symmetry

MDPI

Article

Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches

Hsiu-Min Chuang [1,2,*], Fanpyn Liu [1,2] and Chung-Hsien Tsai [1,2]

[1] Department of Computer Science and Information Engineering, Chung Cheng Institute of Technology, National Defense University, Taoyuan City 335, Taiwan; fanpyn.liu@ccit.ndu.edu.tw (F.L.); chunghsien.tsai@ccit.ndu.edu.tw (C.-H.T.)
[2] System Engineering and Technology Program, National Yang Ming Chiao Tung University, Hsinchu City 30010, Taiwan
* Correspondence: showmin1205@gmail.com or showmin@ccit.ndu.edu.tw

Abstract: Recent developments have made software-defined networking (SDN) a popular technology for solving the inherent problems of conventional distributed networks. The key benefit of SDN is the decoupling between the control plane and the data plane, which makes the network more flexible and easier to manage. SDN is a new generation network architecture; however, its configuration settings are centralized, making it vulnerable to hackers. Our study investigated the feasibility of applying artificial intelligence technology to detect abnormal attacks in an SDN environment based on the current unit network architecture; therefore, the concept of symmetry includes the sustainability of SDN applications and robust performance of machine learning (ML) models in the case of various malicious attacks. In this study, we focus on the early detection of abnormal attacks in an SDN environment. On detection of malicious traffic in SDN topology, the AI module in the topology is applied to detect and act against the attack source through machine learning algorithms, making the network architecture more flexible. Under multiple abnormal attacks, we propose a hierarchical multi-class (HMC) architecture to effectively address the imbalanced dataset problem and improve the performance of minority classes. The experimental results show that the decision tree, random forest, bagging, AdaBoost, and deep learning models exhibit the best performance for distributed denial-of-service (DDoS) attacks. In addition, for the imbalanced dataset problem of multiclass classification, our proposed HMC architecture performs better than previous single classifiers. We also simulated the SDN topology and scenario verification. In summary, we concatenated the AI module to enhance the security and effectiveness of SDN networks in a practical manner.

Keywords: machine learning; multiclass classification; SDN; abnormal detection; imbalance dataset

Citation: Chuang, H.-M.; Liu, F.; Tsai, C.-H. Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches. *Symmetry* **2022**, *14*, 1178. https://doi.org/10.3390/sym14061178

Academic Editors: Haipeng Cai and Jia-Ning Luo

Received: 4 May 2022
Accepted: 6 June 2022
Published: 8 June 2022

1. Introduction

With the rapid development of network semiconductors and software technology, a new generation of network technology has focused on developing software-defined networking (SDN) and network function virtualization (NFV) technology. The relevant market output value in 2020 was estimated to be approximately USD 13.7 billion and is expected to reach more than USD 32.7 billion by 2025, of which, the one critical technology is network orchestration [1] (https://www.marketsandmarkets.com/Market-Reports/software-defined-networking-sdn-market-655.html, accessed on 28 October 2021).

Under the new SDN paradigm, the entire system controller can be controlled using a centralized remote device. Advantages of SDN ave motivated many industrial and commercial companies to deploy it in their networking environments [2], including (1) separating the control plane from the data plane and facilitating network system management; (2) enabling IT administrators to easily deploy network equipment or upgrade network infrastructure without restrictions imposed by vendors; (3) providing a global view of the entire network with SDN controllers; (4) deploying various layers of SDN systems to

implement network service virtual environments [3]; (5) no programming language for the underlying infrastructure equipment. Consequently, the operating costs are significantly lower than those of traditional networks.

Most existing administrative units still use traditional distributed network control and network management structures, which cannot cope with emergencies or new network crises. Because SDN involves centralized network management by the controller, it aims to separate network control and data forwarding functions; however, because of this, the network traffic data of the controller are the primary attack target of network hackers and are also an important indicator for an intrusion detection system (IDS) [4] to identify malicious traffic. Traditional network control centers are slow in device setup and threat prediction, making it challenging to satisfy the security requirements of new types of networks; therefore, our goal is to effectively solve the problems existing in current IDS' and propose an SDN architecture that can deeply analyze network threats.

The concept of symmetry in this study includes the sustainability of SDN applications and the robust performance of machine learning (ML) models in managing various malicious attacks. The SDN architecture enables centralized management and communication between controllers and OpenFlow switches for the former. This reduces the overhead of controller–switch communication for a dynamic and programmatically efficient network configuration. For the latter, owing to ever-changing attacks and potential threats, the ML model must be able to identify abnormal traffic and maintain its robustness; therefore, a specific detection performance must be achieved under different attacks and models.

Recent studies indicated that ML, deep learning (DL), and other algorithms [5–12] were used in SDN environments to predict anomalies and improve decision-making capabilities. As deployment challenges in SDN environments arise from vulnerabilities and threats, IDS monitoring of malicious activity has become a critical measure in network architectures. In addition, the centralized view of SDN offers new practical possibilities. Among the datasets used in recent years, the detection of DDoS attacks is the most common example to verify the performance of AI methods; however, prediction performance depends on the quality of the training dataset. In the past, researchers accessed public datasets such as KDD'99 [13] and NSL-KDD [14]; however, these datasets may be incompatible and outdated. Thus, we use two types of SDN datasets generated based on the SDN simulated environment [2,15] to predict network traffic and analyze the best features and model parameters to train the model resembling actual environment. The prediction results and feasible solutions are fed back as SDN parameters that effectively improve the security of the SDN architecture.

We used two SDN public datasets to generate trained models, extract important features, and predict packets on the SDN controller. These are generally divided into two categories: normal and suspicious. Normal packets are sent directly to the destination; suspicious packets are further identified and classified depending on the attack. For the public InSDN dataset [2], there were seven abnormal attacks and normal packets; however, owing to the imbalanced number of classes, which is a challenge for traditional multiclass classifiers, we propose a hierarchical multi-class (HMC) architecture to improve the performance of imbalanced datasets.

The experimental results show the model prediction performance under binary and multiclass classification compared with eight machine learning and five DL models. We found that decision trees (DT), random forests (RF), bagging, and AdaBoost performed the best among the machine learning models, although DL models were also good. In terms of multiclass classification, the performance of the above single multiclass classifier is extremely poor owing to the small amount of training data for the minority classes, such as brute-force attack (BFA), botnet, user-to-root (U2R), and web-attack. We observed a significant improvement in the HMC architecture for multiclass classification.

Finally, we simulated the DDoS attack scenario to verify the network traffic control mechanism under the prediction of the above models. Under the SDN architecture, when the AI module detects abnormal traffic, it alleviates network traffic and provides early detection to maintain SDN environment security.

The contributions of this paper can be summarized in three points.

- We proposed a preplanned cloud service based on the network characteristics of the unit and built an SDN topology architecture based on AI-assisted security prediction.
- We adopted public SDN datasets for training 13 ML and DL models to detect abnormal attacks in an SDN environment.
- We designed the HMC architecture to further identify the attack classes and improve multiclass classification in stratification. The architecture is based on a divide-and-conquer strategy to improve the poor performance of minority classes in imbalanced datasets.
- We evaluated a simulation scenario for security verification when the SDN architecture encountered DDoS attacks, and the mechanism involved early detection and traffic mitigation.

The remainder of this paper is organized as follows. Section 2 describes the related technologies. Section 3 introduces the preplanned SDN architecture and ML models. Section 4 presents the experimental results and evaluations. Finally, Section 5 summarizes the tasks and discusses future research directions.

2. Related Work

2.1. Challenges and Security Concerns of SDN

According to the Cisco Global Networking Trends report in 2020, five technologies will lead the next generation of networking: automation, artificial intelligence, multi-cloud networking, wireless, and cybersecurity. Multi-cloud networking is based on SDN and NFV architectures (https://www.cisco.com/c/dam/m/zh_tw/solutions/enterprise-networks/networking-report/files/Cisco_BlockBuster_2020-Global-Networking-Trends-Report_ZHTW.pdf?ccid=cc001244&oid=rpten018612, accessed on 28 October 2021). The SDN architecture uses software to implement a virtualized function and separates hardware from the management layer emphasizing centralization and programmability for network management. In addition, the SDN controller controls all switching equipment and network architectures performing complex network configuration work [16]. The advantage and core of SDN is its programmable controller with characteristics of automatic and adaptive network management. Among the most critical tasks of the SDN controller is link discovery because it provides the controller with the network topology necessary to direct or create rule-forwarding and routing mechanisms.

For example, Bedhief et al. [17] proposed a topology discovery mechanism in distributed controllers; however, distributed controllers have problems of data consistency and synchronization between controllers. Hence, Ochoa-Aday et al. [18] proposed an innovative protocol that enables Layer 2 discovery through switches, deriving the shortest path to each switch while preventing link failures in server environments and building redundant links; however, the Layer 2 discovery mechanism has delay and loop problems with limited reliability in network fault detection.

In addition, defining an appropriate threshold is a trade-off. Gyllstrom et al. [19] proposed a fault detection method based on an output packet counter mechanism. First, the flow rules installed on the link were marked and monitored. Packets were then counted at the target position. The error rate is calculated from the difference between the sent and received packets and is compared with the threshold value. If the error rate exceeds the threshold, the link is considered faulty; however, if the number of packets only detects errors, the reliability of network failure detection will be poor. To quickly detect a failed link and adapt the traffic to a normal link to improve the efficiency of recovery from link failure, a failed link must be detected before the recovery process; therefore, the SDN architecture must be scalable and adaptable to dynamic conditions to effectively improve network reliability with minimal manual configuration and management.

Although the SDN architecture, which is divided into application, control, and data layers, each layer can become a target of attack owing to its independent characteristics. Security challenges in SDN include targeting the controller through programming vul-

nerabilities, error configurations, and DDoS attacks on the secure channel. For example, most applications in the application layer employ third-party applications. Because of poor-quality programming, hackers can invade, control, and tamper with the program content, affecting the overall network operation. The controller in the control layer singly manages all flows in the topology. If the number of new flows exceeds the threshold, the controller can quickly become the bottleneck of the network, and the controller will be the target of the hacker. DoS and DDoS attacks are common network threats in SDN and traditional network architectures [20].

DDoS attackers send SYN packets to exhaust victim resources such as SYN, UDP, ICMP, and LAND attacks [21]; therefore, some researchers have suggested multi-controller architecture to defend against DoS and DDoS attacks in an SDN environment; however, this architecture can cause cascading failures in all the controllers [22]. The network equipment in the data layer is responsible only for packet forwarding; therefore, identifying true or false flow rules in the data layer is critical. Unfortunately, the space for storing the flow table is limited and is prone to saturation attacks. Furthermore, whenever the controller is threatened, the data layer devices will be damaged and become inoperable once the controller fails or the link is cut off [23].

As shown in Figure 1, when a flow does not conform to the table rules, the device stores some of the flow packets in the buffer. The remaining packet-in packets are sent to the controller through the interface with the control layer, and the controller then sends the flow rule to the data layer. The device implements forwarding or discarding operations; however, an attacker can send many packets to the data layer device to saturate the storage buffer and memory capacity of the flow table causing saturation [24]; therefore, a large number of packets can cause threats, such as API blocking, flood attacks [25], and controller saturation [26], reducing the overall SDN network performance. In addition to using the transmission mechanism and storage space limitations to launch DDoS attacks, they may also impersonate a device with the same IP and name it as a legitimate switch. When the legitimate switch establishes a connection with the controller, the impersonated switch is turned on and establishes a connection [27]. The controller then cuts off the connection with the legitimate switch and communicates with fake control or uses malicious programs to perform unauthorized access [28].

