluated against classification algorithms using metrics such as F1 and AUC-ROC. Classification metrics often require ground-truth labels, yet labelled data is rarely available in the NIDS domain. Simply adopting these metrics can provide a false sense of security. One future direction we recommend is to develop new metrics and evaluation frameworks for adversarial attacks and defences with outlier detection algorithms.

3) Explainability and Interpretability of NIDS: While adequately trained ML and DL algorithms provide high accuracy, their decision-making process is entirely unknown to researchers. Hence, the explainability and interpretability of ML and DL algorithms have triggered broad interest in the research community. Existing approaches to explainable ML involve using intrinsically explainable models or applying post-hoc explanations. Intrinsically explainable models, such as decision trees and linear regression, have simple decision boundaries that can be easily explained. On the other hand, post-hoc explanations, such as SHAP [159], aim to explain complex models such as DNNs via approximating the local region of the feature space. With the linear approximation, intrinsically explainable models can be applied to produce an explanation.

Explaining the NIDS decision in security-sensitive domains such as NIDS is extremely helpful in increasing the performance of the NIDS. During testing, researchers can utilise the explanations to gain insight into wrongly classified packets and improve the performance of the NIDS. During deployment, the explanations help the network operator quickly identify whether the packet is malicious or a false alarm. Moreover, the explanations can be applied to adversarial traffic to increase the robustness of the NIDS. The defender can utilise the explanations to investigate how the adversarial attack bypasses the NIDS to increase the robustness of the NIDS. On the other hand, the attacker can utilise the explanations to analyse how the adversarial defence defends against the attacker and create stronger attacks.

Existing works in explainable NIDS involve analysis of the gradient of the model with respect to the reconstruction error [54], applying existing post-hoc techniques to explain the NIDS [160], [161], or utilise adversarial attacks to empirically find the decision boundary of the NIDS [162]. However, these explanation techniques focus on explaining the NIDS' decision via the input features without considering whether the input features themselves are interpretable. In other words, the explanations only find the coefficients of a linear combination of features that produce NIDS' output and assume the variables are explainable. While this may be true for certain domains, it is certainly not the case for network traffic data. For example, a post-hoc explanation might indicate that a byte rate of 100 bytes contributes most towards classifying the traffic as malicious. However, the feature value of 100 bytes is merely just a number without any context. What is the practical significance of having a byte rate of 100 bytes? Is it too high or too low compared to benign traffic? Without answering these questions, an explanation of NIDS is incomplete.

Future Directions: Existing explanations of NIDS have strongly focused on explaining the significance of the features. However, they did not consider the explainability of the features. In the NIDS domain, even a seemingly obvious feature can be hard to interpret without context (*e.g.,* what is the practical significance of a forward byte rate of 100 bytes?), let alone much more complex features such as the covariance of packet sizes between two hosts. Existing explanation methods have provided solid theoretical foundations of the significance of the features, and we encourage future work to analyse and simplify the meaning behind the features.

B. Open Problems for Practical Adversarial Attacks in NIDS

In Section IV, we have introduced typical adversarial attack formulations that find singularities and blind spots within the high dimensional input feature space that causes the DL model to produce erroneous output. In the NIDS domain, launching practical adversarial attacks face three major problems: ensuring the practicality of feature manipulations, efficiently searching for packet-level modifications, and comprehensive evaluation of the attacks.

1) Practicality of Feature Manipulation: The majority of adversarial attacks we surveyed in Section V use feature-level perturbations to modify the highly correlated network features. Although some constraints have been placed on categorical features to ensure the features look realistic, the relationships between numerical features are unexplored. We have identified the following four characteristics of the network feature space that make practical feature-level attacks difficult.

- *Low Dimensional:* Network traffic data has extremely low dimensionality compared to image data. Even a small greyscale image of size 28×28 (MNIST [140]) contains 784 features, whereas a typical network feature extractor without payload inspection extracts around 80-100 features [47], [63]. The low dimensionality of network features naturally reduces the complexity of the data manifold and reduces the number of blind spots in the detection model, which makes it harder for the adversarial attack algorithm to find a solution.
- *Incomplete:* Network features are extracted from the same sequence of network traffic, therefore they are naturally related to each other and there will combinations of feature values that are impossible to create in practice. As a trivial example, if we extract the maximum packet size over different time windows, then the maximum packet size over the last 10 seconds cannot be higher than the maximum packet size over the last minute. The inherent structures in network traffic features indicate there are gaps within the network feature space that cannot be recreated with any real sequence of packets. Due to the complex relationships between network features, the gaps' exact location and size are unknown, making it difficult for the feature-level attacks to generate realistic features.
- *Unstable:* Network feature extractors often extract statistical information over a particular time window to reduce traffic diversity. As a result, traffic features are also sequentially related to the previous features. The

position of the gaps in the network feature space changes over time depending on previous features. The complex relationships between the network feature space make it difficult to recognise how the gaps move and introduce additional difficulty for feature-level adversarial attacks.

