symmetry

MDPI

Review

A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks

Abdussalam Ahmed Alashhab [1,2], Mohd Soperi Mohd Zahid [1,*], Mohamed A. Azim [3], Muhammad Yunis Daha [1], Babangida Isyaku [4] and Shimhaz Ali [5]

[1] Department of Computer and Information Science, Universiti Teknologi Petronas, Seri Iskandar 32610, Malaysia; abdussalaam91@gmail.com (A.A.A.); muhammad_19001726@utp.edu.my (M.Y.D.)
[2] Faculty of Information Technology, Alasmarya Islamic University, Zliten, Libya
[3] Department of Computer Science, University of Prince Mugrin, Medina 40202, Saudi Arabia; m.zayed@upm.edu.sa
[4] Department of Mathematics and Computer Science, Sule Lamido University, Kano 700271, Nigeria; babangida.isyaku@slu.edu.ng
[5] Faculty of Information Technology, University Sains Islam Malaysia, Nilai 71800, Malaysia; shimhaz@police.gov.mv
* Correspondence: msoperi.mzahid@utp.edu.my

Abstract: Software-defined networking (SDN) is a new networking paradigm that provides centralized control, programmability, and a global view of topology in the controller. SDN is becoming more popular due to its high audibility, which also raises security and privacy concerns. SDN must be outfitted with the best security scheme to counter the evolving security attacks. A Distributed Denial-of-Service (DDoS) attack is a network attack that floods network links with illegitimate data using high-rate packet transmission. Illegitimate data traffic can overload network links, causing legitimate data to be dropped and network services to be unavailable. Low-rate Distributed Denial-of-Service (LDDoS) is a recent evolution of DDoS attack that has been emerged as one of the most serious vulnerabilities for the Internet, cloud computing platforms, the Internet of Things (IoT), and large data centers. Moreover, LDDoS attacks are more challenging to detect because this attack sends a large amount of illegitimate data that are disguised as legitimate traffic. Thus, traditional security mechanisms such as symmetric/asymmetric detection schemes that have been proposed to protect SDN from DDoS attacks may not be suitable or inefficient for detecting LDDoS attacks. Therefore, more research studies are needed in this domain. There are several survey papers addressing the detection mechanisms of DDoS attacks in SDN, but these studies have focused mainly on high-rate DDoS attacks. Alternatively, in this paper, we present an extensive survey of different detection mechanisms proposed to protect the SDN from LDDoS attacks using machine learning approaches. Our survey describes vulnerability issues in all layers of the SDN architecture that LDDoS attacks can exploit. Current challenges and future directions are also discussed. The survey can be used by researchers to explore and develop innovative and efficient techniques to enhance SDN's protection against LDDoS attacks.

Keywords: network security; SDN; OpenFlow; DDoS attacks; low-rate DDoS attacks; machine learning; detection mechanisms

Citation: Alashhab, A.A.; Zahid, M.S.M.; Azim, M.A.; Daha, M.Y.; Isyaku, B.; Ali, S. A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks. *Symmetry* **2022**, *14*, 1563. https://doi.org/10.3390/sym14081563

Academic Editor: Yu-Chi Chen

Received: 29 May 2022
Accepted: 28 June 2022
Published: 29 July 2022

1. Introduction

Traditional IP networks are too complex and difficult to manage due to the vertical bonding between the control plane and the data plane. Network resources are underutilized given the complexity and unpredictability of traditional IP networks [1]. A new concept of software-defined networking (SDN) technology has been introduced to address this problem. SDN technology decouples the control plane from the data plane, allowing the programmability of IP networks [2–5]. The SDN architecture has been adopted by many

businesses and government sectors [6,7] for its appealing features, including network programmability, unified control, capabilities, and a global view of the network topology in the controller. However, similarly to traditional IP networks, SDN confronts various research challenges such as network failures, security, and privacy threats. As SDN is expected to be one of the dominant networking technologies in the near future, developing novel security mechanisms to tackle these security threats has become a necessity.

One of the most common security threats is the distributed denial-of-service (DDoS) attack that has plagued the Internet for more than 20 years, and it is becoming even more aggressive with the development of the Internet of Things (IoT) [8–10]. DDoS attacks have evolved, progressing from basic to high-rate to sensible low-rate traffic [11]. A new security threat known as low-rate DDoS (LDDoS) has emerged as a result of this timely transformation of DDoS attacks [12]. LDDoS attacks can take a variety of forms, such as denying service to servers by sending a very low-rate request to disrupt application services. Attackers may also use tools such as SlowDroid [13] to launch low-rate attacks on smartphones with limited processing power and exploit weaknesses in the application layer using protocols such as Slow Next [14].

LDDoS attacks pose a serious security threat to both traditional IP networks and SDN [15,16]. Detecting LDDoS attacks is more difficult because the generated traffic is disguised as legitimate traffic. The attack is launched from a separate source and the typical rate is low enough that the number of packets delivered is insignificant, making detection difficult. LDDoS attacks are undetectable using standard DDoS detection security measures and implementations [17]. It is envisaged that more efforts are needed to increase the SDN level of protection against LDDoS. A survey on existing detection mechanisms will be useful for researchers to be aware of current the state-of-the-art methods and propose more efficient detection mechanisms.

Many relevant surveys on security detection mechanisms for SDN can be found in the literature, but they are limited only to high-rate DDoS attacks and not LDDoS attacks. In this paper, we survey various research studies developed for LDDoS detection mechanisms for SDN using machine learning (ML) methods. The study is motivated by the fact that LDDoS is a novel and difficult-to-detect type of DDoS attacks, and ML methods are appealing and effective in detecting DDoS attacks. This paper provides a comprehensive overview of security solutions proposed for all layers of SDN against LDDoS using ML models. The various threats and effects of LDDoS attacks on SDN architecture layers are highlighted; a summary and comparison of LDDoS detection mechanisms for SDN based on ML and deep learning (DL) methods are provided, and open research problems and future directions for SDN security researchers working on LDDoS detection are discussed. Therefore, the contribution of this paper summarized in three points:

- To identify various types of threats and effects of LDDoS attacks on different layers of SDN architecture;
- To provide a summary and comparison of LDDoS detection mechanisms for SDN based on machine learning and deep learning approaches;
- To discuss the open research problems and future directions for researchers working in the domain of SDN security based on LDDoS attack detection.

The remaining sections of this paper are organized as follows: Section 2 describes the methodology used in our survey. Section 3 presents related work. Section 4 explains the basic concept of SDN, DDoS attack, and LDDoS attack. Section 5 describes a comprehensive classification of LDDoS attacks targeting SDN three-layer architectures. Section 6 presents recent research conducted in the field of LDDoS detection mechanisms based on ML and DL in SDN. Section 7 discusses the current challenges and future research direction of LDDoS attack detection methods based on machine learning in SDN. Finally, Section 8 concludes the paper.

2. Methodology

This section describes the methodology we used to conduct our survey. The search results show the interest of researchers in developing methods for detecting LDDoS attacks. We also found a large difference in the publications on LDDoS attack detection methods for traditional IP networks compared to LDDoS attack detection methods for SDN, as shown in the diagram in Figure 1.

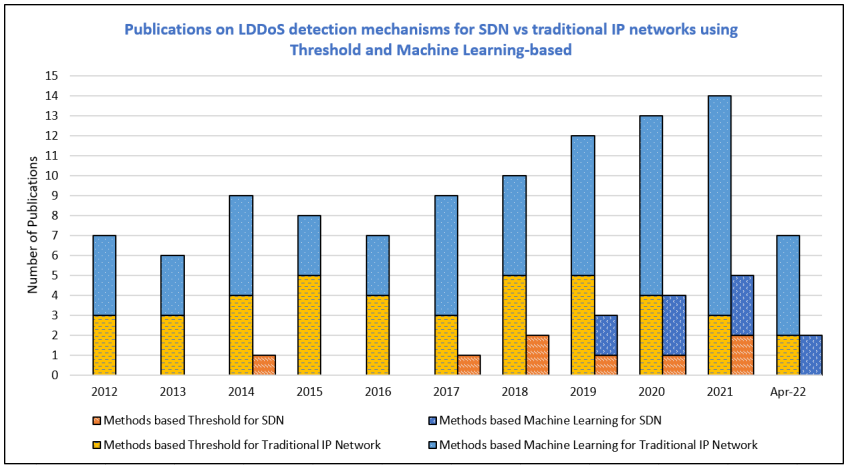

Figure 1. Publications on LDDoS detection mechanisms: SDN vs. traditional IP networks.

2.1. Search Strategy

In this study, we conducted the search and collection process for relevant studies and articles until April 2022 on eight different databases and data sources to extract and collect related studies from the literature. The databases comprise Google Scholar, IEEE Xplore®, ScienceDirect, Scopus, Springer, Wiley, MDPI, and the PubMed database. The electronic links for databases and data sources searched are as follows:

- IEEE Xplore https://ieeexplore.ieee.org/Xplore/home.jsp; accessed on 23 April 2022;
- Scopus Database https://www.scopus.com/search/form.uri?display=basic#basic; accessed on 23 April 2022;
- Science Direct https://www.sciencedirect.com/; accessed on 25 April 2022;
- Springer https://www.springer.com/gp; accessed on 25 April 2022;
- Google Scholar https://scholar.google.com/; accessed on 25 April 2022;
- Wiley Online Library https://onlinelibrary.wiley.com/; accessed on 26 April 2022;
- MDPI https://www.mdpi.com/; accessed on 27 April 2022;
- PubMed https://pubmed.ncbi.nlm.nih.gov/; accessed on 29 April 2022.

This study used both broad and specific keywords to obtain an appropriate number of studies on the research topic of interest. In particular, broad keywords are used to represent the domain such as "SDN security", while specific keywords are used to refer to specific topics within the search domain such as "LDDoS attack detection techniques". The search terms used are: "Software Defined Network", "SDN", "DDoS", "Low-Rate DDoS", "LDDoS", "Machine Learning", "Deep Learning", and "OpenFlow".

In addition, we reviewed the titles and abstracts of the articles using Mendeley [18] to identify studies relevant to SDN research and to exclude unrelated areas. We then inspected these research papers to identify studies that fell within the scope of our study. For example, we only considered studies that used machine learning mechanisms to detect LDDoS attacks in SDN. Last but not least, we need to exclude studies that are not accessible due to restricted access to databases.

2.2. Extraction of Information from the Articles

Information was collected from the target articles as follows:

1. Title of the research paper;
2. Contribution to the research;
3. Category of detection mechanisms;
4. Type of attack detection;
5. The method used;
6. Accuracy and other measures;
7. Dataset;
8. Experimental setup.

2.3. Methodology Used

The main methodological steps used in conducting the survey are as follows:

- Conducting a search for the relevant resources on the Internet;
- Summarize research findings and identify key trends.
- Selecting the most relevant papers;
- Developing a classification of SDN vulnerabilities for LDDoS attacks;
- Summarize and analyze LDDoS detection mechanisms for SDN;
- Identify current limitations and future research directions.

3. Related Work

There are previous surveys that provide a comprehensive overview of research on detection mechanisms for DDoS attacks in SDN as stated in [15,19–25]. The authors of [15,19–25] present several DDoS detection methods in SDN and illustrate some of the remaining frameworks for improving DDoS detection in SDN. Cui et al. [15] analyzed and compared some DDoS detection mechanisms based on the characteristics of SDN and DDoS attacks. The mechanisms are categorized into five types, namely the statistical-based, machine learning-based, threshold-based, hybrid methods-based, and other method-based DDoS detection mechanisms. Balarezo et al. [19] presented a classification approach for existing DoS/DDoS models in different types of networks: SDN, virtual networks, and traditional IP networks. They provided a thorough survey of existing attack models and quantified the damage of DoS/DDoS attacks in network recovery speed and network infection. Dong et al. [20] focused on the analysis DDoS detection mechanisms and attacks in SDN and cloud computing architecture. Aladaileh et al. [21] presented an overview of detecting DDoS attacks on SDN controllers. They described DDoS detection techniques and classified them according to the techniques or methods used. Xu et al. [22] provide some SDN-based mechanisms against DDoS attacks and machine learning-based DDoS detection methods in SDN and show the advantages of SDN over DDoS attacks in traditional IP networks. Rasool et al. [23] presented a systematic survey of link flooding attacks patterns on SDN architecture as a category of DDoS attacks in SDN. Wang et al. [24] analyzed SDN security from a data perspective and surveyed some typical network attack detection methods, including statistical and machine learning methods. Singh and Behal [25] provide a systematic review of various SDN-based DDoS threats and the existing literature on fast DDoS detection and defense in SDN. They classified the mechanisms into four categories, namely: information theory-based methods, machine learning-based methods, artificial neural network (ANN)-based methods, and other miscellaneous methods.

The surveys in [15,19–25] mainly focus on high-rate DDoS attacks detection mechanism. None of them focus on addressing current detection approaches based on machine learning for low-rate DDoS attacks in SDN. Table 1 summarizes a comparison of this survey work with the existing surveys in the security of SDN. The purpose of our survey article is to provide a complete and up-to-date review of LDDoS detection in SDN by presenting LDDoS detection mechanisms based on machine learning in SDN and explanations linked to the SDN layers, which are control layers, application layer, and data layer. Additionally,

we demonstrate an inclusive classification of LDDoS attacks targeting SDN three-layer architecture.

Table 1. Coverage of Security Issues in Survey Papers.