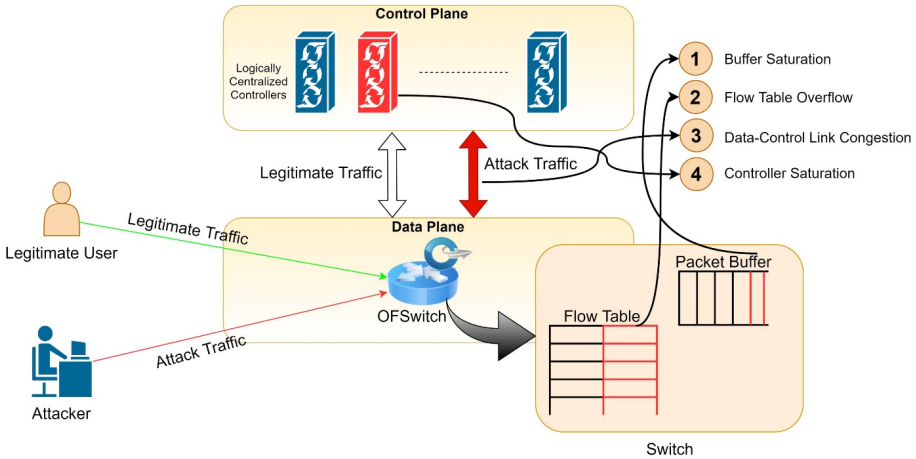

Figure 1. Vulnerabilities in SDN exploited by DDoS [28]. Reprinted/adapted with permission from ref. [28]. 2022, Elsevier.

2.2. Machine Learning in SDN

Artificial intelligence and data mining techniques have been mentioned in recent research [29–33] to enhance the optimization and decision-making capabilities in SDN environments. ML, meta-heuristics, and fuzzy inference systems were the most common approaches for solving various networking-related problems [34–36]. Thus, in our study, we focus on the high identification and quantification efficiency in SDN, so we employ

supervised learning to rapidly improve the model accuracy using labeled data, which is the best and most efficient solution. For example, Latah and Toker [37] surveyed AI applications to the SDN paradigm to provide computer networks with the ability to be programmed based on the separation between the control and forwarding planes. Moreover, hybrid intelligent techniques can be the core for achieving advanced behavior in SDN-based networks.

The challenges of SDN environment deployment arise from vulnerabilities and threats, so an IDS that monitors malicious activity is a critical measure in the network architecture. Although a centralized model of SDN can facilitate new practical applications, the detection of DDoS attacks is used to test the effectiveness of AI approaches based on datasets used in related research. The model prediction accuracy is directly related to the training dataset. Most public datasets collected from traditional networks are used to predict anomaly detection in SDN; however, public datasets such as KDD'99 [13] and NSL-KDD [14] are incompatible and outdated. Elsayed et al. released the InSDN dataset [2] in 2020 and carried out multi-class classification for seven abnormal attacks, including DDoS, DoS, password-guessing attack (PGA), web attack, botnet, probe, user to root (U2R), and normal packets. We evaluated eight common ML algorithms: decision trees, random forests, AdaBoost, KNN, naive Bayes, support vector machine with linear kernel, radial basis function (RBF), and multilayer perceptron (MLP).

For feature selection on datasets, Elsayed et al. [2] obtained eight features, including protocol type, link length, and a number of packets from the SDN network. In addition, the SDN controller based on the OpenFlow platform can calculate five statistical characteristics: network flow duration, number of packets, number of bytes, features in specific directions, and data distribution (for example, maximum, minimum, mean, and standard deviation). These statistical characteristics are essential for botnets to be particularly effective at identifying such attacks. There were 79 features in the InSDN dataset, and Elsayed et al. selected 48 features for model training. Referring to Krishnan's experiments, they added source IP and target IP features to a total of 50 features.

Recently, some studies that have adopted machine learning are applied to develop SDN-enabling security systems [38–41]. The deployment of SDN controllers makes them a key target for various attacks and vulnerabilities, such as link flood attacks (LFA). For example, Rasool et al. proposed the CyberPulse approach [42]. The method adopts machine learning data from UCI LFA attacks. Taking the burst header packet as an example, they selected 14 features and used artificial neural networks (ANN) and MLP models for binary classification prediction. The SDN network topology was simulated and evaluated using Mininet and Floodlight controllers. After the SDN environment was started, the simulated attack traffic increased. When identified as a traffic attack by the AI model, appropriate measures were taken to mitigate the traffic; otherwise, it functioned normally. Tseng et al. proposed another related study that used ML to detect DDoS attacks in SDN/NFV environments [43]. This method is similar to the study by Rasool et al. [42], mainly for TCP flooding, ICMP flooding, and UDP flooding, which are usually used for network environment simulation, anomaly detection, and system defense. The results show that MLP achieves the best performance of 99.1% F1-score with binary prediction and a training time of 5.5 s.

Aslam et al. [8] proposed an adaptive machine learning framework based on SDN to detect and mitigate distributed denial-of-service (DDoS) attacks. The framework uses a ML classification model to detect and mitigate DDoS attacks using an SDN-assisted security mechanism for IoT devices. Tonkal et al. [44] applied a ML algorithm equipped with neighborhood component analysis (NCA) to classify SDN traffic as normal or attack-related. The dataset is adapted from the public dataset "DDoS Attack SDN Dataset" for binary classification. In addition to the NCA algorithm, Tonkal et al. used decision trees, ANN, and SVM for classification. The experimental results indicated that the decision tree had greater accuracy (100 %) than that of other algorithms. Multiclass classification in an imbalanced dataset is also challenging for anomaly detection in IDS'. Toupas et al. [45]

proposed a neural network comprising multiple stacked fully connected layers. They used the updated CICIDS2017 dataset for training and evaluation to implement a flow-based anomaly-detection IDS for multiclass classification. Experiments showed that the MLP model achieves promising precision, recall, and false positives.

In summary, with the development of AI technology, related research applied to solve security issues in the SDN environment, such as anomaly detection and conflict flow detection, has gradually increased in recent years. Most study uses the existing ML or DL models to meet this challenge. The purpose is to combine AI to provide early detection in the SDN environment, so we organize and compare some related work from 2018 to 2022, as shown in Table 1. We found that some studies only analyze the classification performance of anomaly detection. Recent studies have discussed scenario validation of the overall system architecture of imported AI models and analysis of the results. We summarize the problem, datasets, models, SDN topology, and scenario verification of related work as follows.

Table 1. Comparison of related work in SDN network. (In the model field, bold represents the best model.)

Reference	Problem	Network	Dataset	Model	Topology	Verify
Yu [46], 2018	Abnormal detection	Mininet simulated	CAIDA, DDoS	SVM	✓	✓
Khamaiseh [32], 2019	Abnormal detection	Mininet	Physical & simulated	SVM, NB, **KNN**	✕	✓
Rasool [42], 2019	Abnormal detection	Mininet simulated	UCI	MLP, RF, **SLR**, NB	✓	✓
Elsayed [2], 2020	Abnormal detection	Mininet simulated	InSDN	KNN, **Adaboost**, **DT**, **RF**, rbf-SVM, lin-SVM, MLP, NB	✓	✓
Kuranage [31], 2020	Flow detection	Mininet simulated	Kaggle	**SVM**, DT, RF, KNN	✕	✕
Hamdan [30], 2020	Flow detection	Mininet simulated	MAWI, UNI1&2	**VFDT**, EDMAR, FlowSeer, BayesNet	✕	✓
Huseyin [47], 2020	Abnormal detection	Docker simulated	Simulated generation	SVM, **KNN**, ANN, NB	✓	✕
Khairi [29], 2021	Flow detection	Mininet simulated	Simulated generation [48]	DT, SVM, DT-SVM, **EFDT**	✓	✓
Aslam [8], 2022	Abnormal detection	Mininet real-time	AMLSDM	SVM, NB, RF KNN, LR, **EV**	✓	✓
Fan [7], 2022	Abnormal detection	Mininet simulated	Simulated generation	Entropy variants	✓	✕
Maheshwari [9], 2022	Abnormal detection	Mininet simulated	CICDDoS2019	SVM, RF, GBM, **MVE**	✓	✓
Liu [10], 2022	Abnormal detection	Mininet simulated	CICIDS2017	RF, SVM, **CNN**, DNN, PSO-BPNN	✓	✓
Our study	Abnormal detection	EstiNet simulated	DDoS-SDN, InSDN	**HMC**, KNN, NB, **DT**, **RF**, **Adaboost**, Bagging, rbf-SVM, lin-SVM, MLP, CNN, RNN, LSTM, GRU	✓	✓

1. **Problem**: Many studies only focus on the qualitative detection (i.e., normal and abnormal); some studies further conduct quantitative analysis, that is, the performances of multiclass classification, such as DoS, DDoS, PGA, botnet, web attack, probe, U2R, etc. In addition, they discussed further the performance improvement of minority classes under imbalanced datasets. In anomaly detection, selecting relevant features is also a critical research issue; therefore, there are also studies on extracting the features of the SDN environment for anomaly detection. This study compared and validated the above three classification and feature selection problems.
2. **Datasets**: Most of the early studies used KDD'99 and NSL-KDD as training data sets; however, using these datasets for anomaly detection in SDN environments may have issues of incompatibility and obsolescence; therefore, researchers rarely used the above datasets in recent years but generated or adopted existing public SDN datasets through simulators. We adopted two datasets derived from DDoS-SDN and InSDN that were published in 2020. Due to related work with different datasets and attack types, it is difficult to evaluate performances directly; however, using these datasets for anomaly detection in SDN environments may have issues of incompatibility and obsolescence; therefore, researchers rarely used the above datasets in recent years but generated or adopted existing public SDN datasets through simulators. We adopted two datasets derived from DDoS-SDN and InSDN that were published in 2020. Due to related work with different datasets and attack types, it is difficult to evaluate performances directly.
3. **Models**: Most studies used the current common ML models for performance comparison. In our research, we not only compared the existing ML and DL models but also proposed our HMC model. The core model is based on the best of the above single models and performs multiple binary classifications to improve the performance of multiclass classification under the imbalanced dataset.
4. **SDN topology and scenario verification**: In this study, we plan the cloud service system architecture according to practical needs and characteristics, including private cloud customized services, general services, and data centers. To facilitate the setting up of the SDN simulation environment in advance, the details of controllers, switch connections, interfaces, and MACs are described in correspondence with practical equipment. Moreover, after the ML and DL models have been trained, a DDoS attack scenario is used to verify the effectiveness of the architecture design and model prediction for the actual future environment.
5. **Symmetry**: In recent related work [49,50], learning-based anomaly detection faces two main challenges: the robustness and the sustainability of the learning model based on the SDN network architecture against ever-changing network attacks. In this study, the sustainable simulation of SDN network architecture and the robustness of experimental results can symmetrically face the challenges in SDN networks and provide specific solutions. The cloud service system we plan adopted an SDN topology design, which provides a general architecture for future sustainable development. Moreover, our proposed HMC architecture for anomaly detection can use any supervised learning models to combine them for the robustness of the architecture to overcome the poor performance of minority classes due to imbalanced datasets in real network environments. From the perspective of symmetric performance evaluation, we perform coarse-grained anomaly detection and further analyze fine-grained classification to predict the multiple types of anomaly attacks.
6. **Sustainability**: Machine learning has been successfully applied to studying malware classification over the past several years; however, the distribution of test samples with a new form of malware becomes increasingly different from the original training sample as concept drift occurs. In addition, the malware authors may modify their attack methods to avoid detection; therefore, the classifier uses the inherent training materials to encounter inefficiency, making the prediction results unreliable. There are two ways to solve the problem of concept drift. One is to develop more powerful

functions by developing more powerful functions [51] and, for example, using neural networks because potential feature space is better generalized. Despite this, the diversity of malware makes designing such a feature space extremely challenging. Another solution is to adapt to drift, use incremental retraining or online update model learning or reject drift points [52,53]. A key issue is when and how to refuse accurate tests and quantitative drift; therefore, in the topic of sustainability, in addition to maintaining accurate prediction capabilities on multiple training datasets, a key issue is to have the capability to identify aging classification models. The concept differs from retraining methods based on various datasets. For example, Jordaney et al. [54] proposed the Transced framework to identify concept drift to establish prediction indicators. Barbero et al. [55] based on the former framework for performing rejection classification, has improved efficiency and reduced computing expenses. For an ML-based classifier to be highly sustainable, it is critical to understand the underlying features: the ability to distinguish benign applications from malware and extract the changing pattern of those features through evolutionary processes [56]. In this study, facing the deterioration of anomalous attacks, we adopt multiple hierarchical classifiers to overcome the bias or inefficiency of a single classifier.