- *Irregular:* Network traffic features can contain numerous different types, such as categorical, continuous, and discrete features. For example, consider a sequence of TCP connections [163]. The possible values for the total number of ACK flags can be any positive integer, but valid values for mean traffic jitter can be any float value from 0.001 to 300 seconds. As a result, the feature-level attacks must adjust the step size for a particular feature depending on the possible values it may take. For complex network features, calculating the possible values of a feature is difficult.

Another challenge in feature-level attacks is producing the corresponding packets that generate the adversarial features. However, the feature extraction process is irreversible [70], and methods to reverse the features back to packets are unknown. Due to the complications in feature-level attacks, we encourage future research to investigate generating adversarial examples with packet-level attacks, as described in Section V-D.

2) Searching for Packet-Level Modifications: Packet-level attacks aim to find an optimal set of mutation operations to apply to the malicious packet that lowers the anomaly scores. Early attempts at finding the mutation operations involve vigorous trial and error [104], [107], which provides no theoretical guidance. Later, finding the mutation operation was improved to become a two-step process. The first step modifies the malicious features to adversarial features, followed by the second step that finds mutation operations to modify the malicious packet to resemble the adversarial feature [70], [109]. Finding the set of mutation operations is often formulated as a combinatorial optimisation problem, and the solution is found using heuristic search algorithms. There is, however, no guarantee that the optimal solution found using the heuristic search algorithm will be similar to the target adversarial features. In other words, the distance between the adversarial and actual features is only minimised but not bounded.

Future Directions: In Section VII-B1, we have discussed that network feature manipulations can be highly complex; therefore, we recommend future research to search for the packet-level modifications directly. For example, by exploiting the adversarial transferability [97] of DL algorithms, the attacker could apply a set of mutation operations that minimises the anomaly scores of the malicious packet on a surrogate NIDS and transfer it to the target NIDS.

3) Lack of Comprehensive Attack Evaluations: The attacker's goal when launching an adversarial attack against the network is to bypass the detection of NIDS and conduct the original intended attack. This means both the evasiveness and the maliciousness of the adversarial attack have to be evaluated. However, existing evaluations of adversarial attacks we surveyed in Section V focus on the evasiveness of the attacks rather than their maliciousness. Evaluating the maliciousness of adversarial attacks is problematic because it requires setting up a network and replaying the generated adversarial packets, which can be labour-consuming and costly. The lack of maliciousness evaluation reduces the practicality of adversarial attacks, and we recommend a few guidelines for thorough evaluations of adversarial attacks.

Future Directions: We recommend that future research focus on evaluating three main aspects of adversarial traffic: validity, evasiveness, and maliciousness. All three criteria can be evaluated by a single replay of adversarial traffic.

Validity refers to whether the adversarial traffic can be replayed correctly in the network. The modifications of the packets by the adversarial attack could introduce side effect features/packets [121] that can disrupt the intended flow of packets. Hence, the first step is to validate whether the generated attack can be replayed without disruption. Validity can be evaluated by checking any early termination of connections.

Evasiveness means whether the replayed traffic still evades the NIDS. When replaying the packets, inherent propagation and processing delays during transmission can slightly alter the adversarial packets' arrival time, leading to minute changes in extracted features. Furthermore, poorly crafted redundant packets may trigger additional packets from the victim, altering the traffic pattern and making the attack detectable. Thus, adversarial packets may evade the NIDS in theory but not in practice. Evasiveness can be evaluated via monitoring the NIDS and collecting existing metrics, such as false-positive-rate.

Maliciousness refers to whether the crafted traffic remains malicious. Although the attack traffic is modified with minimal change in its behaviour during the generation process, the exact effect of the modifications on maliciousness is unknown until replayed in the network. Maliciousness is the most demanding criterion to evaluate objectively because different attacks have different malicious intents and must be evaluated differently. Moreover, the definition of a successful attack largely depends on the context and can vary drastically. As a general rule of thumb, we suggest that researchers first define the minimum level of maliciousness that can be objectively measured for the attacks depending on the context. If the measurements of the adversarial attack are above the minimum threshold, it is considered successful. Consider a DoS attack on a smart music player in an IoT environment as a simple example. The normal response time for the smart music player is around 6 milliseconds, and a DoS attack causes the response time to reach 200 milliseconds. Depending on the context, users may find that a 50-millisecond delay is enough to make them feel annoyed and not use the device. Thus, the adversarial attack can be considered successful as long as it increases the response time above 50 milliseconds.

C. Open Problems for Adversarial Defences

The emergence of adversarial attacks has caused researchers to develop various defence mechanisms to secure DNNs from such attacks. Existing adversarial defences applied to NIDS are adopted from related works in CV, which have ignored some practical aspects of the NIDS threat model. We have identified three open problems in NIDS adversarial defences, including performance constraints, paradigm changes, and comprehensive defence evaluations.