Prominent Security Issues	Cui et al. [15]	Balarezo et al. [19]	Dong et al. [20]	Aladaileh et al. [21]	Xu et al. [22]	Rasool et al. [23]	Wang et al. [24]	Singh and Behal [25]	Our Survey
Vulnerabilities of all SDN Layers	✓	✓	✓	Control Layer	Control Layer	✓	Data Layer	Control & Data Layer	✓
DDoS Attacks	✓	✓	✓	✓	✓	✓	✓	✓	✓
LDDoS Attacks	✗	✗	✗	✗	✗	✗	✗	✗	✓
Detection or Mitigation Schemes using Machine Learning	✓	✓	✗	✓	✓	✓	✓	✓	✓
Taxonomy of Security Attacks	✓	✓	✓	✗	✗	✗	✗	✓	✓
Categorize Detection Solutions	✓	✗	✓	✓	✓	✓	✓	✓	✓
Limitation of Existing Work	✓	✓	✗	✓	✗	✓	✗	✓	✓
Discussion on Possible Future works	✓	✗	✓	✓	✗	✓	✗	✓	✓

4. Background

This section provides an overview of different machine learning techniques, SDN and the SDN architecture, DDoS attacks, and LDDoS attacks.

4.1. Machine Learning

In this section, we first introduce machine learning; then, we shed the light on different machine learning techniques. Machine learning is the field in computer science that focuses on providing computers with the ability to solve problems through learning, such as in humans [26]. There are some variations in defining the types of machine learning algorithms. Still, in general, they can be categorized by their purpose into supervised, unsupervised, semi-supervised, reinforcement, and deep learning, as discussed and presented in [27–29]. Based on [27–29], machine learning techniques with multiple algorithms that fall under these types can be summarized as in Figure 2.

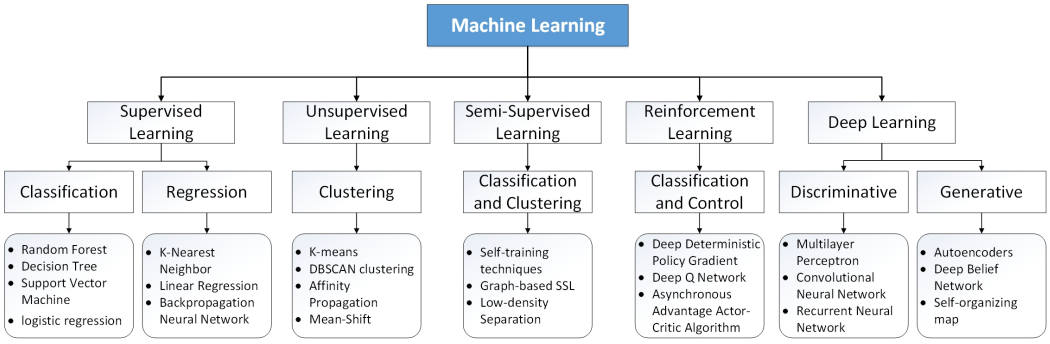

Figure 2. Overview of Machine Learning Techniques.

A supervised learning (SL) model or algorithm is mostly used in solving prediction problems. The model is trained using historical data with labeled inputs to outputs [30]. The training data are analyzed by a supervised learning algorithm, which generates a reasoning function that could be used to map new inputs that have not been seen before [31]. Supervised learning contains two types of techniques, which include classification and regression. A classification technique concludes observed values as one or more outcomes in a definite form. Examples of classification algorithms includes random forest (RF) [32], support vector machine (SVM) [33], decision tree [34], and logistic regression (LR) [35], while a regression technique is mainly used to make predictions on numbers, i.e., when the output is a natural or continuous value. As it falls under supervised learning, it works

with trained data to predict new test data. There are some Regression algorithms such as backpropagation neural network (BPNN) [36], K-Nearest Neighbor (KNN) [37], linear regression [38].

Unsupervised learning (UL) model or algorithm learns from training data that has not been classified, categorized, or labeled [39]. Rather than reacting to feedback, the unsupervised learning model discovers similarities in data and reacts depending on the existence or lack of such commonalities in each new dataset. This is the primary application of unsupervised learning in statistical density estimation. In contrast to supervised learning, which classifies the training data into acceptable categories, unsupervised learning models must understand the connections between the items in the dataset and categorize the raw data without "assistance" [40]. Clustering is an unsupervised learning technique that involves classifying data points into specific groups. If we have some objects or data points, we can apply a clustering algorithm to analyze and group them according to their properties and features. This method of unsupervised technique is used due to its statistical techniques. Cluster algorithms make predictions based on training data and create clusters based on similarity or unfamiliarity. There are several clustering algorithms such as K-means [41], DBSCAN clustering [42], affinity propagation [43], and mean shift [44].

Semi-supervised learning model or algorithm uses mixed classified and unclassified data for training [45], usually a small amount of classified data with a large amount of unclassified data. Semi-supervised learning is in between supervised and unsupervised learning. There are typically two activities in a semi-supervised learning algorithm: The general rule in a semi-supervised model is first tested with classified data, and the rule is then utilized to infer the unclassified data. At the moment, semi-supervised learning performance is still unstable and needs improvement [46,47]. There are different semi-supervised learning algorithms, including self-training techniques [48], graph-based SSL [49], and low-density separation [50].

Reinforcement learning is a technique that allows a machine to interact with its surroundings. The machine can eventually learn from its experience by repeating the process thousands or millions of times. Reinforcement learning differs from supervised learning in that there is no need to introduce input/output pairs; hence, there is no need to explicitly correct sub-level actions. Instead, this technique focuses on finding a balance between exploring uncharted territory and exploiting existing knowledge. There are different types of Reinforcement learning algorithms, including Deep Deterministic Policy Gradient, Deep Q Network, and Asynchronous Advantage Actor-Critic Algorithm [51].

Deep learning is a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain, called artificial neural networks. Recently, deep learning has proven itself in many applications including network security. Deep learning can extract elementary features from data without human intervention. Deep learning provided impressive results with high performance by automatically finding correlations in raw data. With the advent of deep learning-based models, the accuracy in detecting attacks has further improved [51]. Deep learning techniques can be divided into two major categories: discriminative learning and generative learning; discriminative learning technique is utilized to provide a discriminative function in supervised or classification applications. Discriminative deep architectures are typically designed to provide discriminative power for pattern classification by describing the posterior distributions of classes conditioned on visible data [29]. Discriminative architectures mainly include convolutional neural networks (CNN) [52], multi-layer perceptron (MLP) [53], and recurrent neural networks (RNN) [54], along with their variants.

Generative learning technique is typically used to characterize the high-order correlation properties or features for pattern analysis or synthesis, as well as the joint statistical distributions of the visible data and their associated classes [29]. Commonly used deep neural network techniques for generative learning are Autoencoder (AE) [55], Deep Belief Network (DBN) [56], and Self-Organizing Map (SOM) [57].

4.2. Software Defined Network

SDN is a network architecture that separates control logic from forwarding logic in a network in order to provide high flexibility and programmability [58]. The control plane is logically centralized and mainly perform control functionalities of routers in traditional IP networks, as well as providing Application Programming Interfaces (APIs) to support different services offered at the application layer [59]. SDN can support new control functions by creating software-based logic that maps the entire network with various services and applications running at the control plane [60]. The data plane consists of a network of passive switches where each switch mainly perform packet forwarding tasks. Figure 3 depicts the architecture of an SDN. This architecture consists of three main layers namely, application layer, control layer, and data layer. These layers are interconnected by the northbound interface and the southbound interface.

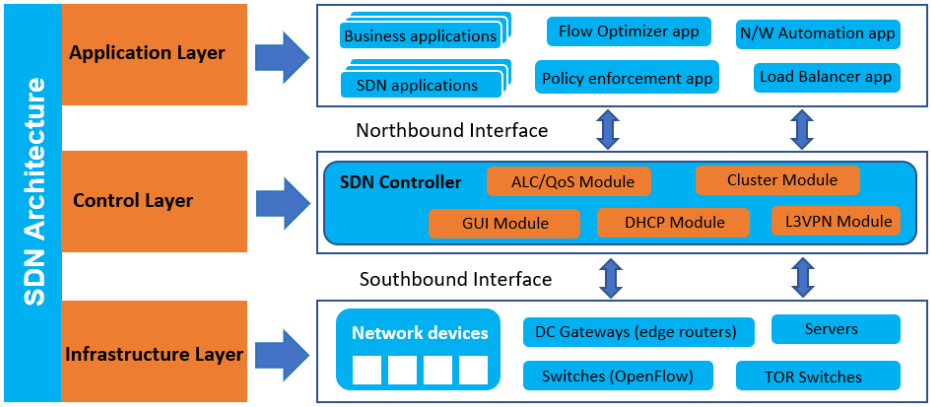

Figure 3. Software Defined Network Architecture [9].

The application layer has a number of programs that can perform network operations. An SDN may consist of more than one controller. The northbound interface allows communications between different controllers.

In an SDN, the controller is the network's primary processing unit. It collects and processes all network's state data. The control layer provides services to the data layer through the southbound interface, which transmits or receives control packets regarding network events such as topology changes of data plane and addition or deletion of forwarding or security rules in the switches.

The Open Networking Foundation (ONF) [61] created the first and most well-known southbound interface, OpenFlow, to describe how the SDN controller should communicate with the data plane. According to OpenFlow [62], rules are called "flows" and are stored in "flow tables" of switches. Various messages are defined by OpenFlow to enable communication between the switches and the controller, including connection setup messages, configuration messages, getting status, lifetime, asynchronous event, and error and experimenter messages [63].

4.3. Distributed Denial of Service (DDoS) Attacks

DDoS attacks are the most significant cybersecurity threats that affect server platforms and network nodes [64] of various networks such as IoT, WSN, and SDN [65–67]. Such attacks transmit a huge volume of malicious traffic to the networks with the purpose of overloading the network resources including network interface throughput or computational load such as CPU loading, causing faults and/or congestion [68]. As typical IP blacklist remedies are based on static regulations, these schemes become ineffective in blocking these sorts of challenging attacks [69,70].

There are several types of DDoS attacks including, SYN flooding, ICMP flooding, UDP flooding, DNS flooding, ping of death, DRDoS, zero-day, and low-rate DDoS. UDP/TCP/ICMP packets generated by any of the aforementioned types of attacks not only overload the

transmission, computation, and storage resources of the attack targets such as servers or SDN controllers but also impact the transmission capacity of other network elements such as OpenFlow switches that must handle the excessive traffic generated by the attackers. Furthermore, a number of DDoS attack tools are accessible for download, thus making it relatively simple to conduct DDoS attacks such as Slowloris, HULK, Tor's Hammer, Xoic, LOIC, PyLoris,DDOSIM, and RUDY. These tools can be used by an inexperienced attacker 1to launch DDoS attacks that are easy to begin but hard to defend and defeat [71].

DDoS attacks may be categorized into three categories, according to the three-layer SDN architecture:

- DDoS attacks at the application layer: SDN apps transmit particular packets to all or most of the switches that support the SDN in an attempt to mislead the application and cause it to fail.
- DDoS attacks on the controller layer: SDN controller adversaries send a huge number of new packets to all or most of the switches that cause overloading the controller's computing or bandwidth capacities.
- DDoS attacks on the data layer: Network devices with SDN capabilities transmit a high number of new packets to the target OpenFlow switch, and attackers attempt to confuse the SDN-enabled switch's stream table storage resource.

According to [72], three defense techniques are commonly used to mitigate DDoS attacks, which are classified depending on the detection engine's location:

- Attacking hosts are used for achieving source-based discovery.
- Victim hosts implement destination-based detection.
- Network-based discovery is implemented in switches and routers that serve as network intermediary nodes.

4.4. Low-Rate Distributed Denial of Service (LDDoS) Attacks

An LDDoS attack is a variant of a DDoS attack. However, the attack traffic launched by LDDoS is always lower compared to DDoS attack where target network's links and nodes are overwhelmed with high traffic. In an LDDoS attack, only 10 to 20 percent of the traffic toward the target is generated, which is hidden in the normal data flow causing no obvious anomaly for network monitors [73]. Although the nature of the attack is small, it will lead to massive destruction at the level of the target network or system. Multiple individual LDDoS attacks producing bursts of pulses, each consisting of low throughput rate with a specific duration, are injected into the network. These bursts can overload the target network nodes or links causing high packet drop, network performance degradation, or even unavailability of service. In short, although the average throughput of an LDDoS attack is low, its impact is comparable to that of a DDoS attack [74]. In the following points, the main characteristics of LDDoS attacks are listed:

- LDDoS is launched to specific target victims. Before attacks are launched, the attacker 1obtains the application service information of the victim or the vulnerability existing in network protocols.
- An attacker 1transmits attack data packets in a low-density and periodic mode so that the network resources of the attacker 1are exhausted, and the network and service performance is reduced.
- LDDoS attack behavior is extremely covert.

Figure 4 illustrates the concept of LDDoS attack targeting two network components: a network link (Target 1) and a node (Target 2). Target 1 is the link between router 4 and router 6, as well as between router 6 and the server. Target 2 is the server itself. Servers are considered the most frequently attacked devices, as they are considered the data repository needed by the users. As shown in the figure, two attackers (Attacker 11 and Attacker 2) launch LDDoS attacks into the network [74]. The traffic of LDDoS attack generated is indicated by the burst of rectangular pulses. The pulses describe a period of packet transmission in short duration and are repeated at a given frequency. The square

pulse amplitude and length define the attack's energy. In other words, the LDDoS attack traffic is averaged across time in the square pulse [75].