3. System Architecture

The system architecture plans to implement SDN-designed cloud services. Owing to the inherent properties of SDN, AI-assisted modules are designed to detect abnormal attacks early. Furthermore, we propose a hierarchical ML architecture for multiclass classification to identify the types of attacks and act against them.

3.1. SDN Topology Design

According to the needs and characteristics of the unit, we have planned a cloud service system architecture, as shown in Figure 2, in which AC, BC, and CC provide customized services for private clouds, and DC provides general services and data centers. Information between domains is connected through switches (red dotted lines), and each controller is responsible for the topology and packet forwarding rules in the domain, as shown in the control plane (green dotted lines) and data plane (blue lines). In each network domain, switches provide link information to the controllers. When no entry information is found in the flow table, the switch requests the controller for packet-forwarding rules. The cloud service system in this study was verified using an SDN design and implementation. The system topology is illustrated in Figure 3.

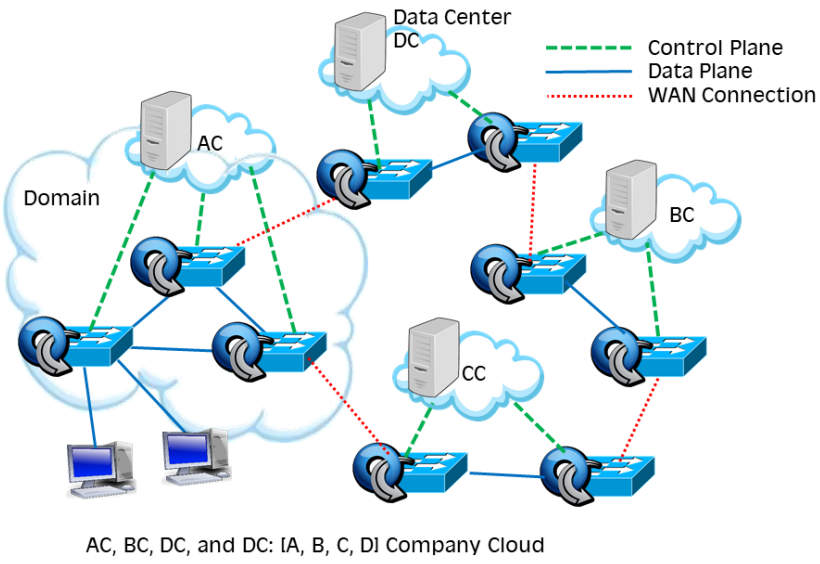

Figure 2. Cloud service architecture.

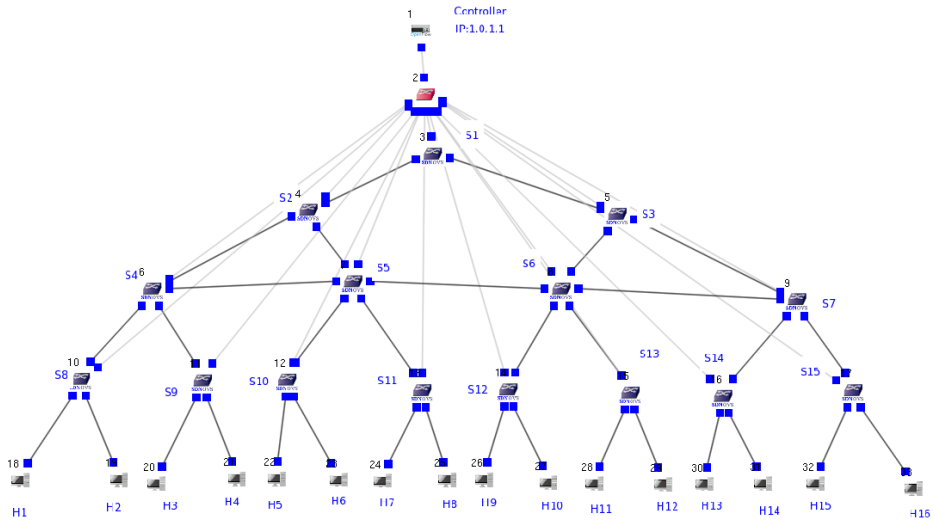

Figure 3. Topology architecture.

The main components of this topology include the Ryu OpenFlow controller and OpenFlow switches (see S1–S15 in Figure 3). We assumed that the switch from S8 to S15 is connected to two hosts, denoted by H1 to H16. Enabling the MAC address-learning function on the controller avoids network loops. The configuration settings of each device are listed in Table 2, which records the device configuration (including the interface and MAC address) of the 33 nodes. The switch in red in Figure 3 is the control plane switch designed by the EstiNet simulation software, which connects the interface between the OpenFlow controller and OpenFlow switch. The device is not part of the OpenFlow architecture because the OpenFlow switch is a L2 device, so it cannot be configured with IP addresses. To communicate with the controller, EstiNet adds an OpenFlow controller switch device to the GUI to provide an interface. The switch can be connected to this OpenFlow controller switch by an IP address, allowing the controller to control any specified OpenFlow switch and communicate with it properly.

Table 2. Configuration setting of devices in Figure 3.

ID	Device	Interface	MAC	IPv4	Connected Device
1	Controller	eth0	00:01:00:00:00:01	1.0.1.1	Controller plane switch
2	Controller plane switch	eth0	00:01:00:00:00:02	-	Controller
		eth1	00:01:00:00:00:03	-	S1
		eth2	00:01:00:00:00:10	-	S2
		eth3	00:01:00:00:00:14	-	S4
		eth4	00:01:00:00:00:15	-	S5
		eth5	00:01:00:00:00:19	-	S8
		eth6	00:01:00:00:00:1a	-	S9
		eth7	00:01:00:00:00:1e	-	S10
		eth8	00:01:00:00:00:47	-	S11
		eth9	00:01:00:00:00:48	-	S12
		eth10	00:01:00:00:00:49	-	S6
		eth11	00:01:00:00:00:4a	-	S3
		eth12	00:01:00:00:00:4b	-	S13
		eth13	00:01:00:00:00:4c	-	S14
		eth14	00:01:00:00:00:4d	-	S7
		eth15	00:01:00:00:00:4e	-	S15

Table 2. *Cont*.

ID	Device	Interface	MAC	IPv4	Connected Device
3	OpenFlow Switch S1	eth0	00:01:00:00:00:04	1.0.1.2	Controller plane switch
		eth1	00:01:00:00:00:05	-	S2
		eth2	00:01:00:00:00:06	-	S3
4	OpenFlow Switch S2	eth0	00:01:00:00:00:07	-	S1
		eth1	00:01:00:00:00:08	-	S2
		eth2	00:01:00:00:00:09	-	S3
		eth3	00:01:00:00:00:4f	1.0.1.3	Controller plane switch
5	OpenFlow Switch S3	eth0	00:01:00:00:00:0a	-	S1
		eth1	00:01:00:00:00:0b	-	S6
		eth2	00:01:00:00:00:0c	-	S7
		eth3	00:01:00:00:00:50	1.0.1.12	Controller plane switch
6	OpenFlow Switch S4	eth0	00:01:00:00:00:0d	-	S2
		eth1	00:01:00:00:00:0e	-	S8
		eth2	00:01:00:00:00:0f	-	S9
		eth3	00:01:00:00:00:51	1.0.1.3	Controller plane switch
		eth4	00:01:00:00:00:5f	-	S5
7	OpenFlow Switch S5	eth0	00:01:00:00:00:11	-	S2
		eth1	00:01:00:00:00:12	-	S10
		eth2	00:01:00:00:00:13	-	S11
		eth3	00:01:00:00:00:52	1.0.1.3	Controller plane switch
		eth4	00:01:00:00:00:5d	-	S6
		eth5	00:01:00:00:00:60	-	S4
8	OpenFlow Switch S6	eth0	00:01:00:00:00:16	-	S12
		eth1	00:01:00:00:00:17	-	S13
		eth2	00:01:00:00:00:18	-	S3
		eth3	00:01:00:00:00:53	1.0.1.11	Controller plane switch
		eth4	00:01:00:00:00:5e	-	S5
		eth5	00:01:00:00:00:61	-	S7
9	OpenFlow Switch S7	eth0	00:01:00:00:00:1b	-	S3
		eth1	00:01:00:00:00:1c	-	S14
		eth2	00:01:00:00:00:1d	-	S15
		eth3	00:01:00:00:00:54	1.0.1.15	Controller plane switch
		eth4	00:01:00:00:00:62	-	S6
10	OpenFlow Switch S8	eth0	00:01:00:00:00:1f	-	S4
		eth1	00:01:00:00:00:20	-	H1
		eth2	00:01:00:00:00:21	-	H2
		eth3	00:01:00:00:00:55	1.0.1.6	Controller plane switch
11	OpenFlow Switch S9	eth0	00:01:00:00:00:22	-	S4
		eth1	00:01:00:00:00:23	-	H3
		eth2	00:01:00:00:00:24	-	H4
		eth3	00:01:00:00:00:56	1.0.1.7	Controller plane switch

Table 2. *Cont*.

ID	Device	Interface	MAC	IPv4	Connected Device
12	OpenFlow Switch S10	eth0	00:01:00:00:00:25	-	S5
		eth1	00:01:00:00:00:26	-	H5
		eth2	00:01:00:00:00:27	-	H6
		eth3	00:01:00:00:00:57	1.0.1.8	Controller plane switch
13	OpenFlow Switch S11	eth0	00:01:00:00:00:28	-	S5
		eth1	00:01:00:00:00:29	-	H7
		eth2	00:01:00:00:00:2a	-	H8
		eth3	00:01:00:00:00:58	1.0.1.9	Controller plane switch
14	OpenFlow Switch S12	eth0	00:01:00:00:00:2b	-	S6
		eth1	00:01:00:00:00:2c	-	H9
		eth2	00:01:00:00:00:2d	-	H10
		eth3	00:01:00:00:00:59	1.0.1.10	Controller plane switch
15	OpenFlow Switch S13	eth0	00:01:00:00:00:2e	-	S6
		eth1	00:01:00:00:00:2f	-	H11
		eth2	00:01:00:00:00:30	-	H12
		eth3	00:01:00:00:00:5a	1.0.1.13	Controller plane switch
16	OpenFlow Switch S14	eth0	00:01:00:00:00:31	-	S7
		eth1	00:01:00:00:00:32	-	H13
		eth2	00:01:00:00:00:33	-	H14
		eth3	00:01:00:00:00:5b	1.0.1.14	Controller plane switch
17	OpenFlow Switch S15	eth0	00:01:00:00:00:34	-	S7
		eth1	00:01:00:00:00:35	-	H15
		eth2	00:01:00:00:00:36	-	H16
		eth3	00:01:00:00:00:5c	1.0.1.16	Controller plane switch
18	Host H1	eth0	00:01:00:00:00:37	1.0.2.1	S8
19	Host H2	eth0	00:01:00:00:00:38	1.0.2.2	S8
20	Host H3	eth0	00:01:00:00:00:39	1.0.2.3	S9
21	Host H4	eth0	00:01:00:00:00:3a	1.0.2.4	S9
22	Host H5	eth0	00:01:00:00:00:3b	1.0.2.5	S10
23	Host H6	eth0	00:01:00:00:00:3c	1.0.2.6	S10
24	Host H7	eth0	00:01:00:00:00:3d	1.0.2.7	S11
25	Host H8	eth0	00:01:00:00:00:3e	1.0.2.8	S11
26	Host H9	eth0	00:01:00:00:00:3f	1.0.2.9	S12
27	Host H10	eth0	00:01:00:00:00:40	1.0.2.10	S12
28	Host H11	eth0	00:01:00:00:00:41	1.0.2.11	S13
29	Host H12	eth0	00:01:00:00:00:42	1.0.2.12	S13
30	Host H13	eth0	00:01:00:00:00:43	1.0.2.13	S14
31	Host H14	eth0	00:01:00:00:00:44	1.0.2.14	S14
32	Host H15	eth0	00:01:00:00:00:45	1.0.2.15	S15
33	Host H16	eth0	00:01:00:00:00:46	1.0.2.16	S15

3.2. AI Module and Machine Learning Models

To detect abnormal packets in the SDN network early and classify the attack type, we deployed an AI module in the controller. The AI module process is shown in Figure 4. In the training phase, we used two SDN public datasets to train the models for binary and multiclass classification. Binary classification is designed to distinguish between normal and abnormal, and multi-class classification is aimed at abnormality. Attacks can be classified into several types. There were eight ML and five DL algorithms in the AI model category. In the prediction phase, the network flows first enter the AI module, perform feature extraction, and then predict the binary classification or multiclass classification model. Finally, the controller uses the predictions to decide whether to act on network traffic.