1) Performance Constraints: The advancement of network technologies has drastically increased the speed of connections. As a result, even a small IoT network can generate large volumes of packets in a short time. Therefore, the NIDS must be able to classify thousands of packets every second to ensure the usability of the network. Similarly, the adversarial defences must not introduce significant overheads when classifying the packets. From this perspective, adversarial detectors and input pre-processing methods may not be suitable for adversarial defence as they inevitably increase the overall detection time to classify the input. A more reasonable approach is to use adversarial training, where the overhead is introduced in the training stage rather than the execution stage.

Besides ensuring the detection time, adversarial defences must ensure it reduces the False-Positive Rate (FPR) of NIDS on clean data. The large volume of network packets means that even a 0.1% decrease in FPR can raise thousands of false alarms daily. However, existing evaluations of adversarial defences have failed to evaluate processing overheads and the trade-off in the detection accuracy of clean examples. We encourage future works in NIDS adversarial defences to evaluate the cost of accuracy and overheads of NIDS in networks with different speeds to gain more insights into the trade-off between the performance and adversarial robustness of NIDS.

2) Account for Characteristics of Network Features: The fundamental design principle of adversarial detector defences relies on the fact that the adversarial example is generated with synthetic feature perturbation. Therefore, the adversarial examples lie extremely close to the decision boundary of the classifier compared to clean examples. Existing adversarial detector defences for NIDS, such as MANDA, [145] also rely on this and are evaluated against feature-level attacks. However, packet-level attacks with realistic packet-level modifications exhibit more natural perturbations in the feature space than feature-level attacks. Moreover, adversarial attacks against NIDS do not have the similarity constraint in feature space. Instead, the similarity constraint is placed on the semantics of the attack. As a result, packet-level adversarial attacks can introduce large perturbations on the features, violating the necessary assumption of adversarial detector defence. Moreover, existing adversarial defences aim to protect a classification algorithm with access to multiple classes during training. Nevertheless, practical NIDS systems are trained with outlier detection algorithms that only have access to benign data. The lack of attack data means adversarial training cannot be applied to NIDS. Defending NIDS against adversarial attacks requires a new approach, and we provide some possible recommendations below.

Future Directions: The packet-level attacks described in Section V-D rely on one of the three key operations: finding mutation operations that do not alter their behaviour, transferring the attack to the target model, and obtaining the extracted features. We encourage future adversarial defence designs to break these three key operations for the attacker. In the following, we provide some ideas for adversarial defence.

Resilient Features: An essential criterion for mutation operations is that they must change the packet's features without modifying its malicious behaviour. Therefore, the feature extractor can be designed to be mutation resistant. For example, we suggest including features directly correlated with common network attacks so that modifying a feature can reduce the maliciousness of attacks. This way, the adversarial traffic cannot be both evasive and malicious at the same time. One such example is to emphasise the packet inter-arrival time for DoS attacks so that any modification to the packet inter-arrival time can cause a change in its malicious behaviour (*e.g.,* DoS will become ineffective since the number of packets becomes manageable) so that the traffic cannot be both malicious and evasive. The main weakness of this defence is that finding these features requires expert knowledge and vigorous analysis of time-consuming attacks. Moreover, finding a comprehensive set of features that capture all network attacks' malicious behaviour is challenging.

Block Transferability: The generation of adversarial packets relies on adversarial transferability and finding a suitable substitute model. Therefore, blocking transferability will increase the difficulty of adversarial generation. Carlini and Wagner [133] found that the most promising way of blocking adversarial transferability is to introduce randomness in the network by applying dropout [164]. Defences in the NIDS domain can consider introducing randomness in the detection algorithm. For example, use a slightly different network architecture each time for detection and force the attacker to create universal adversarial examples transferable to a wide range of networks. Creating universal adversarial examples is a significantly more demanding task [97]. One possible direction for future work is applying Moving Target Defence (MTD) techniques [165] to periodically change the detection model. However, MTD could potentially introduce additional overhead in processing speed and accuracy that are not desirable.

Obfuscating Feature Extractor: Most attacks use a GB threat model, which assumes that the attacker knows the features being extracted. However, even under a BB threat model, the attacker can still take an educated guess of the extracted features since the number of possible network features extracted is limited [70]. In order to protect the feature extractor, the defender can consider randomising the feature extractor so that the NIDS is trained with random subsets of input features each time [166], which forces the adversarial attack to be universally adversarial against a wide range of feature extractors.

3) Lack of Standardised Defence Evaluations: From our review of adversarial attacks and defences in general, it is widely believed that there is no *universal* defence against all adversarial attacks [30], [31], which results in an arms race between the attackers and defenders. Hence, it is essential to comprehensively evaluate the defences to make sure it does increase the robustness of the NIDS rather than the attacks used being too weak. However, existing NIDS adversarial defences are evaluated against weak and impractical adversarial attacks, which provide a false sense of security.