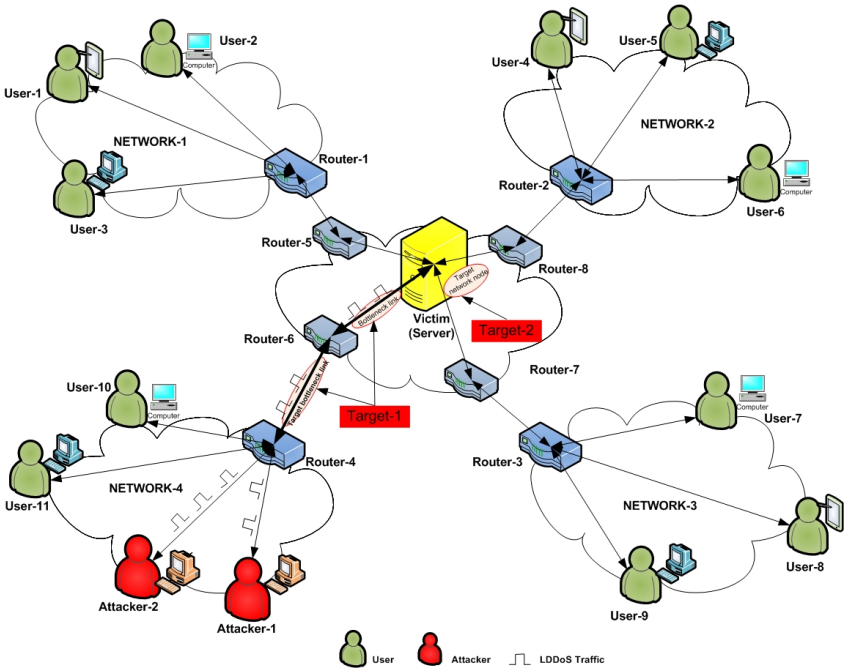

Figure 4. Low-rate DDoS attack concept.

5. Vulnerabilities of SDN to LDDoS Attacks

In this section, the vulnerabilities of SDN to LDDoS attacks are discussed. We have observed that papers of existing work discuss LDDoS attack detection mechanisms for specific SDN layers. For example, References [17,76] discussed LDDoS detection mechanisms for SDN application layer. References [77–80] discussed LDDoS detection mechanisms for the SDN's control layer. References [81–84] discussed LDDoS detection mechanisms for SDN infrastructure Layer. Based on this observation, we developed a classification of SDN vulnerabilities for LDDoS attacks, as shown in Figure 5. In the following subsections, we describe vulnerabilities according to SDN architecture.

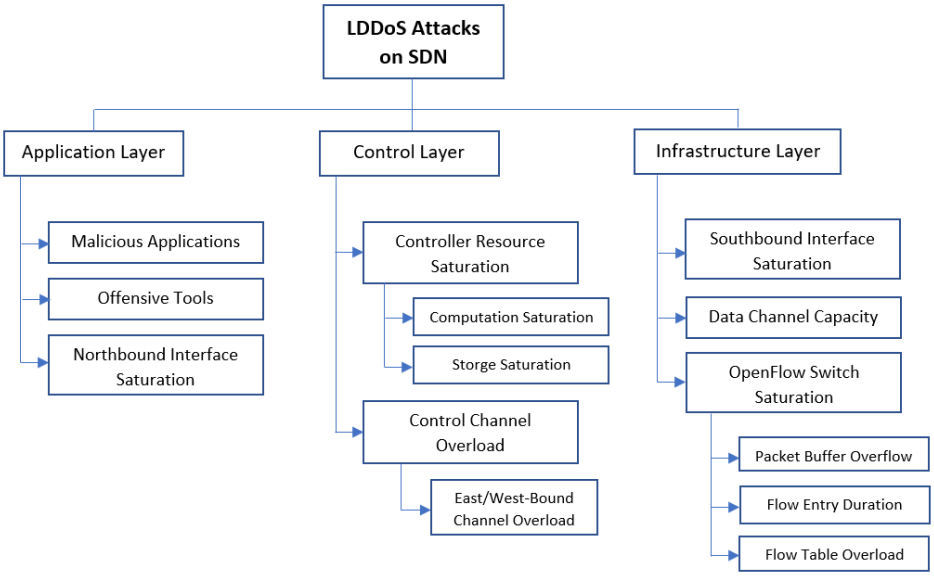

Figure 5. SDN vulnerabilities to LDDoS attacks.

5.1. LDDoS Attack against Application Layer

This section describes the LDDoS attacks corresponding to the specific weaknesses discovered in SDN software and services in the application layer.

5.1.1. Malicious Applications

The malicious program might allow the attacker 1to gain illegal entry and cause wreak destruction by sending backflow requests to the SDN controller in an uncontrollable manner. The attack might cause connection loss between apps and controllers, bringing the entire network down in the worst-case state. The lack of standardization in security methods raises the likelihood of LDDoS attacks while also reducing the reliability and confidence between SDN controllers and the applications [76].

5.1.2. Offensive Tools

The application layer is vulnerable to various attacks carried out by dedicated tools such as DDoSSim [85], GouldenEye [86], H.U.L.K [87], Slowloris [88], and CloudFlare [89]. The success rate of IDS in detecting LDDoS attack from these tools is not high [17]. The threat from these tools needs ML algorithms that do not only look for forms to classify streams as legitimate or illegitimate. Rather, the kind of LDDoS attacks needs to use features such as connection periods and memory marks to obtain a more accurate classification.

5.1.3. Northbound Interface Saturation

SDN fosters creativity and programmability with northbound APIs that connect the SDN controller to applications operating across the network. It enables software designers in creating apps that allow the network to be programmable. This produces northbound APIs without widely accepted standards [90]. This gap is thought to offer vulnerabilities to a variety of security risks originating from trust concerns, app authorizations, and illegal application gain access. Because of these defense concerns, LDDoS attacks directed against the controller may produce congestion on the northbound interface. For instance, the LDDoS attack will utilize a bandwidth on the northbound channel; if a traffic inspection application listens in on Packet-In arriving from the switch in the direction of the controller, the network over time is overwhelmed [91].

5.2. LDDoS Attack against Control Layer

LDDoS attacks are classified in this section into bandwidth saturation attacks and resource saturation attacks against the control layer. In this layer, attackers seek to consume SDN controller resources (e.g., CPU and memory). On the other hand, this consumes the bandwidth capacity of its channel. It is known to target an SDN architecture's control channel, which is the southbound API, data channel, and northbound channel [92].

5.2.1. Saturation of Controller Resources

As mentioned in background section for SDN, the controller is a centralized component in SDN. Although this offers many advantages of SDN, it represents a single point of failure [93]. An LDDoS attack can cause significant damage across the SDN by targeting the controller with a resource overload attack. This attack, similarly to the control channel bandwidth saturation attack, achieves its goal by taking advantage of a vulnerability in the standard SDN design. In order to deal with spurious packets, the SDN controller sends rules and policies packets to network elements such as OpenFlow switches to drop these packets and track the sources of the sent IP addresses, thus draining the controller's resources, such as CPU processing capabilities, RAM, physical memory, etc. [94].

5.2.2. Control Channel Overload

The Control Channel, or Southbound API, connects the SDN controller to OpenFlow switches using the OpenFlow protocol. The controller provides network switches with decision-making capabilities for routing and regulating network traffic [95,96]. An LDDoS

attack can target the control channel by making it always occupied since OpenFlow switches are dumb devices and rely on the controller for routing decisions and regulating traffic. Increasing the motivation of the switch to send query messages will lead to an increase in decision-making messages from the controller to the switch, which causes a control channel bottleneck [97].

5.2.3. East Westbound Channel Overload

Multiple controllers are used to extending the network when a single central controller is not sufficient to control network traffic due to the increase in the number of network adapters. These multiple distributed SDN controllers will connect through East–West APIs as interfaces. East–West API has no defined interface, and each controller must implement its policies and routing protocols [98]. As a result of this restriction, the interoperability area between many SDN controllers is constrained, making them vulnerable to an LDDoS attack by inducing multiple controllers to send recursive synchronous messages and, thus, flooding the east–west bound API channel.

5.3. LDDoS Attack against Infrastructure Layer

This section describes the vulnerabilities of OpenFlow switches to LDDoS attacks. OpenFlow switch has issues such as flow table overfilling, packet buffer overflow, and flow entries timeout length. Because an LDDoS attack may target and exploit many vulnerabilities simultaneously, they are all inextricably connected.

5.3.1. Data Channel Capacity

A data channel is a connection that allows transferring the traffic between two or more OpenFlow switches. The data channel has a larger capacity than the control channel and has a large bandwidth [99]. Despite this, even though the impact of an LDDoS attack on data channel performance is currently unknown, an LDDoS attack can target it in several ways. An LDDoS attack may traverse through many switches before reaching the controller and entering the victim's network. Various data channels or links between switches will be blocked by rogue packets, making them more or less busy as a result.

5.3.2. Packet Buffer Overflow

When the switch receives a new packet, it buffers the packet before forwarding only the header to the controller through a Packet_In message. The SDN controller determines the best path for evaluating the packets and sends them back to the switch as an output packet. The OpenFlow agent (OFA) receives the packet and adds new flow rules to the packet buffer flow and the flow table depending on the controller's response. OpenFlow switches contain a low-capacity packet buffer flow [74]. Because of this limitation, LDDoS attackers may keep sending new packets to induce the OpenFlow switch to send the Packet_In message, which will consume and fill the packet's buffer flow.

5.3.3. Flow Entries Timeout Length

In the OpenFlow switch, each flow table has a flow entrance timeout process. A timeout specifies the lifespan, or period, of a flow account in the switch. Idle and hard timeouts are the two sorts of timeouts associated with flow entry. The flow entry will expire if no traffic is collected earlier than the idle timeout rate is reached if the idle timeout value is not zero [100]. Based on this, LDDoS attackers can exploit the lifetime of flow entries by analyzing the behavior of the flow table in flood attacks and then execute LDDoS attacks that continuously send a flow over a long time to mimic the behavior of a regular flow. Therefore, this malicious flow can pass without being detected by traditional detection mechanisms. The impact of the attack is limited to the switch because it exploits the timeout mechanism within the switch [78].

5.3.4. Flow Table Load

In the OpenFlow switch design, the flow table rules are stored in a high-cost memory known as TCAM (Three Content Addressable Memory), which is also restricted to holding only 2000–3000 flow rules in OpenFlow switches [101,102]. This shortcoming in the size of the flow tables attracts the attention of potential LDDoS attackers and makes them vulnerable to flooding. The switch sends an Opet_Flow_Mod_Failed error message with the fault code Ofpfmfc_Table_Full when there is not enough room in the flow table for more entries [103]. A study in [104] dubbed this vulnerability as a "TCAM exhaustion attack". It examines the slow TCAM exhaustion technique, a one-of-a-kind variant of this method. Once compared to the TCAM over tiredness attack, this attack proceeds with UDP flow, which is much slower, while still filling the flow table and interrupting essential client services.

In summary, this section presented SDN vulnerabilities to LDDoS attacks at all SDN layers, including the application layer, north interface, control layer, south interface, and infrastructure layer, in the classification shown in Figure 5. Recent working papers [17,76–84] discuss SDN vulnerabilities for LDDoS attacks in specific SDN layers, but none of these papers provides a comprehensive and detailed analysis of all SDN layers. As a result, we found that SDN layers are vulnerable to LDDoS attacks that can exploit one or more of these vulnerabilities, including exploiting malicious applications, using offensive tools, saturating northbound saturation, saturating controller resources, overloading control channels, and overflowing OpenFlow switches.

6. Machine Learning Based LDDoS Detection Mechanisms

We observed that machine learning is the most effective proposed LDDoS detection mechanism after investigating numerous LDDoS detection techniques. This part will discuss machine learning methods for detecting LDDoS attacks. We categorized machine learning-based LDDoS detection mechanisms into classification-based used algorithms. Such as (SVM, KNN, FCM, J48, RT, RF, MLP, FM, and NB) and deep learning-based use algorithms such as (CNN, RNN, and LSTM), as summarized in Table 2.

Table 2. Analysis of LDDoS detection methods used by existing studies.

Reference	Year	SDN Layer Location	Machine Learning-Based	Classifier/Method	Detection Results
Zhijun et al. [81]	2019	Data Layer	Classification Based Detection	FM	95.8%
Phan et al. [76]	2019	Application layer, Northbound		SVM, RF, Q-Learning	98%
JESÚS et al. [17]	2020	Application Layer, Control Layer		J48, RT, RF, MLP	95%
Cheng et al. [78]	2020	Data Layer, Control Layer		SVM, NB, RF	97%
R. Khamkar et al. [77]	2021	Control Layer		SVM	99%
Wencheng et al. [79]	2021	Control Layer		SVM, NB, LR, DT, C4, RF, AB	92%
Tang et al. [83]	2021	Control Layer		GBDT, GBDT-LR	96%
Sudar et al. [80]	2022	Control Layer, Data Layer		SVM, DT, NB	93%
Nugraha et al. [82]	2020	Data Layer	Deep learning-based Detection	CNN-LSTM	99%
Sun et al. [84]	2022	Control Layer, Data Layer		CNN-GRU	99.5%

6.1. Classification-Based LDDoS Detection

Classifying algorithms are frequently used as classifiers to identify LDDoS attacks in SDN for LDDoS detection. According to our findings, SVM, Random Tree, J48, REP Tree, MLP, SVM, and RF are the most often utilized classification algorithms in SDN for identifying LDDoS attacks.