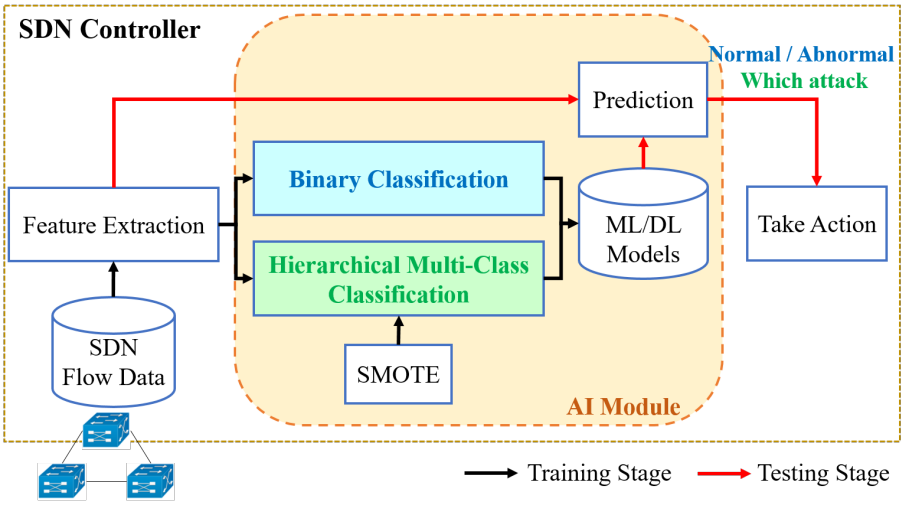

Figure 4. The framework of the AI module.

For binary and multi-class classification, we adopted eight common ML and five DL models to determine the best model, including decision tree, random forest, KNN, naive Bayes, SVM with the RBF kernel or the linear function, bagging and AdaBoost, MLP, convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU). There are two typical approaches to the synthetic minority oversampling technique (SMOTE) [57]: oversampling and undersampling. In the dataset used in this study, the number of samples in the minority class is too small, that is, 17 samples in U2R, 164 samples in Botnet, and 192 samples in Web-Attack; therefore, we adopted an oversampling strategy that did not affect the overall performance. Minority classes are oversampled through the creation of synthetic examples rather than by oversampling through replacement. Each minority class sample is oversampled by introducing synthetic examples along the line segments joining any or all of the k-nearest neighbors in the minority class. The neighbors from the KNN are selected randomly depending on the amount of oversampling required.

3.3. Hierarchical Multiclass Classification

Multiclass classification is required because it is necessary to further identify the category of the abnormal behavior. When the training dataset is unbalanced, the performance of multiclass classifiers decreases with an increasing number of classes, especially with a small number of classes (called minority classes); however, the overall average accuracy was good. Aiming at minority classes, we transform the multiclass classification problem into a hierarchical multi-classification (HMC) architecture. We adopted a top-down approach based on the divide-and-conquer strategy. The architecture is illustrated in Figure 5. The classifiers in the hierarchical mechanism can be any binary classifiers described in the previous subsection. First, we labeled the current class 0. The remaining categories were marked as 1. Training is conducted based on the number of samples in the class, that is,

the class with the greatest number of samples is trained first and training is performed only once.

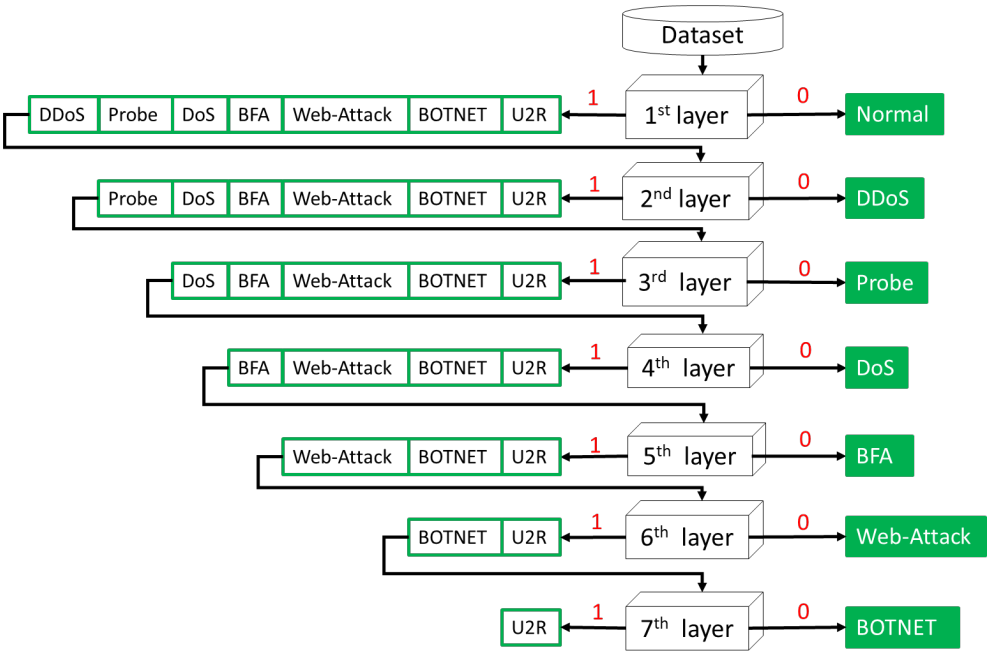

Figure 5. The HMC architecture.

3.4. Datasets and Features

Most of the training datasets published on traditional networks were collected from IDS, such as the KDD dataset [13] and NSL-KDD [14]; however, differences between the traditional and SDN environments remained. To resemble more closer to network attacks that may be encountered in the SDN environment, we selected two public datasets currently released in the SDN simulation environment for training. The binary classifier model adopts the DDOS attack SDN dataset published by Ahuja et al. [15], with 22 features and 104,345 records, as shown in Table 3.

Table 3. Two SDN datasets are used for binary classification and multi-class classification, respectively.

Dataset			DDoS-SDN (2020) [15]	InSDN (2020) [2]
The number of features			22	83
The number of classes			2 (binary)	8 (multi-class)
The number of instances per class	Normal		63,561	68,424
	Abnormal	DoS	40,784	53,616
		DDoS		121,942
		BFA		1405
		Botnet		164
		Web-Attack		192
		Probe		98,129
		U2R		17
The total number of instances			104,345	343,889

The dataset generated using the mininet emulator contained two types of packet characteristics: normal and malicious. Malicious traffic includes TCP Syn, UDP flood, and ICMP attacks. In addition, there are 22 features, such as Switch-id, Packet_count,

byte_count, and duration_sec, as shown in Table 4. Elsayed et al. adopted the InSDN dataset collected from the Mininet network emulator. The dataset includes seven classes of attack and normal packets with 83 features and 343,889 instances, as shown in Table 5. The authors selected 48 features for the ML training models. In this study, we selected these 48 features for the evaluation.

Table 4. Features of the DDoS-SDN dataset.

ID	Feature	Description	ID	Feature	Description
1	dt	Convert a date and time	2	switch	Switch ID
3	src	Source IP	4	dst	Destination IP
5	pktcount	Count of packets	6	bytecount	Count of bytes
7	dur	Duration	8	dur_nsec	Duration in nanoseconds
9	tot_dur	Sum of duration in seconds	10	flow	Flow amount at an interval
11	packetins	# of packets	12	pktperflow	Packets during a flow
13	byteperflow	Bytes during a flow	14	pktrate	Packets per second
15	pairflow	Packet per flow in a interval	16	protocol	TCP/UDP/ICMP
17	prot_no	Port No.	18	tx_bytes	Transfer bytes
19	rx_bytes	Receiving bytes	20	tx_kbps	Transfer kilobytes
21	rx_bytes	Receiving kilobytes	22	tot_kbps	Total kilobytes

Table 5. Features of the InSDN dataset.

ID	Feature	Select	ID	Feature	Select	ID	Feature	Select
1	Flow ID		2	Src IP		3	Src Port	
4	Dst IP		5	Dst Port		6	Protocol	✓
7	Timestamp		8	Flow duration	✓	9	Tot Fwd Pkts	✓
10	Tot Bwd Pkts	✓	11	Tot Len Fwd Pkts	✓	12	Tot Len Bwd Pkts	✓
13	Fwd Pkt Len Max	✓	14	Fwd Pkt Len Min	✓	15	Fwd Pkt Len Mean	✓
16	Fwd Pkt Len Std	✓	17	Bwd Pkt Len Max	✓	18	Bwd Pkt Len Min	✓
19	Bwd Pkt Len Mean	✓	20	Bwd Pkt Len Std	✓	21	Flow Byts/s	✓
22	Flow Pkts/s	✓	23	Flow IAT Mean	✓	24	Flow IAT Std	✓
25	Flow IAT Max	✓	26	Flow IAT Min	✓	27	Fwd IAT Tot	✓
28	Fwd IAT Mean	✓	29	Fwd IAT Std	✓	30	Fwd IAT Max	✓
31	Fwd IAT Min	✓	32	Bwd IAT Tot	✓	33	Bwd IAT Mean	✓
34	Bwd IAT Std	✓	35	Bwd IAT Max	✓	36	Bwd IAT Min	✓
37	Fwd PSH Flags		38	Bwd PSH Flags		39	Fwd URG Flags	
40	Fwd URG Flags		41	Fwd Header Len	✓	42	Bwd Header Len	✓
43	Fwd Pkts/s	✓	44	Bwd Pkts/s	✓	45	Pkt Len Min	✓
46	Pkt Len Max	✓	47	Pkt Len Mean	✓	48	Pkt Len Std	✓
49	Pkt Len Var	✓	50	FIN Flag Cnt		51	SYN Flag Cnt	

Table 5. *Cont.*

ID	Feature	Select	ID	Feature	Select	ID	Feature	Select
52	RST Flag Cnt		53	PSH Flag Cnt		54	ACK Flag Cnt	
55	URG Flag Cnt		56	CWE Flag Count		57	ECE Flag Cnt	
58	Down/Up Ratio		59	Pkt Size Avg	✓	60	Fwd Seg Size Avg	
61	Bwd Seg Size Avg		62	Fwd Byts/b Avg		63	Fwd Pkts/b Avg	
64	Fwd Blk Rate Avg		65	Bwd Byts/b Avg		66	Bwd Pkts/b Avg	
67	Bwd Blk Rate Avg		68	Subflow Fwd Pkts		69	Subflow Fwd Byts	
70	Subflow Bwd Pkts		71	Subflow Bwd Byts		72	Init Fwd Win Byts	
73	Init Bwd Win Byts		74	Fwd Act Data Pkts		75	Fwd Seg Size Min	
76	Active Mean	✓	77	Active Std	✓	78	Active Max	✓
79	Active Min	✓	80	Idle Mean	✓	81	Idle Std	✓
82	Idle Max	✓	83	Idle Min	✓			

4. Experiments

4.1. SDN Simulation

The SDN simulation system environment of this study is based on the Fedora24 operating system version, which supports 64-bit processor i7 machines with 16 GB RAM and 80 GB hard disk. The SDN simulator uses the commercial version of the software EstiNet, which combines the advantages of the simulator and the emulator (https://www.estinet.com/ns/?page_id=21140, accessed on 25 May 2021). Thus, the host of each simulator, including the SDN controller, has independent and real system resources, simulating the test network traffic of the host. The occurrence of network events can also be controlled programmatically. That is, the timing resources of the simulated environment are synchronized with the actual system time [58]. Through its GUI interface, the network configuration and packet transmission behavior of the real network can be presented in the software simulation.