Future Directions: In the following, we present key recommendation guidelines on the evaluation of NIDS adversarial defences.

- *Precisely define the threat model:* Prior works have used terms such as "white-box", "grey-box", and "black-box"

to describe slightly different threat models, introducing some ambiguity. This is especially true in the NIDS domain, where "grey-box" can have multiple meanings (*e.g.,* knowledge of the training data or knowledge of the feature extractor). Therefore, authors should be explicit about the threat model the attack/defence is evaluated to avoid any confusion.

- *Evaluate using a strong attack:* Defences should be evaluated against the strongest attacks known. However, there has yet been a comparison of the strength of packet-level attacks in the NIDS domain. Hence, defences for NIDS should be evaluated on a wide range of packet-level attacks to test their strength.
- *Evaluate against adaptive attacks:* Strong defences should be robust against future attacks as well as existing attacks. An adaptive attack is constructed after the defence is specified (*i.e.,* the attacker knows the defence in place). Therefore, they can manipulate existing attacks to find weaknesses in the defence. Future NIDS adversarial defences should also be evaluated against adaptive attacks.
- *Analyse overheads of defences:* NIDS solutions have significantly higher restrictions on response time and accuracy (Section VII-C1). However, existing evaluations of adversarial defences in NIDS have not considered the overhead of placing the defences during test time. We encourage future works to evaluate the overhead of the defence for networks of different sizes.

TABLE V
ACRONYMS USED THROUGHOUT OUR SURVEY

Acronym/Abbreviation	Stands for
AE	AutoEncoder
API	Application Programming Interface
BB	Black-Box
BNNU	Bayesian Neural Network Uncertainty
BPDA	Backward Pass Differentiable Approximation
CIC	Canadian Institute for Cybersecurity
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
CSE	Communications Security Establishment
CV	Computer Vision
CVAE	Conditional Variational AE
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DT	Decision Tree
EOT	Expectation Over Transformation
FA	Firefly Algorithm
FC	Fully Connected layer
FGSM	Fast Gradient Sign Method
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
GWO	Grey Wolf optimisation
HIDS	Host-based Intrusion Detection System
IDCT	Inverse Discrete Cosine Transform
IDS	Intrusion Detection System
IoT	Internet of Things
JDA	Jacobian-based Dataset Augmentation
JSMA	Jacobian Saliency Map Attack
KDE	Kernel Density Estimate
LSTM	Long Short-Term Memory
ML	Machine Learning
MTD	Moving Target Defence
MMD	Maximum Mean Discrepancy
NIDS	Network Intrusion Detection System
NLP	Natural Language Processing
PCA	Principal Component Analysis
PSO	Particle Swarm Optimisation
RNN	Recurrent Neural Network
SAP	Stochastic Activation Pruning
SDPN	Stacked Deep Polynomial Network
SMO	Spider Monkey Optimisation
SNDAE	Stacked Non-symmetrical Deep AE
SSAE	Stacked Sparse AE
SVM	Support Vector Machine
VAE	Variational AutoEncoder
WB	White-Box
WSN-DS	Wireless Sensor Networks Detection System
ZOO	Zeroth Order Optimisation

VIII. CONCLUSION

We have surveyed adversarial attacks and defences in the NIDS domain. Recent designs of NIDS have frequently used DL algorithms for the accurate detection of malicious activities. However, DL algorithms' inherent vulnerability to adversarial examples exposes a new attack surface for attackers to exploit. In this survey, we begin with a taxonomy of DL-based NIDS and analyse the effect of the NIDS taxonomy on adversarial learning. Next, we provide the general formulations of adversarial attacks to evade CV models and explicitly designed to evade NIDS. Then, we present existing defence mechanisms used to defend against adversarial attacks and discuss their applicability in the NIDS domain.

From our survey, we have found three main challenges facing the design of NIDS that indirectly affect the evaluation of adversarial attacks. First, there is a shortage of benchmark datasets to evaluate NIDS thoroughly. The private nature of network traffic data makes public releases of datasets difficult. Moreover, the fast-growing network technology causes previously released benchmark datasets to be outdated rather quickly. To overcome this challenge, we recommend incorporating TL techniques when training NIDS to learn general traffic patterns on benchmark datasets and apply the learned patterns to a more specific network. Second, the lack of labelled data available favours NIDS designed with an outlier detection paradigm. However, the metrics used in evaluating outlier detection based NIDS follow the metrics used in binary classification, which requires ground-truth labels. Therefore, we recommend that researchers design new metrics that can accurately reflect the performance of NIDS. Finally, the explainability of interpretability of DL-based NIDS remains an open problem. Although existing interpretability methods have successfully explained the output of NIDS based on the input feature attributions, the features themselves are poorly explained. Due to the high variability of network traffic, the meaning of features can be drastically different. For example, a forward byte rate of 100 is meaningless without knowing the benign forward byte rate. Hence, future work should aim to explain the network features themselves by providing contextual information.