An LDDoS Detection framework based on classification may be placed at a crucial node of an SDN, such as the SDN controller or the OpenFlow switch [105]. The framework consists of four main stages, as shown in Figure 6: data collection, feature extraction, classification model, and output. In the first stage, data collection consists of packet capturing, packet filtering, and dataset preprocessing steps. In the second stage, the feature selection and selection policy will be generated based on the previous stage; the feature value can be divided into stateless and stateful. In the third stage, the model is trained for the classification process to detect an LDDoS attack using several binary classification methods such as RF, SVM, J48, and test datasets. In the final stage, the user decision is made based on user-defined feedback, which is continuously taken to support periodic learning and testing of the model prediction process, correlation, and recognition of LDDoS attacks from regular traffic.

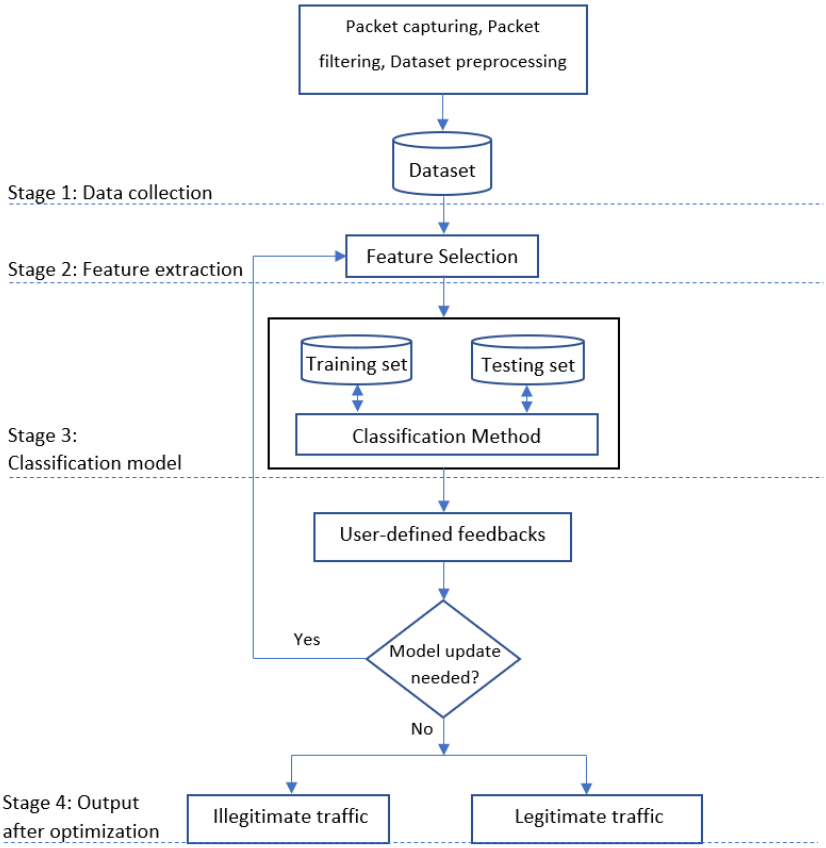

Figure 6. Architecture of machine learning-based classification for LDDoS attack detection frameworks for SDN.

In [81], Zhijun et al. presented a protection mechanism based on dynamic flow rule deletion. They investigate the mechanism of LDDoS attacks on the SDN data layer to increase detection accuracy and then offer a multi-feature LDDoS attack detection method based on Factorization Machine (FM). The authors employ AUC (area under the curve) [106] as the primary metric of recognition performance, with recall rate, accuracy, and precision as supplementary indicators, to assess the impact of machine learning classification FM. The samples were separated into two groups: one for anomalous traffic and the other for regular traffic. The indicators were developed using the number of true/false positive

and true/false negative groups. The dwell time is gathered and computed when the window function's size is set to 20 s; the number of packets, RDMB, and RDPI to save every flow rule in the flow table by the window function; and the variance of the dwell time are calculated. To define the difference between attack and regular traffic, the data were identified based on the attacked port, and a total of 140,000 feature datasets were obtained. A total of 100,000 feature datasets were chosen as training datasets, with 40,000 feature datasets chosen as test datasets. The hidden vector dimension and the learning rate are the two parameters that most influence the machine learning algorithm FM's performance. As a result, four different parameter settings with an average attack rate of 2.2 Mbps were chosen for the experiment.

The testing revealed that the approach could successfully detect LDDoS attacks on the SDN data layer, with a detection accuracy of 95.80%. The FM approach produces a stable environment for guarding against LDDoS attacks since it can identify fine-grained LDDoS attacks. With a 97.85% success rate in forwarding regular packets, it uses experimental simulation and analysis to demonstrate the defensive strategy's efficacy. The NSL-KDD, DARPA98, and CAIDA datasets were used to assess performance in a simulated environment.

In [76], Phan et al. proposed Q-MIND: a machine-learning-based defense framework in SDN to defeat LDDoS attacks. The authors begin by looking at the adversary model for LDDoS attacks, as well as critical weaknesses in SDNs and characteristics of LDDoS attack. They describe Q-MIND, which includes the optimal strategy of the Q-learning proxy to defend against LDDoS attacks in SDN effectively. Throughout the training phase, the authors require a collection of labeled traffic data that includes both anomalous and benign samples. The authors use the Markov Decision Process (MDP) approach [107] to realize the optimal combination of a given feature set and performance for AI/machine learning algorithm detection. The MDP architecture allows AOS (Application Operator and Scheduler) to choose the best action (a mixture of features and AI/ML algorithm) based on its inputs to extend its immediate incentive in each loop. They have implemented ten features, including average packet size per stream, stream change percentage, port extension, etc. These characteristics are calculated for each source IP address using a traffic dataset with 8000 samples divided 50% for regular traffic and 50% for attack traffic. As a result, the Q-learning agent must train the detection mechanism and conduct validation tests. They design an optimization problem to find the best policies in scenarios where the immediate reward increases in each loop, using Q-learning to solve it. This method allows AOS to make the best choice without knowing a collection of features or the associated AI/ML technique beforehand. The Q-MIND framework provides results close to those obtained during the cross-validation phase with the optimal policy (close to 98%). To assess attack mitigation performance, the authors look at a fraction of malicious stream rules rejected in the switch and the web server's request-response time when the network is under attack. Q-MIND achieves a high proportion of correctly downed attack streams because it enforces rules when it recognizes a source IP address emanating from an attacker.

In [17], JESS et al. proposed a modular architecture for detecting and mitigating LDDoS attack in SDN environments. They used six machine learning models (SVM, J48, Random Tree, REP Tree, MLP, and RF) to train their design and used the CIC (Canadian Institute of Cybersecurity) DDoS dataset with implementing ONOS controller running on the Mininet simulator [108]. Despite the challenges of detecting LDDoS attacks, the evaluation findings reveal that the approach has a 95% detection rate. The interface-based classification technique provides a classify object that defines a stream property with a classifier. The identification API offers a way for trained robots to interact with process flows. The classifier is a collection of learned machine learning models available in IDS. A sophisticated stream object gathers stream information from OpenFlow switches to determine if the traffic is an attack or a normal one. This flow complicated object's properties are based on the flow table. IDS obtains a collection of stream parameters from the input request with traffic flow parameters and the output response with flow classification and then creates the model provided in the request. If the stream is considered to be anomalous,

it uses the specified form to classify the input parameters before determining the type of attack. Finally, it generates an attack-type 1 data response or an attack-type 0 response, which it transmits to the IPS.

In [78], Cheng et al. proposed a learning-based technique for detecting LDDoS attacks on SDN control and OpenFlow switches in an IoT network context. The suggested approach is based on two characteristics derived from the OpenFlow package (stateless and stateless features). Learning techniques are used in the detection system to construct classifiers that distinguish between normal and attack flows. The approach trial's evaluation architecture includes an LDDoS attack module, data flow collection, and feature extraction model. They put several learning algorithms to the test and distributed each feature in the raw data based on the platform. Two sets of stateless vectors and stateless features are created. A training set and test set are created from the produced feature vector dataset. The learning and detection phases of classification-based learning are then separated. Part of the dataset created in the preceding phase is used in each of these work steps. There is no intersection between the two datasets. The typical quantitative ratio for test datasets and the training dataset is 20% for the test and 80% for the training. The procedure employs binary classification methods such as SVM, NB, and RF. The experimental results showed that the model accuracy of detecting LDDoS attacks is 97% from the controller view of the network traffic.

In [77], Khamkar et al. proposed a framework to identify and defend against LDDoS attacks based on machine learning for the campus network. The framework consists of two phases: the network traffic acquisition module and the LDDoS attack identification module. The traffic acquisition module extracts network traffic features to prepare traffic identification for SDN network. They used the SVM algorithm and traffic features obtained from statistical flow table data to identify the attack traffic, demonstrating how LDDoS attacks may deplete controller sources and providing a way out to detect attacks with the changing of the target IP address. Traffic data were obtained from entries in the traffic flow tables. The traffic dataset includes traffic that denotes a TCP connection with eight features. These features are divided into two categories: Host-based and Time-based network traffic. The SVM model is built using LKF (Linear Kernel Function) and a decision function is obtained in order to determine how the packets should be sent. It is an LDDoS attack packet if the output of the function is -1; otherwise, it is a normal packet. The experiment yielded an accuracy of 0.998, indicating the effectiveness of the model. The model has a high detection rate for recognizing LDDoS attacks and is implemented as an LDDoS detection module in a simulated SDN context of a campus network. The controller drops packets that match the configured rule when an attack flow is selected.

In [79], Wencheng et al. proposed DIAMOND as a structured coevolution method to optimize features for LDDoS attack detection in SDN-enabled IoT networks. The method aims to find the optimal features to improve the detection performance of LDDoS attack detection methods. DIAMOND consists of a clustering algorithm to sort the achievable number, a commutation strategy, a group structuring method, and a co-crossover strategy. By analyzing the information about SDN-enabled IoT network features in the solution space, the relationship between various SDN-enabled IoT network features and the optimal solution is investigated in DIAMOND. Then, the individuals with the related SDN-IoT network features are divided into different subpopulations, resulting in a structure tree. Moreover, multiple structure trees evolve in coordination with each other. The evaluation results show that DIAMOND can effectively select optimal low-dimensional feature sets and improve the performance of the LDDoS detection method in terms of detection accuracy and response time.

DIAMOND is implemented on the SDN controller and collects network traffic by sending aggregated traffic instructions to the OpenFlow switches. It works as follows: Once the SDN controller receives the instructions, it sends OFPT_STATS_ REQUEST messages to the switches to obtain flow entry statistics. The switches respond to the SDN controller with OFPT_STATS_ REPLY messages. These messages contain stream data in the flow

entries for the switches. After the controller receives OFPT_STATS_ REPLY messages, it deletes them and sends the collected statistical data to DIAMOND. The statistical data of the stream entries collected from the switches are used to calculate the initial SDN-enabled IoT network features. Then, these features are encoded using the binary coding method to generate the initial community, which consists of some individuals. The population is divided into several subpopulations by BONNET. The structuring factor is calculated based on the fitness and membership factors, which assemble the individuals in the subpopulation into a skeleton tree. Multiple skeletal trees evolve cooperatively with each other through mutation. After evaluation and selection, the dominant individuals are retained in the next generation. After several iterations, the individual with maximum fitness is identified as the best individual. Then the best individual in the corresponding feature set is decoded. Furthermore, to verify the detection ability of the optimized feature subset, eight different classifiers have been applied to evaluate the feature optimization methods (SVM, NB, LR, DT, C4.5, RF, and AB). The experimental results showed that DIAMOND has 92% accuracy in detecting LDDoS attacks from the controller view of network traffic and can effectively identify low-dimensional feature groups to improve the performance of the LDDoS detection method.

In [83], Tang et al. proposed a lightweight real-time Performance and Features framework to detect and mitigate LDoS attacks in an SDN environment. They implemented LDoS attacks in SDN, extracted traffic features using OpenFlow, divided all extracted traffic features into attack performance and attack features, and classified them into two categories. By analyzing the performance of normal traffic in the attack state, performance-based detection determines whether LDoS attacks will be effective by building machine learning models with normal and abnormal network state traffic features. Feature-based detection attempts to figure out the LDoS attack flow based on time-frequency analysis. Potential attack flows can be found by analyzing the morphological characteristics of LDoS attack flows with OpenFlow messages and flow table queries. Once the performance and features model ensures that the attack will take effect while the performance and features model detects the LDoS attack flow, the mitigation method is activated. The Performance and Features model locates the attackers' source IP and the victims' ports. Then it sends a defense rule according to the location result to filter the LDoS attack traffic. The authors conduct online and offline experiments to mitigate attacks on the SDN controller. The experimental results show that the proposed method achieves 96% accuracy in detecting LDoS attacks. The proposed framework can also mitigate LDoS attacks in real-time in an average time of 9.39 s with 0.02 to 0.04 CPU and low memory consumption.