EstiNet presents the real network configuration and packet transmission behavior in software simulation. OpenFlow supporting the SDN protocol is part of the EstiNet simulator. The OpenFlow communication protocol regulates the communication between controllers and switches in software-defined networking. An OpenFlow switch can accept control from a single controller or multiple controllers and supports an architecture where the control plane and the data plane overlap (in-band control plane) or separate (out-of-band control plane) architecture. The open-source controller software can execute in the SDN environment of the EstiNet simulation and then go to control the virtual OpenFlow switch. EstiNet also follows the OpenFlow protocol and supports FlowTable, GroupTable, and MeterTable operations in the simulated OpenFlow switch. The user can observe the changes of FlowTable, GroupTable, and MeterTable in the execution and playback mode of the simulator. EstiNet can programmatically control network statuses, such as packet transmission delay and packet loss rate for wired networks and energy/distance for wireless network packet transmission. It sends network packets using various wired and wireless communication protocols. Time dilation is generally used to simulate changes in time velocity. Simulation results are achieved by slowing down the system clock and remaining undistorted. The virtual network time of the simulated network is automatically adjusted based on packet events in the event processing core or through a packet event translator, providing greater accuracy and efficiency than typical time dilation methods. EstiNet uses Linux containers that run the actual software. Build virtual networks to create simulated environments that integrate virtual and real environments. EstiNet simulator

provides a network testing field consisting of physical and virtual network devices. Both types of devices can interact with each other for testing. Since Python language operation is supported, our ML and DL models are placed in the SDN controller for prediction. The Hing tool, a command-line oriented TCP/IP packet assembler/analyzer, is used as a DDoS simulation attack tool to implement SYN flood attacks and random source attack scenarios in the Linux environment. The attacker can send multiple connection requests to perform DDoS and send multiple random packets with different source addresses to the target machine.

4.2. Measure Evaluation

Comparing the classification performance objectively under the imbalanced dataset is not possible using only the accuracy rate because the performance of the average model tends to favor the performance of the majority classes. Thus, we select the F1-score to evaluate these malicious attack models in SDN. The equations are as follows, in which TP true positive (TP) and true negative (TN) indicate that the predictions are correct, and false positive (FP) and false negative (FN) indicate the wrong predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

4.3. Experimental Results

The experimental evaluations are divided into three performance comparison parts, including binary classification, multi-class classification, and the performance of HMC for multi-class classification.

In the first stage, we used the DDoS attack dataset to verify the prediction of the AI module in an SDN environment. We used 5-fold cross-validation with a training and testing data ratio of 8:2 (i.e., 4:1). A different part was used each time for testing, and the remainder was trained for five epochs to obtain the average result. The data were divided into two categories: normal and abnormal. We used eight ML classifiers for performance comparison: decision tree (DT), random forest (RF), naive Bayes (NB), KNN, SVM-RBF, linear kernel (L-SVM), and bagging and boosting for ensemble learning, as shown in Figure 6. In addition, five types of DL classifiers were adopted: MLP, CNN, RNN, LSTM, and GRU, as shown in Figure 7. ML models use default parameters without tuning. The parameter settings of the DL models are shown in Table 6.

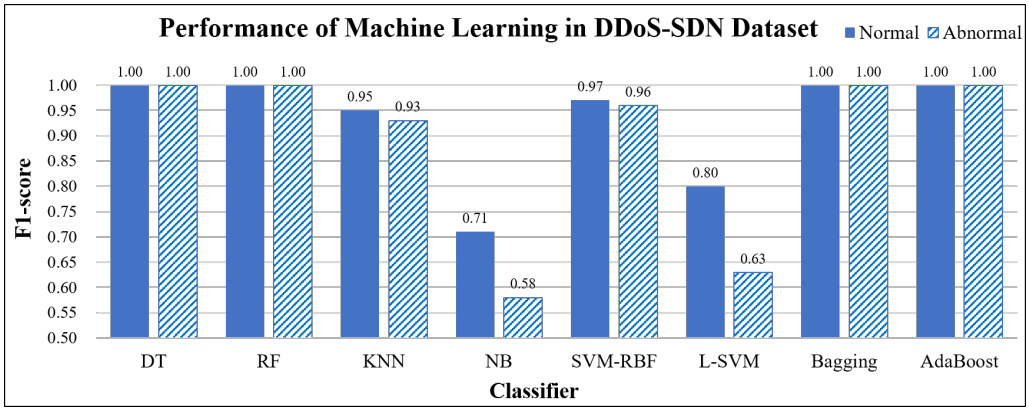

Figure 6. Performance of machine learning in DDoS-SDN dataset.

The experimental results showed that the DT, RF, bagging, and boost of ensemble learning had the best performance (F1 = 1) among the ML models. In contrast, NB had the worst performance (F1 = 0.58, abnormal packets). In addition, Figure 7 shows the performances of the five DL models. After optimizing the parameters, the F1-score for both categories is 0.95 and 0.99. This result indicates that tree, ensemble learning, and neural network models can effectively predict DDoS attacks.

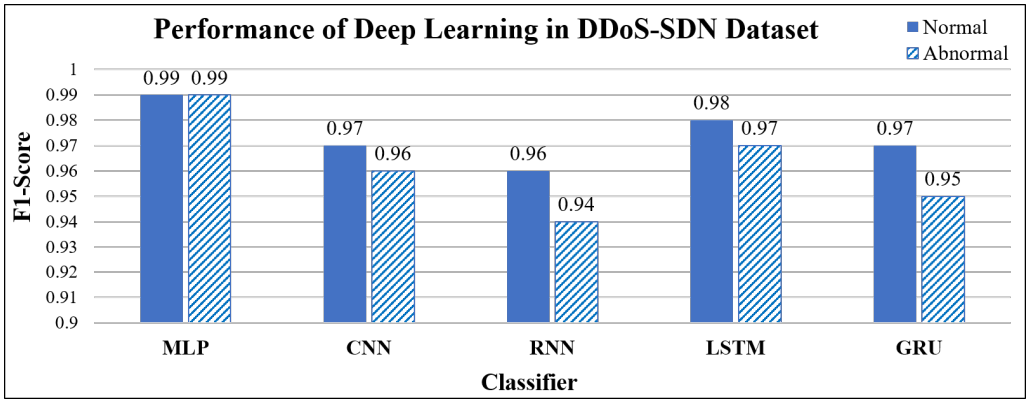

Figure 7. Performance of deep learning in DDoS-SDN dataset.

Table 6. Parameter setting of deep learning models.

Models	# of Layer	Type	# of Neurons	Activation
MLP	Layer1	Dense	80	Relu
	Layer2	Dense	100	Relu
	Layer3	Dense	8	Softmax
CNN	Layer1	Conv1D	64	Relu
	Layer2	Conv1D	64	Relu
	Layer3	MaxPooling1D (pool size = 1)		
	Layer4	Dense	128	Relu
	Layer5	Dropout (p = 0.5)		
	Layer6	Dense	8	Softmax
RNN	Layer1	SimpleRNN	64	Relu
	Layer2	Dense	256	Relu
	Layer3	Dropout (p = 0.35)		
	Layer4	Dense	8	Softmax
LSTM	Layer1	LSTM	64	Relu
	Layer2	Dense	256	Relu
	Layer3	Dense	256	Relu
	Layer4	Dense	8	Softmax
GPU	Layer1	GRU	64	Relu
	Layer2	Dropout (p = 0.2)		
	Layer3	Dense	256	Relu
	Layer4	Dense	256	Relu
	Layer5	Dense	8	Softmax

In the second stage, we used the InSDN dataset to validate the prediction of the AI module for multiclass anomalous attacks in an SDN environment. The same eight machine learning and five DL models were used in the first stage, and the performance classifications are shown in Figure 8 and Figure 9, respectively. The horizontal axis represents the overall F1-score performance and normal and seven abnormal attack performances. Figure 8 shows the performance comparison of machine learning for various attacks. From the

overall average performance (see the first set of data in Figure 8), the performance of each classifier was good (excluding NB; the F1-score was above 0.83). In detail, normal packets, DDoS, DoS, and probe attacks have good identification performance, whereas botnets, BFA, web-attack, and U2R have poor performance in sequence. Next, we compare the number of classes, as shown in Table 3. There is a clear relationship between poor performance and insufficient data; therefore, this is a problem of uneven training performance caused by a significantly imbalanced dataset.

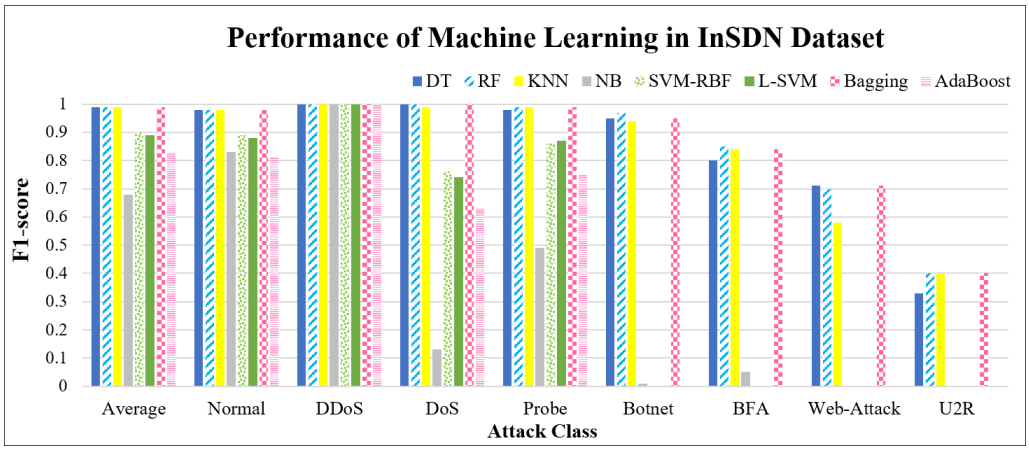

Figure 8. Performance of machine learning in InSDN dataset.

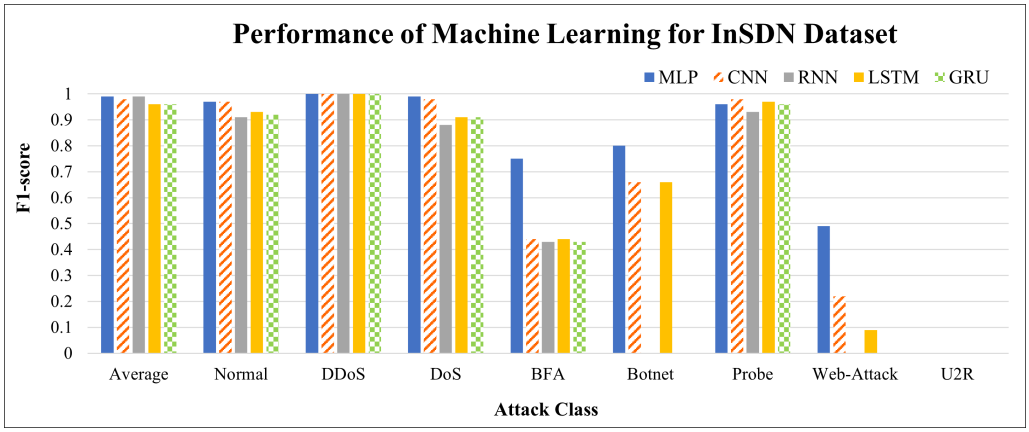

Figure 9. Performance of deep learning in InSDN dataset.