In terms of adversarial attacks against NIDS, we have found that generating realistic network features is extremely difficult

due to the network feature space being low-dimensional, incomplete, unstable, and irregular. Moreover, modifying network features has no practical value because features cannot be transmitted across the network to conduct the attack. We recommend that future work should focus on packet-level attacks. However, the existing evaluations of packet-level attacks mainly focus on the attack's evasiveness and have largely ignored the maliciousness of the adversarial attack. For a more comprehensive evaluation, we have shared guidelines for evaluating adversarial examples, including their validity, evasiveness, and maliciousness.

For NIDS adversarial defences, we have found that existing adversarial defence methods are tailored for feature-level attacks and classification algorithms, which are unsuitable for packet-level attacks against outlier detection algorithms. Moreover, NIDS has much stricter processing and accuracy requirements, yet existing adversarial defences have failed to address both requirements. NIDS Adversarial defence requires new methodologies and evaluation frameworks. We recommend future research to apply MTD techniques that introduce uncertainty in various components of the NIDS, making it difficult for the attacker to compromise the system. In addition, we provide guidelines for future NIDS adversarial defence evaluation, such as precisely defining the threat model, evaluating with strong, adaptive attacks, and measuring the overhead of defences.

Overall, adversarial learning in the NIDS domain is an emerging research field. The unique characteristics of network traffic features and the NIDS detection pipeline have posed brand new and exciting challenges that need to be addressed.

APPENDIX

See Table V.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] N. Virvilis and D. Gritzalis, "The big four—What we did wrong in advanced persistent threat detection?" in *Proc. Int. Conf. Availability, Rel. Security*, 2013, pp. 248–254. [Online]. Available: https://doi.org/10.1109/ARES.2013.32

[3] "John the ripper password cracker." Openwall. 2019. Accessed: Feb. 11, 2022. [Online]. Available: https://www.openwall.com/john

[4] Anna-Senpai, "Mirai source code." 2017. Accessed: Feb. 11, 2022. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code

[5] Y. Said. "ARP spoofing using a man-in-the-middle attack." 2020. Accessed: Feb. 11, 2022. [Online]. Available: https://linuxhint.com/arp_spoofing_using_man_in_the_middle_attack

[6] "LOIC." NewEraCracker. 2019. Accessed: Feb. 11, 2022. [Online]. Available: https://github.com/NewEraCracker/LOIC

[7] U. H. Rao and U. Nayak, *Intrusion Detection and Prevention Systems*. Berkeley, CA, USA: Apress, 2014, pp. 225–243. [Online]. Available: https://doi.org/10.1007/978-1-4302-6383-8_11

[8] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Inf. Sci.*, vol. 239, pp. 201–225, Aug. 2013.

[9] "Snort." Accessed: Feb. 11, 2022. [Online]. Available: https://www.snort.org

[10] "Suricata." The Open Information Security Foundation. 2020. Accessed: Feb. 11, 2022. [Online]. Available: https://suricata-ids.org

[11] D. Mutz, C. Kruegel, W. Robertson, G. Vigna, and R. A. Kemmerer, "Reverse engineering of network signatures," in *Proc. Auscert Asia–Pacific Inf. Technol. Security Conf.*, 2005, pp. 1–12.

[12] "Zeek." 2020. Accessed: Feb. 11, 2022. [Online]. Available: https://zeek.org

[13] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. 31st IEEE Symp. Security Privacy*, 2010, pp. 305–316. [Online]. Available: https://doi.org/10.1109/SP.2010.25

[14] P. Fogla, M. I. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee, "Polymorphic blending attacks," in *Proc. 15th USENIX Security Symp.*, 2006, pp. 1–16. [Online]. Available: https://www.usenix.org/conference/15th-usenix-security-symposium/polymorphic-blending-attacks

[15] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Automatically evading IDS using GP authored attacks," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2007, pp. 153–160. [Online]. Available: https://doi.org/10.1109/CISDA.2007.368148

[16] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, 2021, Art. no. e4150.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] C. Szegedy et al., "Intriguing properties of neural networks," 2014, *arXiv:1312.6199*.

[19] O. Ibitoye, R. A. Khamis, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security—A survey," 2019, *arXiv:1911.02621*.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[21] A. Hannun et al., "Deep speech: Scaling up end-to-end speech recognition," 2014, *arXiv:1412.5567*.

[22] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 3rd Quart., 2020.

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

[24] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.

[25] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[26] G. Li, P. Zhu, J. Li, Z. Yang, N. Cao, and Z. Chen, "Security matters: A survey on adversarial machine learning," 2018, *arXiv:1810.07339*.

[27] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018, *arXiv:1810.00069*.

[28] X. Wang, J. Li, X. Kuang, Y.-A. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *J. Parallel Distrib. Comput.*, vol. 130, pp. 12–23, Aug. 2019.