In [80], Sudar et al. proposed a flow-based detection and mitigation framework using machine learning as a classifier to detect LDDoS attacks. The authors extracted the basic attack detection features such as flow duration, number of packets, the relative distribution of packet interval, and relative distribution of matched bytes using a flow management module. The mitigation phase is then performed, where the attack flow information is processed and mitigation rules are checked to avoid an LDDoS attack from the same source. To detect LDDoS attacks by examining the features extracted from the stream management module, the authors applied a machine learning model with SVM, Naïve Bayes, and a Decision Tree. They used the Scikit-Learn 0.23 library to classify the malicious traffic by using the output of the flow management module as input to the model. The mitigation phase is alerted to further process the attack packets when attacks are detected. After analyzing the traffic characteristics, the machine learning models return a value of 0 for regular traffic and 1 for attack traffic. The model forwards the corresponding source IP address to the blocklist table via the detection module as soon as the traffic is identified as malicious, i.e., when it is assumed that it might be an attacker. The mitigation module then alerts the OpenFlow switch to prevent the given stream by immediately creating a port blocking rule. In this manner, the authors hope to match the source IP address of each data stream with the entries in the blocking list table before redirecting them for further operation so that the installed traffic control rules remove them from the flow table after

seven days. This process should help to analyze and update the data flow on a regular basis. Since data flow management and analysis are already part of the SDN architectural design, this model is unlikely to increase network traffic complexity. The accuracy of the performance evaluation was 93% for the classification of the traffic by the proposed machine learning model.

6.1.1. Implementation and Traffic Analysis of Classification-Based Methods for LDDoS Attack Detection

This section presents the implementation and deployment of classification-based LDDoS attack detection methods as well as the analysis of traffic and datasets used. Table 3 shows a summary of the simulation environment analysis and traffic/datasets used by existing methods.

Table 3. Implementation and traffic analysis of existing methods.

Reference	Experiments	Dataset	Controller	Scale
Zhijun et al. [81]	Simulation using Mininet	NSL-KDD, DARPA98, CAIDA	RYU	Medium (4 switches, 9 hostes)
Phan et al. [76]	Simulation using MaxiNet	CAIDA	ONOS	Small (1 switche, 8 hosts)
JESÚS et al. [17]	Simulation using Mininet	CIC	ONOS	Small (3 switches, 5 hosts)
Cheng et al. [78]	IoT hybrid Network	Custom	Floodlight	Medium (4 switches, 9 hostes)
R Khamkar et al. [77]	Simulation using Mininet	KDD99	Ryu	Small (2 switches, 6 hostes, and 1 webserver)
Wencheng et al. [79]	Simulation using Mininet-WiFi	Custom	Ryu	Medium (4 switche, 80 hosts)
Tang et al. [83]	Simulation using Mininet	Custom	Ryu	Small (2 switches, 6 hosts)
Sudar et al. [80]	Simulation using Mininet	CIC	POX	Medium (4 switches, 9 hostes)
Nugraha et al. [82]	Simulation using Mininet	Custom	ONOS	Small (2 switches, 10 hosts)
Sun et al. [84]	Simulation using Mininet	CIC	Ryu	Medium (6 switches, 21 hostes)

In [81], the authors carried out the experiment using a Mininet simulator and Ryu controller to set up an SDN network simulation platform. Under the simulation stage, the server targets the data layer with LDDoS attacks. Two actual hosts are utilized in this experiment. One of the hosts is in charge of operating Mininet. The other host is in charge of running the Ryu controller. The OpenFlow protocol, version 1.3, was employed as the southbound interface protocol in the experiment. The topology of the experimental simulation network uses a tree network design with 4 OpenFlow switches and nine hosts. The link bandwidth is set to 10 Mbps for all hosts linked with a delay of 5 ms, link loss is 0, and the maximum queue size is 1000. The experiment employs the D-ITG (distributed Internet traffic generator) to produce traffic in the background using The ITGSend, and ITGRecv commands on each host node to send and accept traffic. The authors used the CAIDA dataset and injected 100 flows into the network topology as a background flow. Among the 100 background flows, TCP streams account for 80%, ICMP 15%, and UDP 5%, with an average background traffic rate of roughly 1.3 Mbps.

In [76], the experiment uses MaxiNet [109] as a simulation platform and ONOS as the SDN controller. The authors established a simple SDN-based network with a web server and eight hosts. There are four malicious hosts and four benign hosts. All hosts and the webserver run in Linux containers connected by an OpenFlow switch. The authors consider three machine learning-based classifiers (SVM, RF-, and SOM). The feature used is created from 10 convenient features. These features are extracted from a traffic dataset containing 4000 normal traffic samples and 4000 attack samples for each source IP address. The dataset was generated from the conducting simulation experiment for stealthy DoS attacks and the CAIDA dataset. Accordingly, the Q-Learning agent is instructed to train the detection engine and perform cross-validation tests.

In [17], the authors implemented their scenario with one ONOS controller [110], three OpenFlow switches, five hosts, and one web server as a victim. The scenario implies that the attacker 1hosts have been infiltrated and are now part of the botnet, with each attacking computer capable of generating malicious traffic using the SlowHTPTest tool [111]. Legitimate hosts generate regular traffic with a pseudo-random function using curl targeting the webserver. The authors used the CIC DoS dataset, including regular traffic and eight low rate attack variations. They used the Flowtbag function to convert the packet data as traffic to adapt the dataset for SDN environments. It takes 44 features as input set and 44 features as output flows.

In [78], the experiment is conducted in a heterogeneous IoT network deployed on a Mininet simulator. The network topology consists of one physical switch, three OpenFlow virtual switches, and forty virtual nodes. All switches are connected to the Floodlight controller, and the nodes are connected to the switches; 10 nodes are normal nodes, and 30 nodes are attacking nodes. The link bandwidth between the nodes and the switches is 100 Mb/s. The dataset is generated in real-time during the experimental simulation collected on the Floodlight controller and the OpenFlow switches. The classifier was trained with 80% of the combined dataset of regular and attacking traffic, and classification accuracy was tested with a test set from the remaining dataset. The regular traffic includes HTTPS, HTTP, MQTT, ping messages, etc., while the attacking traffic includes a large number of TCP-syn and TCP-retransmission messages in addition to those included in the regular traffic.

In [77], the authors simulated the SDN environment to develop the detection model running on the Ryu controller and network topology with two OpenFlow switches and a web server used as a website. The KDD99 [112] dataset was used in the experiment. The dataset is divided into five key categories, regular, attack, Probe, U2R, and R2L, and utilizes 41 features to describe a connection. The authors' primary focus is on HTTP flood attacks. As a result, data are represented using attack and regular traffic. As a data collection, TCP network connections are employed. The raw dataset is divided into a training dataset and a test dataset. Then the training dataset is used to build the attack detection model. Moreover, the best parameters are determined by repeated testing. The decision function is used to decide how to forward packets. If the output of the process is -1, it is an attack packet. Otherwise, it is a regular packet.

In [79], the authors used Mininet-Wifi to verify DIAMOND (the logic of the program) and evaluate its performance by implementing it on the Ryu controller. Mininet-Wifi is used to simulate an SDN-enabled IoT network consisting of eighty IoT endpoints, eight access points, and eight links. They simulated the background traffic of the IoT network, which includes 10% HTTP, 40% HTTPS, 10% TCP, 5% DNS, 10% SSDP, and 5% NTP. Attack traffic consists of traffic generated by 20 IoT endpoints to launch LDDoS attacks against IoT servers and accounts for 20% of regular traffic.

In [83], the authors set up the experimental topology on the Mininet simulator and built a topology with one Ryu controller connected to two OpenFlow switches (OpenvSwitch) and six hosts. The authors create a 10 Mbps bottleneck link between switch S1 and S2. For comparison, the bandwidth of the rest of the links is 100 Mbps with a delay of 20 ms. They used a Python socket to create five long TCP connections for a regular user group

and used the Iperf tool to measure the bandwidth of the bottleneck link created by a short connection. They used two hosts as attackers and a UDP background generator. They considered LDoS attacks initiated by both TCP and UDP as a dataset. In a TCP-based attack, the attacker 1uses the TCP raw socket to create sudden congestion in the network. In a UDP-based attack, the attacker 1initiates UDP pulses and uses this protocol without congestion control to affect TCP directly. The duration of each set of experiments is 720 s, including LDoS attacks based on TCP and UDP for 3 min each. The default detection window is set to 10 s, and the sampling interval is 0.5 s. The construction of the detection window overlaps, and the subsequent detection window is created from a temporary sampling interval behind the previous detection window. Each set of data contains about 1440 detection windows. The training set includes two-thirds of the detection windows, and the rest are used as a test set.

In [80], the authors use Mininet to simulate the network topology with nine hosts and four OpenFlow switches. The hosts are configured as legitimate devices and offensive devices. All hosts are connected to the switches with a bandwidth of 10 Mbps. All switches are connected to the POX Python controller to simulate the control plane. To construct LDDoS attacks, they used the Scapy program [113,114] to generate a low rate attack by spoofing packet fields and a transmission rate of 1.6 to 2.2 Mbps. The CIC DOS 2019 dataset was used to train the machine learning model, with 80 features that provide statistical information about the streams. The Flowtbag method [17] is used in this work to convert traditional traffic states to flow states with the same set of features.

6.1.2. Limitations of Classification-Based Methods for Detecting LDDoS Attacks

This section will present some of the challenges and limitations in classification-based LDDoS attack detection methods. Table 4 represents a summary of analysis methods used by existing work and their shortcomings or disadvantages. In [81], the authors presented a protection mechanism based on the dynamic deletion of flow rules. They applied the mechanism of LDDoS attacks to the SDN data layer to increase the detection accuracy and provide multiple functions based on FM. Although the proposed mechanism has acceptable accuracy (95%) in detecting an LDDoS attack, it was derived from a small network simulation (four switches and nine hosts). Although the model targets the data layer, it does not consider the synchronization process at the data layer and the scenario of connecting to more than one SDN controller. Therefore, the proposed model may fail in large-scale networks where more than one SDN controller is deployed in the absence of a mechanism based on synchronization between SDN controllers and data layer devices. In [76], the authors proposed Q-MIND to detect LDDoS attacks using a reinforcement learning technique based on Q-learning. Throughout the training phase, the authors require a collection of labeled traffic data that includes both anomalous and benign samples. These characteristics are calculated for each source IP address using a traffic dataset with 4000 regular traffic samples and 4000 attack samples. As a result, the Q-learning agent must train the detection engine and conduct validation tests. They design an optimization problem to find the best policies in scenarios where the immediate reward increases in each loop, using Q-learning to solve it. However, compared to similar models, the cost of this method for implementation and training is very high. In [17,78], the authors have used various machine learning techniques to detect and identify LDDoS attacks and have indeed achieved high accuracy.

Nevertheless, these methods can increase controller overhead and decrease response efficiency when used on an extensive network. Analyzing the work [77], where the authors developed an SDN framework based on machine learning to identify and mitigate LDDoS attacks. They utilized two aspects of the system (traffic collecting and flow table delivery). They used the SVM algorithm and traffic characteristics obtained from statistical flow table data to identify the attack traffic. However, the method identification process of determining the feature obtained from the flow table allows the SDN controller to write effective new rules to prevent LDDoS attacks is not addressed, which is essential for efficiency and network connectivity. In [79], the authors proposed DIAMOND as a structured coevolution

feature optimization method to detect LDDoS for SDN enabled IoT networks. They used the group structuring method, counting strategy, and the intersection strategy to sort the count. The authors presented the method's shortcomings, as it does not provide a solution for forming sub-populations that build the individuals into a hierarchical tree. Moreover, the excessive load on OpenFlow switches by querying statistical data from flow entries increases switching overhead and decreases responsiveness to regular traffic. In [83], the authors proposed a detection and mitigation framework for LDoS attacks in SDN. The performance and features framework considers both attack performance and attacks flow features for LDoS attack detection decision. They performed a joint detection approach based on two aspects and implemented a real-time mitigation system to defend against LDoS attacks based on an SDN controller. However, the authors testify that feature classification and selection still need improvement. In addition, the framework forces the controller not to install new rules for the upcoming traffic during LDoS attacks, which results in the legitimate user experiencing more delays when the network is attacked. In [80] the authors proposed their detection mechanism uses four features (duration of the flow, number of packets, relative distribution of matched bytes, and relative distribution of the packet interval) of the network flow. Once the module detects the LDDoS attack, it adds specific details of the attack flow to the blocklist table. It warns the controller to remove a particular flow from the flow table by entering mitigation rules. Although this model reduces resource consumption, it has a high false-positive rate for traffic flows such as ICMP.

Table 4. Analysis methods used by existing work and their shortcomings or disadvantages.

Ref.	Model Type	Objectives			Accuracy	Shortcoming/Disadvantages
		To Increase the Detection Rate by Extracting Features	To Make More Rapidly Classification by Fully Utilizing the GPU	To Improve Detection Rate by Collecting Flow Statistics from Wwitches		
[81]	A multi-feature method based on the FM algorithm	✓			95.8%	Not provide a mechanism based on synchronization between SDN controllers and data layer devices.
[76]	Learning-based approach using the QMIND framework		✓		98%	A complex training process and high cost of implementation.
[17]	Modular architecture based on SDN.		✓		95%	Increase the controller overhead and decrease response efficiency.
[78]	Learning-based detection using stateful and stateless features	✓		✓	97%	Excess load on the controller, which reduces the efficiency of its work.
[77]	Traffic summation framework based on SDN	✓		✓	99%	Not identifying the feature obtained from the flow table to detect LDDoS attacks.
[79]	A structured coevolution feature optimization method	✓	✓		92%	Overloads OpenFlow switches and reduces responsiveness to the regular traffic.
[83]	Performance and Features framework	✓			96%	A legitimate user experiences more delays when the network is attacked.
[80]	Flow-based detection framework using ML classifiers	✓		✓	93%	High false-positive rate for legal traffic flows such as ICMP packets.
[82]	CNN-LSTM Model for SDN-based networks			✓	99%	Requires a long time to train and is limited to traffic types.
[84]	CNN-GRU Model for SDN-based networks			✓	99.5%	Resource intensive from a training perspective and not designed to detect online attacks in the contextof live SDNs.