The performance of DL on imbalanced datasets is shown in Figure 9. Compared with the experimental results in Figure 8, the overall performance of the DL model was better than that of the ML model (F1 was 0.96∼0.99); however, in terms of detailed categories, the prediction performance of the U2R class is still the worst, and the best classification performance of web attack is only 0.49; therefore, combining the results obtained in Figures 8 and 9, we need to strengthen the identification performance of the few-shot categories (i.e., U2R, web-attack, BFA, and botnet).

In the third stage, we compare the performance of HMC with 13 single classifiers, which are the same as above, focusing on the minority classes, as shown in Figure 10 and Figure 11, respectively. As shown in Figure 10, the HMC architecture with AdaBoost as the core classifier exhibited the best performance, followed by bagging. Regarding minority classes, we see that AdaBoost-based HMC in the U2R class attaches a 0.5 F1-score (original F1 is 0) and bagging-based HMC attaches a 0.67 F1-score (original F1 is 0.4). For the web-attack class, the AdaBoost-based HMC obtained a 0.88 F1-score (original F1 was 0.71) and bagging obtained a 0.9 F1-score (original F1 was 0). Because the two classification methods, AdaBoost and bagging, use ensemble learning as the strategy, it

also shows that multi-classifier decision-making contributes to the identification ability of the final prediction.

As shown in Figure 11, the DL-based HMC architecture improves in most classes but fails to identify effectively for the botnet and U2R classes. Comparing Figure 10 and Figure 11, we found that although the DL model has been significantly improved in most categories, AdaBoost and bagging, which are still based on ensemble learning strategies, are still used to identify minority categories and exhibit better performance under the HMC architecture. Finally, we averaged the eight types of performance using the Micro F1-score, as shown in Table 7. With AdaBoost improving the most (51%) and RNN good (21%); however, under the HMC architecture, the MLP degraded by 6%. We assumed the overall average drop might be due to misrecognition prediction for botnet attacks.

These two SDN datasets were simulated using the Mininet software and released publicly for research purposes. The authors labeled anomalous traffic categories so that the model could be trained offline, predicted, and identified online. According to Elsayed et al. [2], the InSDN dataset improves the accuracy of model identification in predicting real-time traffic in an SDN environment. This study used these data to train an AI model to enable an SDN system. We can also replace this with real traffic and extract features to provide model predictions in real situations; therefore, our SDN system is scalable and independent of datasets used.

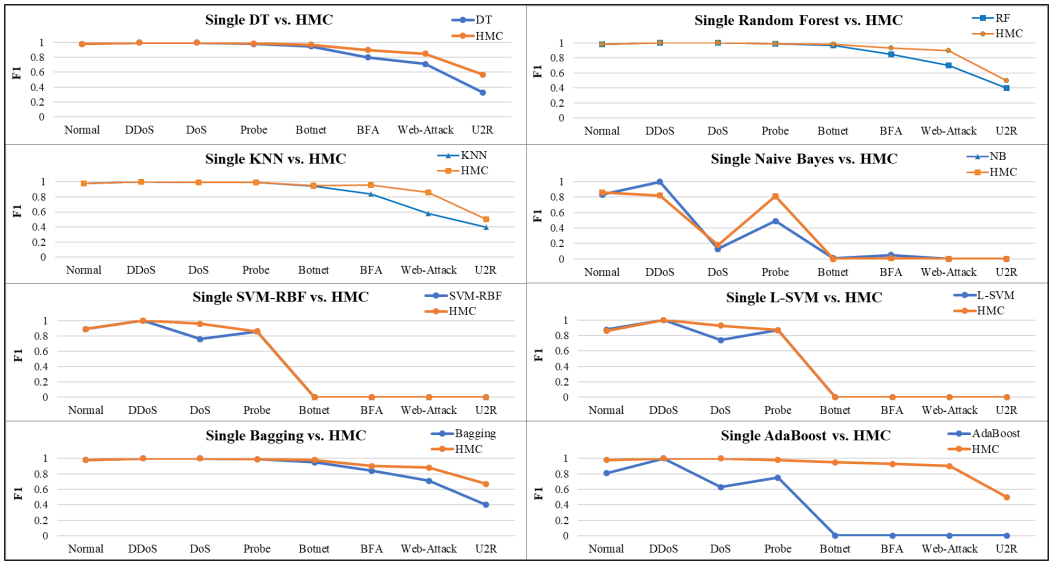

Figure 10. Performance comparison of HMC with single machine learning classifiers.

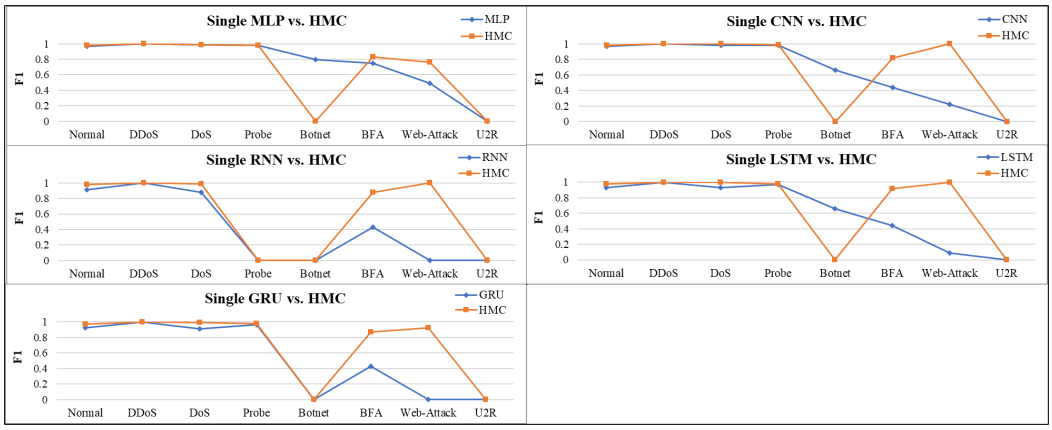

Figure 11. Performance comparison of HMC with single deep learning classifiers.

Table 7. Summary of HMC improved performance.

Model	Single Classifier (F1)	HMC Architecture (F1)	Improved (%)
Decison tree	0.85	0.98	↑ 13%
Random forest	0.86	0.91	↑ 5%
KNN	0.84	0.90	↑ 6%
Naïve Bayes	0.33	0.46	↑ 13%
SVM-RBF	0.44	0.46	↑ 2%
L-SVM	0.44	0.46	↑ 2%
Bagging	0.86	0.93	↑ 7%
AdaBoost	0.40	0.91	↑ **51%**
MLP	0.75	0.69	↓ 6%
CNN	0.66	0.72	↑ 6%
RNN	0.52	0.73	↑ 21%
LSTM	0.63	0.74	↑ 11%
GRU	0.53	0.72	↑ 19%

4.4. Case of Attack Simulation

During this experiment, we generated traffic and abnormal packets using the Hping network tool to simulate the SDN controller's traffic monitoring and the response of the SDN controller to the packets generated by a DDoS attack scenario. We collected normal traffic for analysis in a simulated DDoS attack environment and distinguished between normal and DDoS attack traffic. Finally, the flow rule is dynamically updated to the data layer devices to discard or block abnormal packets. Figure 12 illustrates the number of packets received and transmitted and the number of bytes received and transmitted by the switch-on ports 1, 2, and 3. Moreover, packets were sent to a specific destination in large numbers and after the packets exceeded the threshold value, the packets were blocked by the SDN controller and discarded. In addition, the message "Destination Host Unreachable" was displayed.

To better visualize the DDoS attack traffic and the throughput load imposed on the victim host, we used EstiNet software to simulate the entire simulation process, where node 1 acts as the initiator of the DDoS attack and node 2 is the victim. The DDoS attack simulation tool is Hping. EstiNet can support self-built Docker mounting technology. In the following example, a new image file is created using the Fedora Docker file, and the Hping tool is installed on node 1 using EstiNet software. When the Hping command (hping 3–c 10000–d 60–S–p 80 –flood –rand-source 1.0.1.2) simulates a DDoS attack, the throughput load of the victim host rises rapidly. This DDoS attack simulation scenario verifies whether the architecture can detect and mitigate the attack traffic. The number of attacks includes one, two, and three hosts to attack the victim host. When the system detected abnormal traffic, the mitigation mechanism was activated within 1–2 s to restore the traffic of the victim host to normal.

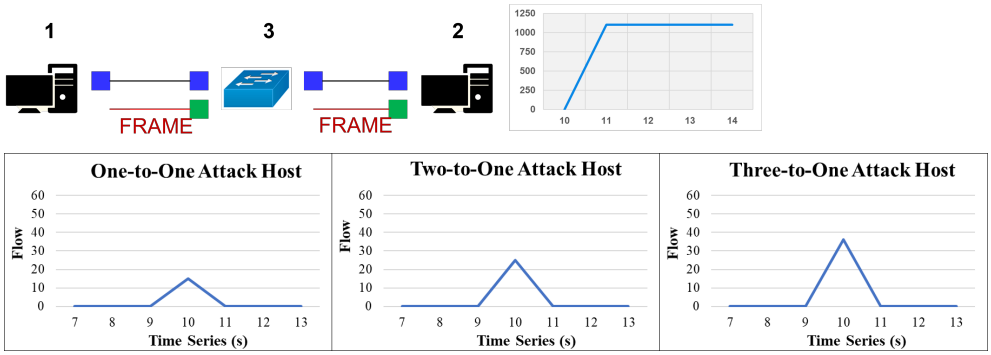

Figure 12. DDoS attack simulation under one to three number of attack hosts.

4.5. Feature Importance and Selection

A high the number of features in a dataset can help identify the differences between normal and abnormal packets; however, a significant disadvantage is the relative increase in model complexity and training cost; therefore, finding important features in the dataset and performing feature selection are beneficial for effectively identifying abnormal attack categories and reducing training costs. The information gain (IG) indicator is often used in the field of information theory as an indicator of the importance of an attribute and decision making; therefore, IG value was used to indicate the important features in the dataset, and the calculation equation is given in Equation (5).

$$\text{Entropy} = -\sum_{i=1}^{c} p_i log p_i \tag{4}$$

$$\text{IG}(T, A) = \text{Entropy}(T) - \sum_{v \in A} \frac{T_v}{T} \cdot \text{Entropy}(T_v) \tag{5}$$

where T is target, A is the variable, and v is each value in A.

In addition to the information gain index, another related index is entropy, which measures the degree of chaos in data categories. When the value is higher, the data are more chaotic, and vice versa, and it is easy to divide. Through ablation analysis, we calculated the IG values per feature for the overall model prediction (i.e., the importance of features).

As shown in Figure 13, in the DDoS-SDN dataset, the features are sorted according to their importance, which is calculated using the information gain. For example, byteperflow, pktperflow, pktrate, bytecount, pktcount, and packets were the first six important characteristics for identifying DDoS packets. Compared with the InSDN dataset, as shown in Table 8, we list the top 20 features (IG > 0) that can be used as a basis for feature selection for model performance and efficiency. The first three features (i.e., Dst Port, Init Bwd Win Bytes, and Pkt Len Max) have large IG values, and these features also help correctly identify whether the packet is normal or malicious.