[29] S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru, "A study of black box adversarial attacks in computer vision," 2019, *arXiv:1912.01667*.

[30] S. H. Silva and P. Najafirad, "Opportunities and challenges in deep learning adversarial robustness: A survey," 2020, *arXiv:2007.00753*.

[31] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–38, 2020.

[32] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, pp. 1–41, 2020.

[33] F. O. Olowononi, D. B. Rawat, and C. Liu, "Resilient machine learning for networked cyber physical systems: A survey for machine learning security to securing machine learning for CPS," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 524–552, 1st Quart., 2021. [Online]. Available: https://doi.org/10.1109/COMST.2020.3036778

[34] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35403–35419, 2020.

[35] H. A. Alatwi and C. Morisset, "Adversarial machine learning in network intrusion detection domain: A systematic review," 2021, *arXiv:2112.03315*.

[36] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *Proc. 41st Int. Conf. Softw. Eng.*, 2019, pp. 1245–1256. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00126

[37] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," *ACM Comput. Surveys*, vol. 54, no. 5, pp. 1–36, 2021. [Online]. Available: https://doi.org/10.1145/3453158

[38] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–11.

[39] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[40] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. K. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3854–3861. [Online]. Available: https://doi.org/10.1109/IJCNN.2017.7966342

[41] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[42] V. L. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf.* 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WCNC.2017.7925567

[43] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," *IET Inf. Security*, vol. 13, no. 1, pp. 48–53, 2018.

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

[45] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.

[46] S. Naseer et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[47] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.

[48] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[49] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, 2022, Art. no. e3803.

[50] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.

[51] X. Zhang, J. Chen, Y. Zhou, L. Han, and J. Lin, "A multiple-layer representation learning model for network-based attack detection," *IEEE Access*, vol. 7, pp. 91992–92008, 2019.

[52] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[53] M. Roopak, G. Y. Tian, and J. A. Chambers, "Deep learning models for cyber security in IoT networks," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf.*, 2019, pp. 452–457. [Online]. Available: https://doi.org/10.1109/CCWC.2019.8666588

[54] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "GEE: A gradient-based explainable variational autoencoder for network anomaly detection," in *Proc. 7th IEEE Conf. Commun. Netw. Security*, 2019, pp. 91–99. [Online]. Available: https://doi.org/10.1109/CNS.2019.8802833

[55] P. Sun et al., "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security Commun. Netw.*, vol. 2020, pp. 1–11, Aug. 2020. [Online]. Available: https://doi.org/10.1155/2020/8890306

[56] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CISDA.2009.5356528

[57] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exp. Returns Security*, 2011, pp. 29–36. [Online]. Available: https://doi.org/10.1145/1978672.1978676

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

[59] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Security*, vol. 45, pp. 100–123, Sep. 2014.

[60] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Military Commun. Inf. Syst. Conf.*, 2015, pp. 1–6. [Online]. Available: https://doi.org/10.1109/MilCIS.2015.7348942

[61] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.

[62] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Comput. Netw.*, vol. 127, pp. 200–216, Nov. 2017.

[63] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic Characterization," in *Proc. 4th Int. Conf. Inf. Syst. Security Privacy*, 2018, pp. 108–116. [Online]. Available: https://doi.org/10.5220/0006639801080116

[64] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st Quart., 2016.

[65] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[66] N. Moustafa, "New generations of Internet of Things datasets for cybersecurity applications based machine learning: TON_IoT datasets," in *Proc. eRes. Aust. Conf.*, 2019, pp. 1–2.

[67] M. Rigaki, *Adversarial Deep Learning Against Intrusion Detection Classifiers*, Luleå Univ. Technol., Luleå, Sweden, 2017.

[68] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[69] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *Proc. IEEE Mil. Commun. Conf.*, 2018, pp. 559–564. [Online]. Available: https://doi.org/10.1109/MILCOM.2018.8599759

[70] D. Han et al., "Practical traffic-space adversarial attacks on learning-based NIDSs," 2020, *arxiv:2005.07519v1*.

[71] A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, "Using feature selection for intrusion detection system," in *Proc. Int. Symp. Commun. Inf. Technol. (ISCIT)*, 2012, pp. 296–301.

[72] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.

[73] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.

[74] M. M. Sakr, M. A. Tawfeeq, and A. B. El-Sisi, "Filter versus wrapper feature selection for network intrusion detection system," in *Proc. 9th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, 2019, pp. 209–214.

[75] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Comput. Stat. Data Anal.*, vol. 143, Mar. 2020, Art. no. 106839.

[76] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Security*, vol. 81, pp. 148–155, Mar. 2019.

[77] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," *Symmetry*, vol. 12, no. 6, p. 1046, 2020.

[78] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Stat. Soc. B, Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[79] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Royal Stat. Soc. B, Stat. Methodol.*, vol. 67, no. 2, pp. 301–320, 2005.