Although the discussed methods show high accuracy in detecting LDDoS attacks, no work indicates the implementation steps and the feature selection process. Therefore, there is still a need to exploit the characteristics of the SDN against LDDoS attacks and to develop detection models based on machine learning with a high true-positive rate and a low false-positive rate for all traffic features. In addition, an effective extracting feature model must be provided for SDN traffic and dynamically updated to improve LDDoS detection mechanisms.

6.2. Deep Learning-Based LDDoS Detection

CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), LSTM (Long Term Memory), and other deep learning models have been applied to increase SDN protection against LDDoS threats [82,84]. Feature extraction is automatic in deep learning

models [115]. The general structure framework for deep learning-based LDDoS attack detection mechanisms is similar to the machine learning classification-based framework, as shown in Figure 6. However, in the case of a deep learning-based model, selecting features is performed automatically. Table 2 summarizes deep-learning-based LDDoS detection mechanisms along with classification-based mechanisms.

In [82], B. Nugraha et al. present an ensemble convolutional neural network solution for LDDoS attack based on deep learning. To detect LDDoS attack in SDN-based networks, the authors suggest CNN-LSTM, a hybrid Convolutional Neural Network-Long-Short Term Memory model. Custom datasets are utilized to evaluate the strategy's effectiveness. The results are impressive; all of the studied performance metrics have a value greater than 99%. The hybrid model CNN-LSTM outperforms traditional deep learning approaches comparable to MLP (MultiLayer Perceptron) [116] or 1-class SVM [117]. The detector unit in this deep learning-based model evaluates the extractor unit's output and determines whether the entries are classed as harmful or harmless traffic. As a result, it functions as a binary classifier, classifying malicious traffic as "1" and non-malicious traffic as "0". The labels are connected with the original IP address since a single host might be authentic or fraudulent. In depicting the hybrid CNNLSTM model used in the detector module, the pointers $\times 1$, $\times 2$ .., $\times n$ relate to the features associated with each entry.

In [84], W. Sun et al. developed a technique for detecting LDoS attacks based on the hybrid CNN-GRU model: Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU). This work is based on the three main stages of the LDoS attack detection process. The values of variables such as n packets and n bytes are retrieved from the base traffic flow in the first stage, and the average number of packets and bytes is gathered as input data for the combined model. The model enhances detection performance in the second phase by refining the Sailfish method to automatically adjust the CNN and GRU hyperparameters throughout the training process. In the last step, the model employs the CNN hyperparameters and the improved GRU to extract the input data's deeper spatial and temporal aspects to identify LDoS attacks accurately. These stages gradually improve the manual setup approach to automatically identify the CNN and GRU metaparameters before introducing the Sailfish optimization algorithm to optimize CNN-GRU metaparameters. In addition to the nine field values collected from the flow rule. Given that LDoS data have the property of integrating Spatio-temporal features, the authors employ CNN and GRU to extract the spatio-temporal aspects of the data in order to detect attack flows.

The feature selection technology of the CNN-GRU model-based LDoS attack detection method is divided into three stages: feature selection, structure and parameter optimization, and attack detection. Feature selection is performed by requesting flow table information from the switch; the controller sends a flow status request to the switch and selects and develops input features for LDoS attack detection. In the second stage, the syntax is optimized using the ASFO method to optimize the CNN-GRU hyperparameters automatically. The individual population mode of the CNN-GRU model is a form of combining hyperparameters. In the final stage, the optimized CNN-GRU hyperparameter model is used to detect the attacks. Suppose the result of detection is an attack. In that case, the model sends instructions to the controller to change the flow table to the switch and change the configuration of matching elements in the flow table that discards the attack data packet to avoid risks. The most significant value of this model's accuracy detection findings was 99.5%, with a false positive rate of 0.07%.

6.2.1. Implementation and Traffic Analysis of Deep Learning Methods Based on LDDoS Attack Tetection

This section presents the implementation and deployment of LDDoS detection mechanisms based on deep learning as well as the analysis of traffic and datasets used. Table 3 shows a summary of the simulation environment analysis and traffic/datasets used by existing methods.

In [82], the authors experimented with Minnet in a simulated environment where they created a small SDN topology and generated regular traffic and an LDDoS attack. They build a network topology scenario with one ONOS controller, two OpenFlow switches, a web server victim, six regular hosts and four malicious hosts. The authors used the Flow Collector module to generate regular traffic flows (which is responsible for requesting the flow statistics periodically through the REST API from the SDN controller). For generating LDDoS attacks traffic, they used an open-source tool Slowloris-HTTP [118]. In total, they generated 468,002,403 traffic flows, consisting of 355,758,086 regular flows and 112,244,317 LDDoS attack flows. The collected data are divided into 20% test data and 80% training data. The training datasets are then split into 20% validation data and 80% training data.

In [84], the authors used Mininet to design and simulate the network. The network topology contains 1 Python-based Ryu controller, 6 OpenFlow switches, and 21 hosts. They performed experiments using two datasets. The first dataset was collected from the network topology traffic, and the second dataset is CAIDA. Each data collection contains both the training and test sets. The Hping3 tool creates the attack traffic in the first dataset. The idle timeout is set to 10 s, and the attack period is set to 9 s. The data from the network traffic dataset are used to construct the regular flow. The second dataset comprises normal traffic and LDoS attack traffic, with a total of 79 parameters and an 80% to 20% training set to test set ratio.

6.2.2. Limitation of Deep Learning Based LDDoS Attack Detection

This section provides an overview of the limitations of LDDoS detection mechanisms based on deep learning. According to our findings, the number of LDDoS detection mechanisms based on deep learning is lower than those based on classification. This could be due to the fact that classification-based LDDoS detection mechanisms have reasonable detection accuracy with lower computational complexity. Table 4 represents a summary of analysis methods used by existing work and their shortcomings.

In any case, LDDoS detection mechanisms based on deep learning perform better than classification-based LDDoS. This is because learning-based LDDoS detection approaches have a more profound ability to extract traffic than classification-based techniques. Therefore, it can improve detection accuracy in identifying LDDoS attacks and obfuscation.

Various drawbacks or limits must be addressed when utilizing deep Learning to identify LDDoS. The first is that a deep learning model's training period might be exceedingly long. As a result of the lengthy training period, the deep learning unit may have significant hardware needs. The second barrier is that several parameters must be given when developing a deep learning module. Using CNN as an example, the user should pick the optimal or sub-optimal parameters, such as the number of layers, neurons, filters, epochs, learning rate, objective function, and the weight configuration.

Analyzing the work in [82], we note that it has an apparent shortcoming in the use of test parameters since the evaluation of the detection framework was performed with offline datasets in a small SDN topology. There is also a limitation in traffic diversity as the regular traffic flows are limited to UDP and HTTP, and LDDoS attack traffic flows are HTTP only. Moreover, since the work uses CNN-LSTM based on deep learning, the training performance requires a long time when we compare it with classification-based methods. This is because the hybrid CNN-LSTM model requires a 3D data format as input [119], so the dataset needs to be transformed accordingly.

In [84], the authors improved the LDoS attack detection method by developing a sailfish algorithm to enhance the parameters of the CNN-GRU characteristic. Although the model performed well compared to traditional machine learning algorithms, It requires a lot of training time and lacks real-world implementation. Furthermore, it is not intended for online attack detection in live SDNs since it must identify traffic flows that can be split across different capture time frames. Despite the limitations of the deep learning-based LDDoS detection approaches, we nevertheless recommend employing deep learning to construct an LDDoS attack detection mechanism.

7. Current Challenges and Future Research Directions

This section summarizes the current challenges for all research articles that propose approaches to detect LDDoS attacks based on machine learning and deep learning. Thus, researchers interested in this area can point out the challenges and propose efficient solutions for LDDoS attack detection.

The recent attacks, such as the one on AWS (Amazon Web Services) [120], show that the nature of the DDoS attacks is changing from the typical high-volume flooding attacks to more stealthy low-rate attacks. Since the launch of the Shrew attack [121] categorized as performing an LDDoS attack, several variants of LDDoS attacks have been proposed and analyzed. An LDDoS attack is more difficult to detect since its average traffic volume is not significantly different from that of normal traffic. However, it cannot be ignored as it impacts legitimate traffic over time.

Despite the fact that there has been a little study in the field of LDDoS attacks in SDN, the effectiveness of machine learning methods is more likely to classify and detect LDDoS attacks. This is why machine learning algorithms have outperformed manual checking methodologies when data amounts are large. Because threshold detection methods have usually depended on only a few measurements, it is simple to confuse a normal random burst in a real network for an attack. Furthermore, the outcomes of these approaches are susceptible to the detection threshold, which must be altered according to the network scene, or the correct detection probability will suffer. Several LDDoS attack detection using machine learning approaches [17,76–84] have been investigated in this paper. They bring several recent results on LDDoS attack detection. These results can summarize as follows:

7.1. Specific Datasets

There are a few publicly available datasets for traditional DDoS attacks that are not designed for SDN architecture, such as the NSL-KDD, DARPA98, CAIDA, and CIC DoS 2019 datasets. Few of the datasets that capture LDDoS attacks are available, such as CIC DoS, which is also not intended for SDN architecture. There is an urgent need to find a dataset for LDDoS attacks in the SDN architecture to adapt to the programmability of SDN controllers and the flow tables in OpenFlow switches.

7.2. Real Evaluation Instead of Simulation

In the previous analysis of methods to defend against LDDoS attacks in SDN, most methods were implemented in a simulation environment. There is a need to implement and test these methods in a realistic SDN architecture to more accurately evaluate performance and adjust network traffic depending on the topology and size of the network. Although the results obtained in simulation are acceptable for research. In certain cases, researchers use a simplified simulated SDN environment with a single controller and two hosts with limited resources. Therefore, a great effort is needed to design up-to-date and large-scale test rules that enable SDN to validate and minimize LDDoS detection in real network hardware environment as available in traditional IP networks.

7.3. Unauthenticated Application

Many applications at the application layer can access network resources for better management. Furthermore, many unauthorized applications obtain this access through instances of approved applications. An attacker 1might take advantage of this vulnerability and utilize these programs to launch LDDoS attacks, reducing network performance. As a result, there is a need to identify a reliable application authenticator to validate the application before authorizing access to network resources.

7.4. LDDoS Attacks in the Industrial Domain

Incorporating the SDN concept into the industrial environment introduces new security issues that must be addressed. The inclusion of SDN opens the door for cyberattacks on the network. LDDoS attacks are one type of potential cyberattack. The destructive impact

of an LDDoS attack on a real industrial environment and the entire manufacturing process must be demonstrated.

7.5. Controller Overhead

Due to the lack of a secure and stable SDN controller, implementing an LDDoS attack detection method based on machine learning using a single controller may overwhelm the controller, reducing its working efficiency. However, this single controller can be a point of failure for the entire network. On the other hand, the implementation of multiple controllers and distributed defense solutions can be a better choice for future defense plans as it can distribute the load between different devices and load balancing can be achieved according to the requirements. There is an urgent need from the research community to provide a light machine based on machine learning that can be implemented in multiple controller scenarios while keeping synchronization and communication to a minimum.

7.6. Large-Scale Network with More than One Controller

The increasing expansion of network capacity and controller scalability owing to computation limit is another significant difficulty. The stability and scalability of a multi-controller platform must be prioritized. The controllers in a distributed controllers scenario are conceptually central. The main controller is a root with a global view of the network, and the others are local controllers with network information only in their domain. However, synchronizing these controllers is another issue that must be addressed. Proposing an LDDoS attack method based on machine learning effective in large-scale networks with more than one controller is a challenge for the SDN community.

7.7. OpenFlow Switch Overload

In OpenFlow switches, the flow table rules are stored in TCAM. This type of memory is restricted to a limited number of flows and is expensive, which attracts the attention of LDDoS attackers. LDDoS use less resources to load the flow table than it takes to exhaust the network controller. The attacker 1attempts to load and run through its restricted capacity, causing the switch to stop working and potentially terminating network services. Although applying a detection method in the OpenFlow switches can reduce the controller's computing burden and communication between the data plane and the control plane, it increases the complexity of the hardware and incurs an additional expense. Therefore, effective deployment of security modules in switches while reducing the complexity of communication between devices is an open problem.

7.8. High Cost of Implementation

Machine learning-based LDDoS detection methods used multiple traffic engineering parameters for the current state of the network. These methods are implemented at the control level of the SDN architecture. A central controller responsible for policy-making can serve a limited number of new flow requests to manage OpenFlow switches. As a result, the controller must put forth a lot of effort to handle network traffic efficiently. However, this extra overhead by the detection methods for computation for multiple parameters can harm the controller's performance. Therefore, researchers must apply an appropriate approach to reduce the number of parameters used in LDDoS detection methods.