Finally, we selected the top k features to compare the feature performances, as shown in Figure 14. The experiment was evaluated based on a decision tree model. The horizontal axis in Figure 14 is the selection of the top 10 to the top 70 and a total of 83 features; the selected is the baseline (48 features are used in InSDN). The ranking of k is based on the information gain. The higher the value of k, the more critical it is. The test method employed ablation analysis. Each feature was excluded individually, relative to the overall performance, to examine the importance of that feature. The results show that selecting the top 20 and 60 features has the best F1 score (0.95), which is an interesting result. When we prioritize a lower model complexity, a model trained on the top 20 features is likely to exhibit the best performance and efficiency. Nonetheless, when we are concerned about the need to identify multiple types of attacks, particularly for minority sample classes, using the top 60 features may be more helpful in indicating the differences between attack classes;

therefore, using few-shot learning by selecting relevant features to tackle unknown threat prediction is a future research direction.

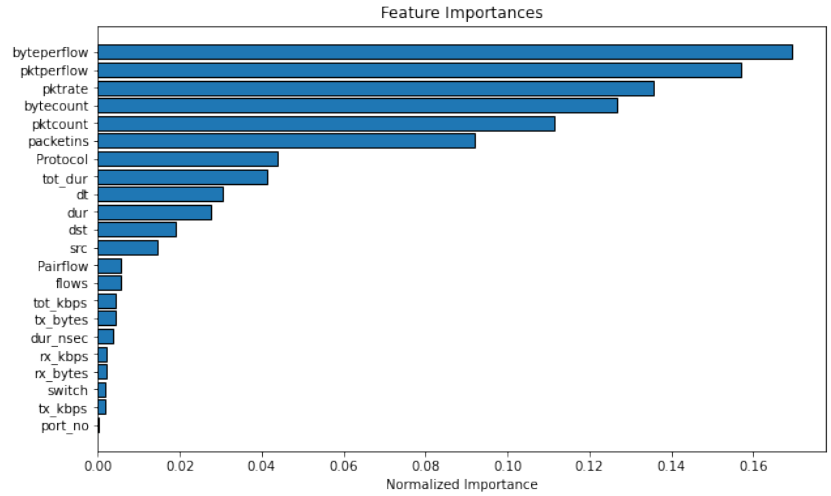

Figure 13. The rank of features in the DDoS-SDN dataset.

Table 8. InSDN dataset ranks the top 20 features by information gain.

Order	Feature	IG	Order	Feature	IG
1	Dst Port	0.466337	2	Init Bwd Win Byts	0.188352
3	Pkt Len Max	0.184689	4	Bwd Header Len	0.060474
5	Bwd Pkt Len Max	0.024999	6	PSH Flag Cnt	0.024586
7	Src Port	0.018379	8	Flow IAT Std	0.018379
9	Flow IAT Max	0.005798	10	Active Min	0.004491
11	Idle Min	0.002452	12	Subflow Fwd Pkts	0.002375
13	Bwd Pkt Len Mean	0.002219	14	Subflow Fwd Pkts	0.001317
15	Flow IAT Mean	0.000926	16	Fwd Act Data Pkts	0.000898
17	Down/Up Ratio	0.000595	18	Protocol	0.000538
19	RST Flag Cnt	0.000483	20	Tot Fwd Pkts	0.000451

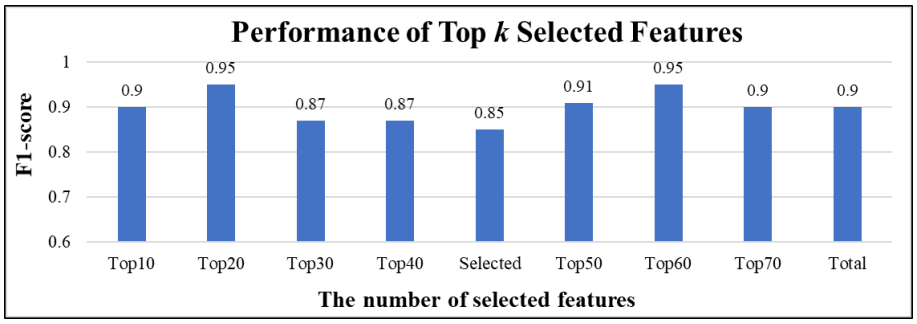

Figure 14. Performance of top *k* selected features.

4.6. Discussion

- **ML experimental design.** We qualitatively compare experimental results with previous related work, exploring model combination performance, feature selection performance, and overall and minority performance.
    1. *Single model and multi model.* The experimental results show that our HMC model, ensemble voting, random forest, and Adaboost have outstanding performance among many models and can summarize the predictions of multiple models. The performance is better than the prediction of a single model. In practice, the data distribution is often uneven, so the combination of multiple approaches, such as the same classifiers with different samples, and various classifiers combined with different weighted voting, will effectively improve the performance of the minority class under the imbalanced dataset.
    2. *All features and selected features.* There is no need to consider the feature selection for related work with a small number of features. That is, all of them are selected as the model training classification; however, for datasets with many features, the model complexity and the training time cost increase, so feature selection needs to be considered. There may also be irrelevant features that even degrade performance. The indicators of feature selection are also critical, especially when some features are not uniformly distributed. As a result, they may be biased towards the features with the majority of attributes.
    3. *Overall and minority performance.* In most studies, we find that accuracy is often used to measure the overall performance, but in the imbalanced datasets, the overall results may be misleading because they may not reveal the performance of minority attacks; therefore, we proposed the HMC model to improve multiclass performance through multiple binary classifiers, reduce the number between the majority class and the minority class, and adopt SMOTE sampling to increase the samples of the minority.
- **The advantages and threats of SDN in security.** SDN is derived from decoupling the control and forwarding planes of the network so that the complex routing mechanism of the traditional network is responsible for a centralized controller, which works according to the described strategy through an intelligent and programmable logic centralized controller. Due to the core differences between SDN and traditional networks, it brings new solutions to security issues. For example, the global network view can support higher detection of malicious traffic intrusion and help detect the malicious behavior of network switches; however, the SDN architecture itself also faces new security threats. In conclusion, the single point of failure problem (SPOF) is essentially due to centralization; therefore, we briefly discuss the security issues of the SDN environment arising from the forwarding plane, the control plane, and links.
    1. *Forwarding plane.* Due to the limited storage capacity of the switch, a reactive caching mechanism is used. Whenever the switch does not find a matching rule for its incoming packet flow, the packet will be temporarily stored in the switch buffer and sent to the controller. Sending a query requires missing rules. This reactive caching mechanism makes the switch vulnerable to DoS attacks.
    2. *Control plane.* The control plane is also vulnerable to DDoS attacks because multiple infected hosts distributed in the network may send packets to the network switch synchronously. Since not all rules are already available in the switch's table, many queries will be generated and sent to the controller, eventually exploiting the controller's limited processing power to cause legitimate queries to be delayed or discarded. Replication can be used to address such attacks, with multiple controllers managing the network instead of a single controller; however, when multiple controllers work the network, deciding where to deploy the controllers is a critical issue. In addition, the distance separating the switch from its central controller is a crucial factor to consider when making a place-

ment. Keeping this distance short ensures low latency on the switch controller communication link, improving network responsiveness and making switch DoS attacks more difficult. In addition, SDN also needs to have some resilience against compromised controller attacks. The control duplication workaround described above is resistant to this type of attack. Still, if all controllers are installed on similar platforms, once successfully hacking one of them, an attacker can hack all controllers.

3. *Forwarding the control link.* Sending unencrypted communication messages over the link connecting the control and forwarding planes makes the link vulnerable to man-in-the-middle attacks. In this case, an attacker can infer control policies, tamper with rules, and create new rules by eavesdropping on the communications exchanged on the link, giving the attacker full control of the switch; therefore, encryption must prevent eavesdropping, thereby protecting the link layer from such attacks.

- **System scalability.** According to the technical development and the actual needs of the unit, a preplanned cloud service based on the network characteristics of the unit is proposed, and an SDN topology architecture based on AI-assisted early detection is constructed; therefore, the premise of our study is further to deploy this system for practical use in the future. Considering the lack of labeled datasets in real-time networks, we adopted the two datasets used in this study to provide offline training for ML and DL models. These two SDN datasets were collected by Mininet software and publicly released for research use. The authors labeled the types of abnormal traffic and extracted 22 and 83 features from the SDN environment. According to Elsayed et al., the features of the InSDN dataset can effectively train for traffic anomaly detection in SDN environments. In this study, our final goal is to adopt these datasets for offline training and online prediction. The follow-up work in our research will use transfer learning to use the model parameters of the source domain for fine-tuning models of the target domain; therefore, our AI-based SDN system is scalable and does not depend on the original datasets.

5. Conclusions

With the rapid development of SDN and enhancement of information protection awareness, the type and intensity of attacks pose a major threat to information security. SDN architecture has the advantages of centralized control, hierarchical management, and rapid expansion; however, owing to its centralized control, it has also become a prime target for attackers; therefore, based on security considerations, this study used ML and DL models to assist in building the SDN topology architecture of cloud services to achieve early anomaly detection. In addition, we propose a hierarchical multiclass classification architecture to improve multi-class classification performance and a variety of model selections by replacing the combination of core classifiers. Finally, we simulate DDoS attack scenarios to verify that the model has detection capabilities. This study was implemented to improve the security of an SDN environment using attack prediction and mitigation mechanisms. The main tasks of this study include (1) constructing the SDN network topology, (2) training ML and DL models to analyze their ability to identify abnormal attacks, and (3) improving the performance of multiclass classification based on our proposed HMC architecture to improve imbalanced dataset classification, especially the minority class.

In a future study of an SDN centralized control design, we will first consider adopting multiple controllers to enhance the controller failover and load-balancing functions. Second, we applied few-shot learning to address the prediction accuracy challenge of the minority class in anomaly attack training. Finally, because the types of attacks change very quickly, we use transfer learning to predict unknown threats based on the trained attack type model to achieve early detection of abnormal attacks. We will focus on the controller design and integrated development of cross-cloud applications.

Author Contributions: H.-M.C. conducted methodology, software, and writing—original draft preparation. F.L. conducted conceptualization and review and editing. C.-H.T. conducted review and editing, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research is grateful for the support of the National Defense Science and Technology Academic Collaborative Research Project in 2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The SDN public datasets can be downloaded from the websites of the original authors. For the experimental results of this manuscript, you can contact the corresponding author.