[80] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253517305328

[81] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

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

[83] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[84] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2015, pp. 1412–1421. [Online]. Available: https://doi.org/10.18653/v1/d15-1166

[85] A. Vaswani et al., "Attention is all you need," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[86] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2013.

[87] S. Chan, "Lecture notes in adversarial attacks," Lecture Notes, School Electr. Comput. Eng., Purdue Univ., West Lafayette, IN, USA, 2020.

[88] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582. [Online]. Available: https://doi.org/10.1109/CVPR.2016.282

[89] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy*, 2016, pp. 372–387. [Online]. Available: https://doi.org/10.1109/EuroSP.2016.36

[90] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=BJm4T4Kgx

[91] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 39–57. [Online]. Available: https://doi.org/10.1109/SP.2017.49

[92] P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innovative Appl. Artif. Intell. (IAAI), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, 2018, pp. 10–17. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16893

[93] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quart. Appl. Math.*, vol. 2, no. 3, pp. 258–261, 1944.

[94] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2017, pp. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[95] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Security Privacy*, 2020, pp. 1277–1294. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00045

[96] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Security*, 2017, pp. 15–26. [Online]. Available: https://doi.org/10.1145/3128572.3140448

[97] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*.

[98] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2142–2151. [Online]. Available: http://proceedings.mlr.press/v80/ilyas18a.html

[99] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," 2016, *arXiv:1612.06299*.

[100] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018, *arXiv:1712.04248*.

[101] E. Aarts, E. Aarts, and J. Lenstra, *Local Search in Combinatorial Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2003. [Online]. Available: https://books.google.co.nz/books?id=MOk9DwAAQBAJ

[102] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2341–2368, 2013.

[103] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural evolution strategies," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3381–3387. [Online]. Available: https://doi.org/10.1109/CEC.2008.4631255

[104] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanácek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent Obfuscations: An adversarial approach," *EAI Endorsed Trans. Security Safety*, vol. 5, no. 17, p. e4, 2019. [Online]. Available: https://doi.org/10.4108/eai.10-1-2019.156245

[105] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," 2018, *arXiv:1809.02077*.

[106] J. Clements, Y. Yang, A. A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," 2019, *arXiv:1903.11688*.

[107] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *Proc. 3rd ACM CoNEXT Workshop Big DAta, Mach. Learn. Artif. Intell. Data Commun. Netw.*, 2019, pp. 14–21. [Online]. Available: https://doi.org/10.1145/3359992.3366642

[108] O. Ibitoye, M. O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks," in *Proc. IEEE Global Commun. Conf.*, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOBECOM38437.2019.9014337

[109] A. Kuppa, S. Grzonkowski, M. R. Asghar, and N. Le-Khac, "Black box attacks on deep anomaly detectors," in *Proc. 14th Int. Conf. Availability, Rel. Security*, 2019, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3339252.3339266

[110] E. Alhajjar, P. Maxwell, and N. Bastian, "Adversarial machine learning in network intrusion detection systems," *Expert Syst. Appl.*, vol. 186, Dec. 2021, Art. no. 115782.

[111] Q. Cheng, S. Zhou, Y. Shen, D. Kong, and C. Wu, "Packet-level adversarial network traffic crafting using sequence generative adversarial networks," 2021, *arXiv:2103.04794*.

[112] Y. Sharon, D. Berend, Y. Liu, A. Shabtai, and Y. Elovici, "Tantra: Timing-based adversarial network traffic reshaping attack," 2021, *arXiv:2103.06297*.

[113] R. Sheatsley, N. Papernot, M. J. Weisman, G. Verma, and P. McDaniel, "Adversarial examples for network intrusion detection systems," *J. Comput. Security*, vol. 30, no. 5, pp. 727–752, 2022.

[114] B.-E. Zolbayar et al., "Generating practical adversarial network traffic flows using NIDSGAN," 2022, *arXiv:2203.06694*.

[115] C.-H. Huang, T.-H. Lee, L.-h. Chang, J.-R. Lin, and G. Horng, "Adversarial attacks on SDN-based deep learning IDS system," in *Mobile and Wireless Technology*. Singapore: Springer, 2019, pp. 181–191.

[116] A. Warzynski and G. Kolaczek, "Intrusion detection systems vulnerability on adversarial examples," in *Proc. Innovat. Intell. Syst. Appl.*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/INISTA.2018.8466271

[117] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Security Privacy*, 2016, pp. 582–597. [Online]. Available: https://doi.org/10.1109/SP.2016.41

[118] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[119] B. Zong et al., "Deep Autoencoding gaussian mixture model for unsupervised anomaly detection," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[120] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-based anomaly detection," 2018, *arXiv:1802.06222*.

[121] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 1332–1349.