7.9. Feature Selection

One obvious drawback of traditional machine learning techniques is that they rely heavily on feature engineering. LDDoS attacks can disguise themselves with regular traffic, making them difficult to detect. The first problem with the current solution is that the feature selection is not straightforward. Since most of the current detection methods are associated with an LDDoS attack of the SDN controller and OpenFlow switches, the features are sampled using the entire flow table as a sample. There is an urgent need to find apparent features of LDDoS attacks disguised with flow traffic in SDN architecture.

Feature selection is quite tricky. Thus, we expect deep learning models to be proposed in the future as the feature selections can be automatic in deep learning models.

7.10. Exploiting SDN Capabilities for LDDoS Detection

SDN adds new qualities that did not exist in traditional IP networks, resulting in a distinct network design from traditional IP networks. SDN-based networks employ OpenFlow as southbound interfaces, and messages such as packet-in, packet-out, flow-mod, echo-reply, echo-request, error messages, and hello messages have the capacity to reverse network state. In other words, these messages and the features of the packets may be utilized by machine learning in the SDN to identify LDDoS attacks. However, based on a review of the relevant literature, implementing these messages features is a shortcoming in leveraging the SDN's potential for LDDoS detection.

8. Conclusions

LDDoS attack is a recent evolution of DDoS attack that is more challenging and difficult to detect. As traditional security schemes proposed for defending against DDoS are not efficient in detecting LDDoS attacks, novel security mechanisms developed for LDDoS are needed. In this paper, we presented an extensive review of several existing LDDoS detection methods for SDN. In particular, we have proposed a taxonomy of LDDoS attacks in all layers of the SDN architecture. Furthermore, integrated analysis and identification of shortcomings or disadvantages of all LDDoS detection approaches were discussed and summarized. Classification-based and deep learning mechanisms for LDDoS detection have received growing attention from researchers because of the nature of security attacks that are dynamically changing and evolving. Our survey shows that mechanisms based on deep learning and incorporating a hybrid model such as CNN-LSTM and CNN-GRU perform better than classification-based mechanisms that rely on only ML classification algorithms. CNN-LSTM and CNN-GRU provided a high detection rate, up to 99.5%, with a lower false rate than other methods. We have also demonstrated a brief survey about LDDoS attacks in IoT networks based on SDN architecture for the industrial domain. As part of future work, we will provide a more detailed issues on IoT networks based on SDN and propose detection mechanisms to protect IoT networks from LDDoS attacks using machine learning approaches. In addition, open issues are also discussed and analyzed which suggest future developments and pave the way for young researchers to develop novel approaches for detecting and defending the SDN against LDDoS attacks.

Author Contributions: Conceptualization and writing original draft preparation, A.A.A.; supervision—methodology and editing, M.S.M.Z.; writing—reviewing and editing, M.A.A.; data collection and analysis, M.Y.D.; reviewing and editing, B.I.; organizing sections of the manuscript, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by Ministry of Education Malaysia, Fundamental Research Grant No. 015MA0-047.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge the support of this research by Ministry of Education Malaysia, Fundamental Research Grant No. 015MA0-047.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AOS	Application Operator and Schedule;
AUC	Area Under the Curve;
BPNN	Backpropagation Neural Network;
CNN	Convolutional Neural Network;
CIC	Canadian Institute of Cybersecurity;
SDN	Software Defined Networking;
DoS	Denial of Services;
DDoS	Distributed Denial of Services;
LDDoS	Low-rate Distributed Denial of Services;
IoT	Internet of Things;
ML	Machine Learning;
DL	Deep Learning;
SL	Supervised Learning;
LR	Logistic Regression;
SVM	Support Vector Machine;
KNN	K-Nearest Neighbor;
UL	Unsupervised Learning;
LSTM	Long Short-Term Memory;
HTTP	Hypertext Transfer Protocol;
DNS	Distributed Denial of Service;
RNN	Recurrent Neural Network;
PCA	Principal Component Analysis;
ONF	Open Networking Foundation;
OFA	OpenFlow Agent;
TCAM	Three Content Addressable Memory;
OFMF	Open Flow Mod Failed;
IDS	Intrusion Detection System;
MDP	Markov Decision Process;
GRU	Gated Recurrent Unit;
CPU	Central Processing Unit;
ICMP	Internet Control Message Protocol;
UDP	User Datagram Protocol;
DNS	Domain Name System;
RNN	Recurrent Neural Network;
WSN	Wireless Sensor Networks.

References

1. Li, J.; Xue, K.; Liu, J.; Zhang, Y.; Fang, Y. An ICN/SDN-based network architecture and efficient content retrieval for future satellite-terrestrial integrated networks. *IEEE Netw.* **2019**, *34*, 188–195. [CrossRef]
2. Abbasi, M.R.; Guleria, A.; Devi, M.S. Traffic engineering in software defined networks: A survey. *J. Telecommun. Inf. Technol.* **2016**, *4*, 3–14.
3. Jammal, M.; Singh, T.; Shami, A.; Asal, R.; Li, Y. Software defined networking: State of the art and research challenges. *Comput. Netw.* **2014**, *72*, 74–98. [CrossRef]
4. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2014**, *103*, 14–76. [CrossRef]
5. Camacho, F.; Cárdenas, C.; Muñoz, D. Emerging technologies and research challenges for intelligent transportation systems: 5G, HetNets, and SDN. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2018**, *12*, 327–335. [CrossRef]
6. Jia, W.K. PFQDN: SDN-and DNS-Assisted Transparent Communications among Behind-NAT Networks. *IEEE Syst. J.* **2022**, *1*, 1–11. [CrossRef]
7. Dangi, R.; Jadhav, A.; Choudhary, G.; Dragoni, N.; Mishra, M.K.; Lalwani, P. ML-Based 5G Network Slicing Security: A Comprehensive Survey. *Future Internet* **2022**, *14*, 116. [CrossRef]
8. Waseem, Q.; Alshamrani, S.S.; Nisar, K.; Wan Din, W.I.S.; Alghamdi, A.S. Future Technology: Software-Defined Network (SDN) Forensic. *Symmetry* **2021**, *13*, 767. [CrossRef]

9. Alashhab, A.A.; Zahid, M.S.M.; Barka, A.A.; Albaboh, A.M. Experimenting and evaluating the impact of DoS attacks on different SDN controllers. In Proceedings of the 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA, Tripoli, Libya, 25–27 May 2021; pp. 722–727.
10. Kalkan, K.; Gur, G.; Alagoz, F. Defense mechanisms against DDoS attacks in SDN environment. *IEEE Commun. Mag.* **2017**, *55*, 175–179. [CrossRef]
11. Lei, G.; Ji, L.; Ji, R.; Cao, Y.; Shao, X.; Huang, X. Extracting low-rate DDoS attack characteristics: the case of multipath TCP-based communication networks. *Wirel. Commun. Mobile Comput.* **2021**, *2021*, 2264187. [CrossRef]
12. Shalunov, S.; Teitelbaum, B. TCP use and performance on Internet2. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop, San Francisco, CA, USA, 1–2 November 2001.
13. Cambiaso, E.; Papaleo, G.; Chiola, G.; Aiello, M. *Mobile Executions of Slow DoS Attacks*; Oxford Academic: Oxford, UK, 2016.
14. Cambiaso, E.; Papaleo, G.; Chiola, G.; Aiello, M. Designing and modeling the slow next DoS attack. In Proceedings of the Computational Intelligence in Security for Information Systems Conference, Burgos, Spain, 15–17 June 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 249–259.
15. Cui, Y.; Qian, Q.; Guo, C.; Shen, G.; Tian, Y.; Xing, H.; Yan, L. Towards DDoS detection mechanisms in software-defined networking. *J. Netw. Comput. Appl.* **2021**, *190*, 103156. [CrossRef]
16. Xingshu, C.; Qiang, H.; Yitong, W.; Long, G.; Yi, Z. Research on low-rate DDoS attack of SDN network in cloud environment. *J. Commun.* **2019**, *40*, 210.
17. Pérez-Díaz, J.A.; Valdovinos, I.A.; Choo, K.K.R.; Zhu, D. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. *IEEE Access* **2020**, *8*, 155859–155872. [CrossRef]
18. Reiswig, J. Mendeley. *J. Med. Libr. Assoc. JMLA* **2010**, *98*, 193. [CrossRef]
19. Balarezo, J.F.; Wang, S.; Chavez, K.G.; Al-Hourani, A.; Kandeepan, S. A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks. *Eng. Sci. Technol. Int. J.* **2021**, *31*, 101065. [CrossRef]
20. Dong, S.; Abbas, K.; Jain, R. A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. *IEEE Access* **2019**, *7*, 80813–80828. [CrossRef]
21. Aladaileh, M.A.; Anbar, M.; Hasbullah, I.H.; Chong, Y.W.; Sanjalawe, Y.K. Detection techniques of distributed denial of service attacks on software-defined networking controlle—A review. *IEEE Access* **2020**, *8*, 143985–143995. [CrossRef]
22. Xu, X.; Yu, H.; Yang, K. DDoS attack in software defined networks: A survey. *ZTE Commun.* **2017**, *15*, 3.
23. ur Rasool, R.; Wang, H.; Ashraf, U.; Ahmed, K.; Anwar, Z.; Rafique, W. A survey of link flooding attacks in software defined network ecosystems. *J. Netw. Comput. Appl.* **2020**, *172*, 102803. [CrossRef]
24. Wang, P.; Yang, L.T.; Nie, X.; Ren, Z.; Li, J.; Kuang, L. Data-driven software defined network attack detection: State-of-the-art and perspectives. *Inform. Sci.* **2020**, *513*, 65–83. [CrossRef]
25. Singh, J.; Behal, S. Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Comput. Sci. Rev.* **2020**, *37*, 100279. [CrossRef]
26. Mitchell, T. Does machine learning really work? *AI Mag.* **1997**, *18*, 11.
27. Swana, E.; Doorsamy, W. An Unsupervised Learning Approach to Condition Assessment on a Wound-Rotor Induction Generator. *Energies* **2021**, *14*, 602. [CrossRef]
28. Sarker, I.H. Machine learning: Algorithms, real-world applications and research directions. *SN Comput. Sci.* **2021**, *2*, 1–21. [CrossRef] [PubMed]
29. Sarker, I.H. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Comput. Sci.* **2021**, *2*, 420. [CrossRef] [PubMed]
30. Riedmiller, M. Advanced supervised learning in multi-layer perceptrons—From backpropagation to adaptive learning algorithms. *Comput. Stand. Interfaces* **1994**, *16*, 265–278. [CrossRef]
31. Thacker, N.A.; Abraham, I.; Courtney, P. Supervised learning extensions to the clam network. *Neural Netw.* **1997**, *10*, 315–326. [CrossRef]
32. Biau, G.; Scornet, E. A random forest guided tour. *Test* **2016**, *25*, 197–227. [CrossRef]
33. Schölkopf, B.; Smola, A.J.; Bach, F. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
34. Dietterich, T.G.; Kong, E.B. *Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms*; Report; Citeseer: Pennsylvania, PA, USA, 1995.
35. Dreiseitl, S.; Ohno-Machado, L. Logistic regression and artificial neural network classification models: a methodology review. *J. Biomed. Inform.* **2002**, *35*, 352–359. [CrossRef]
36. Archer, N.P.; Wang, S. Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decis. Sci.* **1993**, *24*, 60–75. [CrossRef]
37. Peterson, L.E. K-nearest neighbor. *Scholarpedia* **2009**, *4*, 1883. [CrossRef]
38. Ari, B.; Güvenir, H.A. Clustered linear regression. *Knowl.-Based Syst.* **2002**, *15*, 169–175. [CrossRef]
39. Hastie, T.; Tibshirani, R.; Friedman, J. Unsupervised learning. In *The Elements of Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 485–585.
40. El Naqa, I.; Murphy, M.J. What is machine learning? In *Machine Learning in Radiation Oncology*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–11.