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

References

1. Clemm, A.; Zhani, M.F.; Boutaba, R. Network Management 2030: Operations and Control of Network 2030 Services. *J. Netw. Syst. Manag*. **2020**, *28*, 721–750. [CrossRef]
2. Elsayed, A.M.S.; Le-Khac, N.-A.; Jurcut, A.D. InSDN: A Novel SDN Intrusion Dataset. *IEEE Access*. **2020**, *8*, 165263–165284. [CrossRef]
3. Jahromi, H.Z.; Delaney, D.T. An Application Awareness Framework based on SDN and Machine Learning: Defining the Roadmap and Challenges. In Proceedings of the 10th International Conference on Communication Software and Networks (ICCSN), Chengdu, China, 6–9 July 2018; pp. 411–416.
4. Ahmed, M.R.; Islam, S.; Shatabda, S.; Muzahidul Islam, A.K.M.; Robin, M.T.I. Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques—A Comprehensive Survey. *TechRxiv Preprint* **2021**. [CrossRef]
5. Thakur, N.; Han, C.Y. A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method. *J. Sens. Actuator Netw.* **2021**, *10*, 39. [CrossRef]
6. Lee, C.; Hong, J.; Heo, D.; Choi, H. Sequential Deep Learning Architectures for Anomaly Detection in Virtual Network Function Chains. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 20–22 October 2021; pp. 1163–1168.
7. Fan, C.; Kaliyamurthy, N.M.; Chen, S.; Jiang, H.; Zhou, Y.; Campbell, C. Detection of DDoS Attacks in Software Defined Networking Using Entropy. *Appl. Sci.* **2022**, *12*, 370. [CrossRef]
8. Aslam, M.; Ye, D.; Tariq, A.; Asad, M.; Hanif, M.; Ndzi, D.; Chelloug, S.A.; Elaziz, M.A.; Al-Qaness, M.A.A.; Jilani, S.F. Adaptive Machine Learning Based Distributed Denial-of-Services Attacks Detection and Mitigation System for SDN-Enabled IoT. *Sensors* **2022**, *22*, 2697. [CrossRef]
9. Maheshwari, A.; Mehraj, B.; Khan, M.S.; Idrisi, M.S. An Optimized Weighted Voting Based Ensemble Model for DDoS Attack Detection and Mitigation in SDN Environment. *Microprocess. Microsyst.* **2022**, *89*, 104412. [CrossRef]
10. Liu, Y.; Zhi, T.; Shen, M.; Wang, L.; Li, Y.; Wan, M. Software-Defined DDoS Detection with Information Entropy Analysis and Optimized Deep Learning. *Future Gener. Comput. Syst.* **2022**, *129*, 99–114. [CrossRef]
11. Chetouane, A.; Karoui, K. A Survey of Machine Learning Methods for DDoS Threats Detection Against SDN. In *Distributed Computing for Emerging Smart Networks (DiCES-N)*; Communications in Computer and Information Science; Jemili, I., Mosbah, M., Eds.; Springer: Cham, Switzerland, 6 April 2022; Volume 1564. [CrossRef]
12. Sudar, K.M.; Beulah, M.; Deepalakshmi, P.; Nagaraj, P.; Chinnasamy, P. Detection of Distributed Denial of Service Attacks in SDN using Machine learning techniques. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 21 April 2021; pp. 1–5.
13. KDD Cup 1999. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 25 May 2021).
14. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
15. Ahuja, N.; Singal, G.; Mukhopadhyay, D. DDOS attack SDN Dataset. *Mendeley Data* **2020**. [CrossRef]
16. Benzekki, K.; El Fergougui, A.; Elalaoui, E.A. Software-Defined Networking (SDN): A Survey. *Secur. Commun. Netw.* **2016**, *9*, 5803–5833. [CrossRef]
17. Bedhief, I.; Kassar, M.; Aguili, T.; Foschini, L. Self-Adaptive Management of SDN Distributed Controllers for Highly Dynamic IoT Networks. In Proceedings of the 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 2098–2104.
18. Ochoa-Aday, L.; Cervelló-Pastor, C.; Fernández-Fernández, A. eTDP: Enhanced Topology Discovery Protocol for Software-Defined Networks. *IEEE Access* **2019**, *7*, 23471–23487. [CrossRef]

19. Gyllstrom, D.; Braga, N.; Kurose, J. Recovery from Link Failures in a Smart Grid Communication Network Using Openflow. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 254–259.
20. Naous, J.; Erickson, D.; Covington, G.A.; Appenzeller, G.; McKeown, N. Implementing an OpenFlow Switch on the NetFPGA Platform. In Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '08), New York, NY, USA, 1–9 November 2008; ACM: New York, NY, USA, 2008; pp. 1–9.
21. Tandon, R. A Survey of Distributed Denial of Service Attacks and Defenses. *arXiv* **2020**, arXiv:2008.01345.
22. Shin, S.; Gu, G. Attacking Software-Defined Networks: A First Feasibility Study. In Proceedings of the second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), New York, NY, USA, 16 August 2013; Foster, N., Sherwood, R., Eds.; ACM: New York, NY, USA, 2013; pp. 165–166.
23. Shin, S.; Yegneswaran, V.; Porras, P.; Gu, G. Avant-guard: Scalable and Vigilant Switch Flow Management in Software-Defined Networks. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS'13), Berlin, Germany, 4–8 November 2013; Sadeghi, A.-R., Ed.; ACM: New York, NY, USA, 2013; pp. 413–424.
24. 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.
25. Akhunzada, A.; Ahmed, E.; Gani, A.; Khan, M.K.; Imran, M.; Guizani, S. Securing Software Defined Networks: Taxonomy, Requirements, and Open Issues. *IEEE Commun. Mag.* **2015**, *53*, 36–44. [CrossRef]
26. Zhang, P.; Wang, H.; Hu, C.; Lin, C. On Denial of Service Attacks in Software Defined Networks. *IEEE Netw.* **2016**, *30*, 28–33. [CrossRef]
27. Dover, J.M. *A Denial of Service Attack against the Open Floodlight SDN Controller*; Dover Networks LLC: Edgewater, MD, USA, 2013.
28. 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]
29. Khairi, M.H.H.; Ariffin, S.H.S.; Latiff, N.M.A.A.; Yusof, K.M.; Hassan, M.K.; Al-Dhief, F.T.; Hamdan, M.; Khan, S.; Hamzah, M. Detection and Classification of Conflict Flows in SDN Using Machine Learning Algorithms. *IEEE Access* **2021**, *9*, 76024–76037. [CrossRef]
30. Hamdan, M.; Mohammed, B.; Humayun, U.; Abdelaziz, A.; Khan, S.; Ali, M.A.; Imran, M.; Marsono, M.N. Flow-aware Elephant Flow Detection for Software-Defined Networks. *IEEE Access* **2020**, *8*, 72585–72597. [CrossRef]
31. Kuranage, M.P.J.; Piamrat, K.; Hamma, S. Network Traffic Classification Using Machine Learning for Software Defined Networks. In Proceedings of the International Conference on Maching Learning for Netwwork (MLN), Paris, France, 3–5 December 2019; Boumerdassi, S., Renault, É., Mühlethaler, P., Eds.; Springer: Cham, Switzerland, 2020; pp. 28–39.
32. Khamaiseh, S.; Serra, E.; Li, Z.; Xu, D. Detecting Saturation Attacks in SDN via Machine Learning. In Proceedings of the 2019 4th International Conference on Computing, Communications and Security (ICCCS), Rome, Italy, 10–12 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8.
33. Tang, F.; Zhang, H.; Yang, L.T.; Cheng, L. Elephant Flow Detection and Differentiated Scheduling with Efficient Sampling and Classification. *IEEE Trans. Cloud Comput.* **2021**, *9*, 1022–1036. [CrossRef]
34. Latah, M.; Toker, L. Artificial Intelligence Enabled Software-Defined Networking: A Comprehensive Overview. *IET Netw.* **2019**,*8*, 79–99. [CrossRef]
35. Comaneci, D.; Dobre, C. Securing Networks Using SDN and Machine Learning. In Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE), Bucharest, Romania, 29–31 October 2018; IEEE: Los Alamitos, CA, USA, 2018; pp. 194–200.
36. Wang, P.; Ye, F.; Chen, X.; Qian, Y. DataNet: Deep Learning based Encrypted Network Traffic Classification in SDN Home Gateway. *IEEE Access* **2018**, *6*, 55380–55391. [CrossRef]
37. Latah, M.; Toker, L. Application of Artificial Intelligence to Software Defined Networking: A Survey. *Indian J. Sci. Technol.* **2016**, *9*, 1–7. [CrossRef]
38. Krishnan, P.; Duttagupta, S.; Achuthan, K. Varman: Multi-plane Security Framework for Software Defined Networks. *Comput. Commun.* **2019**, *148*, 215–239. [CrossRef]
39. Bao, K.; Matyjas, J.D.; Hu, F.; Kumar, S. Intelligent Software-Defined Mesh Networks with Link-Failure Adaptive Traffic Balancing. *IEEE Trans. Cognit. Commun. Netw.* **2018**, *4*, 266–276. [CrossRef]
40. Amaral, P.; Dinis, J.; Pinto, P.; Bernardo, L.; Tavares, J.; Mamede, H.S. Machine Learning in Software Defined Networks: Data Collection and Traffic Classification. In Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 11–16 November 2016; pp. 1–5.
41. Yuan, B.; Zou, D.; Yu, S.; Jin, H.; Qiang, W.; Shen, J. Defending Against Flow Table Overloading Attack in Software-Defined Networks. *IEEE Trans. Serv. Comput.* **2019**, *12*, 231–246. [CrossRef]
42. Rasool, R.U.; Ashraf, U.; Ahmed, K.; Wang, H. Cyberpulse: A Machine Learning based Link Flooding Attack Mitigation System for Software Defined Networks. *IEEE Access* **2019**, *7*, 34885–34899. [CrossRef]
43. Tseng, C.-W.; Wu, L.-F.; Hsu, S.-C.; Yu, S.-W. IPv6 DoS Attacks Detection Using Machine Learning Enhanced IDS in SDN/NFV Environment. In Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea, 22–25 September 2020; pp. 1–10.

44. Tonkal, Ö.; Polat, H.; Başaran, E.; Cömert, Z.; Kocaoğlu, R. Machine Learning Approach Equipped with Neighbourhood Component Analysis for DDoS Attack Detection in Software-Defined Networking. *Electronics* **2021**, *10*, 1227. [CrossRef]
45. Toupas, P.; Chamou, D.; Giannoutakis, K.M.; Drosou, A. An Intrusion Detection System for Multi-class Classification Based on Deep Neural Networks. In Proceedings of the 18th IEEE International Conference On Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; Khoshgoftaar, T.M., Ed.; IEEE: Piscataway, NJ, USA, 2019; pp. 1253–1258.
46. Yu, Y.; Guo, L.; Liu, Y.; Zheng, J.; Zong, Y. An Efficient SDN-based DDoS Attack Detection and Rapid Response Platform in Vehicular Networks. *IEEE Access* **2018**, *6*, 44570–44579. [CrossRef]
47. Huseyin, P.; Polat, O.; Aydin, C. Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models. *Sustainability* **2020**, *12*, 1035. [CrossRef]
48. Khairi, M.H.H.; Ariffin, S.H.S.; Latiff, N.M.A.; Yusof, K.M. Generation and Collection of Data for Normal and Conflicting Flows in Software Defined Network Flow Table. *Indonesian J. Electr. Eng. Comput. Sci.* **2021**, *22*, 307–314. [CrossRef]
49. Dey, S.K.; Rahman, M.M. Effects of Machine Learning Approach in Flow-Based Anomaly Detection on Software-Defined Networking. *Symmetry* **2020**, *12*, 7. [CrossRef]
50. 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]
51. Pendlebury, F.; Pierazzi, F.; Jordaney, R.; Kinder, J.; Cavallaro, L. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19), Santa Clara, CA, USA, 14–16 August 2019; USENIX Association: Berkeley, CA, USA, 2019; pp. 729–746.
52. Narayanan, A.; Chandramohan, M.; Chen, L.; Liu, Y. Context-Aware, Adaptive, and Scalable Android Malware Detection through Online Learning. *IEEE Trans. Emerg. Top. Comput. Intellig.* **2017**, *1*, 157–175. [CrossRef]
53. Xu, K.; Li, Y.; Deng, R.; Chen, K.; Xu, J. Droidevolver: Self-Evolving Android Malware Detection System. In Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 17–19 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 47–62.
54. Jordaney, R.; Sharad, K.; Dash, S.K.; Wang, Z.; Papini, D.; Cavallaro, L. Transcend: Detecting Concept Drift in Malware Classification Models. In Proceedings of the 26th USENIX Conference on Security Symposium, Vancouver, BC, Canada, 16–18 August 2017; USENIX Association: Berkeley, CA, USA, 2017; pp. 625–642.
55. Barbero, F.; Pendlebury, F.; Pierazzi, F.; Cavallaro, L. Transcending Transcend: Revisiting Malware Classification in the Presence of Concept Drift. *arXiv* **2020**, arXiv:2010.03856.
56. Cai, H. Assessing and Improving Malware Detection Sustainability through App Evolution Studies. *ACM Trans. Softw. Eng. Methodol.* **2020**, *29*, 8. [CrossRef]
57. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. Smote: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
58. Wang, S.; Chou, C.; Yang, C. EstiNet Openflow Network Simulator and Emulator. *IEEE Commun. Mag.* **2013**, *51*, 110–117. [CrossRef]