[122] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333. [Online]. Available: https://doi.org/10.1145/2810103.2813677

[123] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 274–283. [Online]. Available: http://proceedings.mlr.press/v80/athalye18a.html

[124] J. Buckman, A. Roy, C. Raffel, and I. J. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=S18Su--CW

[125] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=SyJ7ClWCb

[126] X. Ma et al., "Characterizing adversarial subspaces using local intrinsic dimensionality," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=B1gJ1L2aW

[127] G. S. Dhillon et al., "Stochastic activation pruning for robust adversarial defense," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–13. [Online]. Available: https://openreview.net/forum?id=H1uR4GZRZ

[128] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=Sk9yuql0Z

[129] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 284–293. [Online]. Available: http://proceedings.mlr.press/v80/athalye18b.html

[130] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "PixelDefend: Leveraging generative models to understand and defend against adversarial examples," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–20. [Online]. Available: https://openreview.net/forum?id=rJUYGxbCW

[131] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=BkJ3ibb0-

[132] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=rkZvSe-RZ

[133] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Security*, 2017, pp. 3–14. [Online]. Available: https://doi.org/10.1145/3128572.3140444

[134] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," 2017, *arXiv:1702.06280*.

[135] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," 2017, *arXiv:1704.04960*.

[136] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=SJzCSf9xg

[137] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–9. [Online]. Available: https://openreview.net/forum?id=B1dexpDug

[138] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," 2017, *arXiv:1704.02654*.

[139] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5775–5783. [Online]. Available: https://doi.org/10.1109/ICCV.2017.615

[140] Y. LeCun. "The MNIST database of handwritten digits." 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[141] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255. [Online]. Available: https://doi.org/10.1109/CVPR.2009.5206848

[142] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017, *arXiv:1703.00410*.

[143] N. Carlini and D. A. Wagner, "Defensive distillation is not robust to adversarial examples," 2016, *arXiv:1607.04311*.

[144] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. 25th Annu. Netw. Distrib. Syst. Security Symp.*, 2018, pp. 1–15. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-4_Xu_paper.pdf

[145] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "MANDA: On adversarial example detection for network intrusion detection system," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[146] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–23. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[147] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 135–147. [Online]. Available: https://doi.org/10.1145/3133956.3134057

[148] H. Kannan, A. Kurakin, and I. J. Goodfellow, "Adversarial logit pairing," 2018, *arXiv:1803.06373*.

[149] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=rklOg6EFwS

[150] C. Song, K. He, J. Lin, L. Wang, and J. E. Hopcroft, "Robust local features for improving the generalization of adversarial training," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=H1lZJpVFvr

[151] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482. [Online]. Available: http://proceedings.mlr.press/v97/zhang19p.html

[152] M. Pawlicki, M. Choraś, and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," *Future Gener. Comput. Syst.*, vol. 110, pp. 148–154, Sep. 2020.

[153] C. Zhang, X. Costa-Pérez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms," *IEEE/ACM Trans. Netw.*, vol. 30, no. 3, pp. 1294–1311, Jun. 2022.

[154] M. Usama, M. Asim, S. Latif, J. Qadir, and Ala-Al-Fuqaha, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2019, pp. 78–83.

[155] J. Wang, J. Pan, I. AlQerm, and Y. Liu, "Def-IDS: An ensemble defense mechanism against adversarial attacks for deep learning-based network intrusion detection," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–9.

[156] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa, "GaDoT: GAN-based adversarial training for robust DDoS attack detection," in *Proc. 9th IEEE Conf. Commun. Netw. Security (IEEE CNS)*, 2021, pp. 1–9.

[157] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "IoT network intrusion dataset." 2019. Accessed: Jul. 26, 2021. [Online]. Available: http://ocslab.hksecurity.net/Datasets/iot-network-intrusion-dataset

[158] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[159] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[160] K. Sauka, G.-Y. Shin, D.-W. Kim, and M.-M. Han, "Adversarial robust and explainable network intrusion detection systems based on deep learning," *Appl. Sci.*, vol. 12, no. 13, p. 6451, 2022.

[161] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020.

[162] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable AI in intrusion detection systems," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc.*, 2018, pp. 3237–3243.

[163] J. Postel et al., "Transmission control protocol," Inf. Sci. Inst., Univ. Southern California, Marina Del Rey, CA, USA, RFC 793, 1981.

[164] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[165] S. Sengupta, T. Chakraborti, and S. Kambhampati, "MTDeep: Boosting the security of deep neural nets against adversarial attacks with moving target defense," in *Proc. 10th Int. Conf. Decis. Game Theory Security*, 2019, pp. 479–491. [Online]. Available: https://doi.org/10.1007/978-3-030-32430-8_28

[166] R. Colbaugh and K. Glass, "Moving target defense for adaptive adversaries," in *Proc. IEEE Int. Conf. Intell. Security Inf.*, 2013, pp. 50–55. [Online]. Available: https://doi.org/10.1109/ISI.2013.6578785