41. Ralambondrainy, H. A conceptual version of the k-means algorithm. *Pattern Recognit. Lett.* **1995**, *16*, 1147–1157. [CrossRef]
42. Birant, D.; Kut, A. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data Knowl. Eng.* **2007**, *60*, 208–221. [CrossRef]
43. Wang, K.; Zhang, J.; Li, D.; Zhang, X.; Guo, T. Adaptive affinity propagation clustering. *arXiv* **2008**, arXiv:0805.1096.
44. Comaniciu, D.; Meer, P. Mean shift analysis and applications. In Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1197–1203.
45. Chapelle, O.; Scholkopf, B.; Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Trans. Neural Netw.* **2009**, *20*, 542–542. [CrossRef]
46. Di, H.; Ke, X.; Peng, Z.; Dongdong, Z. Surface defect classification of steels with a new semi-supervised learning method. *Opt. Lasers Eng.* **2019**, *117*, 40–48. [CrossRef]
47. Shinan, K.; Alsubhi, K.; Alzahrani, A.; Ashraf, M.U. Machine learning-based botnet detection in software-defined network: A systematic review. *Symmetry* **2021**, *13*, 866. [CrossRef]
48. Fazakis, N.; Kanas, V.G.; Aridas, C.K.; Karlos, S.; Kotsiantis, S. Combination of active learning and semi-supervised learning under a self-training scheme. *Entropy* **2019**, *21*, 988. [CrossRef]
49. Subramanya, A.; Talukdar, P.P. Graph-based semi-supervised learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2014**, *8*, 1–125.
50. Chapelle, O.; Zien, A. Semi-supervised classification by low density separation. In Proceedings of the International Workshop on Artificial Intelligence and Statistics (PMLR), Bridgetown, Barbados, 6–8 January 2005; pp. 57–64.
51. Chen, H.; Engkvist, O.; Wang, Y.; Olivecrona, M.; Blaschke, T. The rise of deep learning in drug discovery. *Drug Discov. Today* **2018**, *23*, 1241–1250. [CrossRef] [PubMed]
52. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; pp. 1–6.
53. Lorencin, I.; Anđelić, N.; Mrzljak, V.; Car, Z. Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation. *Energies* **2019**, *12*, 4352. [CrossRef]
54. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Sign. Process.* **1997**, *45*, 2673–2681. [CrossRef]
55. Wang, W.; Huang, Y.; Wang, Y.; Wang, L. Generalized autoencoder: A neural network framework for dimensionality reduction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 490–497.
56. Krizhevsky, A.; Hinton, G. Convolutional deep belief networks on cifar-10. *Computers* **2010**, *40*, 1–9.
57. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
58. Liao, Z.; Chen, C.; Ju, Y.; He, C.; Jiang, J.; Pei, Q. Multi-Controller Deployment in SDN-Enabled 6G Space–Air–Ground Integrated Network. *Remote Sens.* **2022**, *14*, 1076. [CrossRef]
59. Kirkpatrick, K. Software-defined networking. *Commun. ACM* **2013**, *56*, 16–19. [CrossRef]
60. Abdou, A.; Van Oorschot, P.C.; Wan, T. Comparative analysis of control plane security of SDN and conventional networks. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3542–3559. [CrossRef]
61. O.N. Fundation. *Open Networking*; O.N. Foundation: Luzern, Switzerland, 2022.
62. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
63. Kaur, K.; Kaur, S.; Gupta, V. Flow statistics based load balancing in OpenFlow. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 378–381.
64. Cheema, A.; Tariq, M.; Hafiz, A.; Khan, M.M.; Ahmad, F.; Anwar, M. Prevention Techniques against Distributed Denial of Service Attacks in Heterogeneous Networks: A Systematic Review. *Secur. Commun. Netw.* **2022**, *2022*, 8379532. [CrossRef]
65. Huraj, L.; Horak, T.; Strelec, P.; Tanuska, P. Mitigation against DDoS Attacks on an IoT-Based Production Line Using Machine Learning. *Appl. Sci.* **2021**, *11*, 1847. [CrossRef]
66. Wang, S.; Gomez, K.; Sithamparanathan, K.; Asghar, M.R.; Russello, G.; Zanna, P. Mitigating DDoS Attacks in SDN-Based IoT Networks Leveraging Secure Control and Data Plane Algorithm. *Appl. Sci.* **2021**, *11*, 929. [CrossRef]
67. Horak, T.; Cervenanska, Z.; Huraj, L.; Vazan, P.; Janosik, J.; Tanuska, P. The vulnerability of securing IoT production lines and their network components in the Industry 4.0 concept. *IFAC-Pap. Online* **2020**, *53*, 11237–11242. [CrossRef]
68. Šimon, M.; Huraj, L.; Horák, T. DDoS reflection attack based on IoT: A case study. In Proceedings of the Computer Science Online Conference, Las Vegas, NV, USA, 12–14 December 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 44–52.
69. Bawany, N.Z.; Shamsi, J.A.; Salah, K. DDoS attack detection and mitigation using SDN: Methods, practices, and solutions. *Arab. J. Sci. Eng.* **2017**, *42*, 425–441. [CrossRef]
70. Wang, B.; Su, J. FlexMonitor: A flexible monitoring framework in SDN. *Symmetry* **2018**, *10*, 713. [CrossRef]
71. Yang, Y.S.; Lee, S.H.; Chen, W.C.; Yang, C.S.; Huang, Y.M.; Hou, T.W. Securing SCADA Energy Management System under DDos attacks using token verification approach. *Appl. Sci.* **2022**, *12*, 530. [CrossRef]
72. Zargar, S.T.; Joshi, J.; Tipper, D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]
73. Xiang, Y.; Li, K.; Zhou, W. Low-rate DDoS attacks detection and traceback by using new information metrics. *IEEE Trans. Inform. Forens. Secur.* **2011**, *6*, 426–437. [CrossRef]
74. Zhijun, W.; Wenjing, L.; Liang, L.; Meng, Y. Low-rate DoS attacks, detection, defense, and challenges: A survey. *IEEE Access* **2020**, *8*, 43920–43943. [CrossRef]

75. Bhuyan, M.H.; Bhattacharyya, D.; Kalita, J.K. An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. *Pattern Recogn. Lett.* **2015**, *51*, 1–7. [CrossRef]
76. Phan, T.V.; Gias, T.R.; Islam, S.T.; Huong, T.T.; Thanh, N.H.; Bauschert, T. Q-MIND: Defeating stealthy DoS attacks in SDN with a machine-learning based defense framework. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
77. Khamkar, R.; Thakre, K.; Kotkar, A.; Jadhav, P.; Hanchate, R. Low rate DDoS Attack Identification and Defense using SDN based on Machine Learning Method. *Int. Res. J. Eng. Technol. (IRJET)* **2021**, *8*, 423.
78. Cheng, H.; Liu, J.; Xu, T.; Ren, B.; Mao, J.; Zhang, W. Machine learning based low-rate DDoS attack detection for SDN enabled IoT networks. *Int. J. Sens. Netw.* **2020**, *34*, 56–69. [CrossRef]
79. Yin, W.; Cui, Y.; Qian, Q.; Shen, G.; Guo, C.; Li, S. DIAMOND: A Structured Coevolution Feature Optimization Method for LDDoS Detection in SDN-IoT. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 9530274. [CrossRef]
80. Sudar, K.M.; Deepalakshmi, P. Flow-Based Detection and Mitigation of Low-Rate DDOS Attack in SDN Environment Using Machine Learning Techniques. In *IoT and Analytics for Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 193–205.
81. Zhijun, W.; Qing, X.; Jingjie, W.; Meng, Y.; Liang, L. Low-rate DDoS attack detection based on factorization machine in software defined network. *IEEE Access* **2020**, *8*, 17404–17418. [CrossRef]
82. Nugraha, B.; Murthy, R.N. Deep learning-based slow DDoS attack detection in SDN-based networks. In Proceedings of the 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Madrid, Spain, 10–12 November 2020; pp. 51–56.
83. Tang, D.; Yan, Y.; Zhang, S.; Chen, J.; Qin, Z. Performance and features: mitigating the low-rate TCP-targeted DoS attack via SDN. *IEEE J. Select.Areas Commun.* **2021**, *40*, 428–444. [CrossRef]
84. Sun, W.; Guan, S.; Wang, P.; Wu, Q. A hybrid deep learning model based low-rate DoS attack detection method for software defined network. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e4443. [CrossRef]
85. Apostolovic, T.; Stankovic, N.; Milenkovic, K.; Stanisavljevic, Z. DDoSSim-System for Visual Representation of the Selected Distributed Denial of Service Attacks. In Proceedings of the 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 30–31 May 2018; pp. 118–122.
86. jseidl.GoldenEye. Available online: https://www.kali.org/tools/goldeneye/ (accessed on 17 February 2022).
87. HULK. Mr4FX. Available online: https://allabouttesting.org/hulk-ddos-tool-complete-installation-usage-with-examples/ (accessed on 17 February 2022).
88. Cloudflare. *Slowloris DDoS Attack*; Cloudflare: San Francisco, CA, USA, 2021.
89. Cloudflare. 2020. Available online: https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/ (accessed on 17 February 2022).
90. Bhushan, K.; Gupta, B.B. Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *J. Ambient Intell. Human. Comput.* **2019**, *10*, 1985–1997. [CrossRef]
91. Ubale, T.; Jain, A.K. Survey on DDoS attack techniques and solutions in software-defined network. In *Handbook of Computer Networks and Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 389–419.
92. Singh, M.P.; Bhandari, A. New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges. *Comput. Commun.* **2020**, *154*, 509–527. [CrossRef]
93. Pashkov, V.; Shalimov, A.; Smeliansky, R. Controller failover for SDN enterprise networks. In Proceedings of the 2014 International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), Moscow, Russia, 28–29 October 2014; pp. 1–6.
94. Muthamil Sudar, K.; Deepalakshmi, P. A two level security mechanism to detect a DDoS flooding attack in software-defined networks using entropy-based and C4. 5 technique. *J. High Speed Netw.* **2020**, *26*, 55–76. [CrossRef]
95. Daha, M.Y.; Zahid, M.S.M.; Husain, K.; Ousta, F. Performance Evaluation of Software Defined Networks with Single and Multiple Link Failure Scenario under Floodlight Controller. In Proceedings of the 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 19–20 February 2021; pp. 959–965.
96. Daha, M.Y.; Zahid, M.S.M.; Isyaku, B.; Alashhab, A.A. CDRA: A Community Detection based Routing Algorithm for Link Failure Recovery in Software Defined Networks. *(IJACSA) Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 11. [CrossRef]
97. Chen, K.y.; Junuthula, A.R.; Siddhrau, I.K.; Xu, Y.; Chao, H.J. SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane. In Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, USA, 17–19 October 2016; pp. 28–36.
98. Benamrane, F.; Benaini, R. An East-West interface for distributed SDN control plane: Implementation and evaluation. *Comput. Electr. Eng.* **2017**, *57*, 162–175. [CrossRef]
99. BN, Y. Preemptive modelling towards classifying vulnerability of DDoS attack in SDN environment. *Int. Electr. Comput. Eng.* **2020**, *10*, 8708.
100. He, C.H.; Chang, B.Y.; Chakraborty, S.; Chen, C.; Wang, L.C. A zero flow entry expiration timeout p4 switch. In Proceedings of the Symposium on SDN Research, Los Angeles, CA, USA, 28–29 March 2018; pp. 1–2.
101. Kandoi, R.; Antikainen, M. Denial-of-service attacks in OpenFlow SDN networks. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 1322–1326.
102. Isyaku, B.; Mohd Zahid, M.S.; Bte Kamat, M.; Abu Bakar, K.; Ghaleb, F.A. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey. *Future Internet* **2020**, *12*, 147. [CrossRef]

103. You, X.; Feng, Y.; Sakurai, K. Packet in message based DDoS attack detection in SDN network using OpenFlow. In Proceedings of the 2017 Fifth International Symposium on Computing and Networking (CANDAR), Aomori, Japan, 19–22 November 2017; pp. 522–528.
104. Pascoal, T.A.; Dantas, Y.G.; Fonseca, I.E.; Nigam, V. Slow TCAM exhaustion DDoS attack. In Proceedings of the IFIP International Conference on ICT Systems Security and Privacy Protection, Rome, Italy, 29–31 May 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 17–31.
105. Dehkordi, A.B.; Soltanaghaei, M.; Boroujeni, F.Z. The DDoS attacks detection through machine learning and statistical methods in SDN. *J. Supercomput.* **2021**, *77*, 2383–2415. [CrossRef]
106. Bowers, A.J.; Zhou, X. Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes. *J. Educ. Stud. Placed Risk (JESPAR)* **2019**, *24*, 20–46. [CrossRef]
107. Andrew, A.M. Reinforcement Learning: An Introduction by Richard S. Sutton and Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning Series; MIT Press (Bradford Book), Cambridge, Mass., 1998, pp. 58..322, ISBN 0-262-19398-1. *Robotica* **1999**, *17*, 229–235. [CrossRef]
108. De Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using mininet for emulation and prototyping software-defined networks. In Proceedings of the 2014 IEEE Colombian conference on communications and computing (COLCOM), Bogota, Colombia, 2–4 June 2014; pp. 1–6.
109. Wette, P.; Dräxler, M.; Schwabe, A.; Wallaschek, F.; Zahraee, M.H.; Karl, H. Maxinet: Distributed emulation of software-defined networks. In Proceedings of the 2014 IFIP Networking Conference, Trondheim, Norway, 2–4 June 2014; pp. 1–9.
110. Open Network. Available online: https://opennetworking.org/onos/ (accessed on 23 March 2022).
111. Shekyan. Available online: https://www.kali.org/tools/slowhttptest/ (accessed on 23 March 2022).
112. Cup. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 23 March 2022).
113. Deepalakshmi, P. DServ-LB: Dynamic server load balancing algorithm. *Int. J. Commun. Syst.* **2018**, *1*, 3840.
114. Scapy. Available online: https://scapy.net/ (accessed on 23 March 2022).
115. Lin, Y.z.; Nie, Z.h.; Ma, H.w. Structural damage detection with automatic feature-extraction through deep learning. *Comput.-Aided Civil Infrastruct. Eng.* **2017**, *32*, 1025–1046. [CrossRef]
116. Taud, H.; Mas, J. Multilayer perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 451–455.
117. Mahadevan, S.; Shah, S.L. Fault detection and diagnosis in process data using one-class support vector machines. *J. Process Control* **2009**, *19*, 1627–1639. [CrossRef]
118. Andersson, O.O. Available online: https://github.com/Ogglas/Orignal-Slowloris-HTTP-DoS (accessed on 15 January 2022)
119. Roopak, M.; Tian, G.Y.; Chambers, J. Deep learning models for cyber security in IoT networks. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 452–457.
120. Khooi, X.Z.; Csikor, L.; Kang, M.S.; Divakaran, D.M. In-Network Defense against AR-DDoS Attacks. In Proceedings of the SIGCOMM'20 Poster and Demo Sessions, Online, 10–14 August 2020.
121. Kuzmanovic, A.; Knightly, E.W. Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants. In Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003; pp. 75–86.