IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 August 2023, accepted 15 September 2023, date of publication 25 September 2023, date of current version 4 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3319214

RESEARCH ARTICLE

Optimized Artificial Intelligence Model for DDoS Detection in SDN Environment

YOUSIF AL-DUNAINAWI[1], BILAL R. AL-KASEEM[2], (Member, IEEE), AND HAMED S. AL-RAWESHIDY[3], (Senior Member, IEEE)

[1]Department of Cybersecurity Engineering, College of Engineering and Information Technology, AlShaab University, Baghdad 10001, Iraq
[2]Department of Communication Engineering, College of Engineering and Information Technology, AlShaab University, Baghdad 10001, Iraq
[3]Department of Electronic and Electrical Engineering, College of Engineering, Design and Physical Sciences, Brunel University London, UB8 3PH London, U.K.

Corresponding author: Hamed S. Al-Raweshidy (hamed.al-raweshidy@brunel.ac.uk)

ABSTRACT Distributed denial of service (DDoS) attacks continue to be a major security concern, threatening the availability and reliability of network services. Software-defined networking (SDN) has emerged as a promising solution to address this issue, enabling centralized network control and management. However, conventional SDN-based DDoS mitigation techniques often struggle to detect and mitigate sophisticated attacks due to their limited ability to analyze complex traffic patterns. This paper proposes an innovative and optimized approach that effectively combines mininet, Ryu controller, and one dimensional-convolutional neural network (1D-CNN) to detect and mitigate DDoS attacks in SDN environments. The proposed approach involves training the 1D-CNN model with labeled network traffic data to effectively identify abnormal patterns associated with DDoS attacks. Furthermore, seven hyperparameters of the trained 1D-CNN model were tuned using non-dominated sorting genetic algorithm II (NSGA-II) to achieve the best accuracy with minimum training time. Once the optimized 1D-CNN model detects an attack, the Ryu controller dynamically adapts the network policies and employs appropriate mitigation techniques to protect the network infrastructure. To evaluate the effectiveness of the optimized 1D-CNN model, extensive experiments were conducted using a simulated SDN environment with a realistic DDoS attack dataset. The experimental results demonstrate that the developed approach achieves significantly improved detection accuracy of 99.99% compared to other machine learning (ML) models. The NSGA-II enhances the optimized model accuracy with an improvement rate of 9.5%, 8%, 5.4%, and 2.6% when it is compared to logistic regression (LR), random forest (RF), support vector machine (SVM), and k-nearest neighbor (KNN) optimized models respectively. This research paves the way for future developments in leveraging deep learning (DL) driven techniques and SDN architectures to address evolving cybersecurity challenges.

INDEX TERMS Artificial intelligence, distributed denial of service, hyperparameters tuning, mininet, NSGA-II, optimized model, Ryu controller, software defined networking.

I. INTRODUCTION

Software-defined networking (SDN) is a networking approach that separates the control plane from the data plane in a network. The control plane is one of the core components responsible for managing and controlling the behavior of the network. It involves making decisions about how network traffic should be handled, how data should flow through the network, and how network resources should be allocated. The control plane essentially defines the rules and policies that govern the operation of the network. The functions of the control plane are shifted from individual networking devices to a centralized software application known as the SDN controller. The controller acts as a brain for the network, making high-level decisions and orchestrating the configuration of network devices. The data plane, also known as the forwarding plane, is responsible for the actual forwarding of network traffic, including data packets, from

The associate editor coordinating the review of this manuscript and approving it for publication was Salekul Islam.

source to destination based on the instructions received from the control plane. It is the operational layer of the network where the physical network devices (such as switches and routers) perform the essential task of transmitting packets. The data plane devices in an SDN architecture follow instructions provided by the centralized controller [1].

In traditional or legacy networks, network devices such as switches and routers have control planes and make independent forwarding decisions based on their routing tables. This feature can lead to inefficient use of network resources and difficulty in managing network configurations [2]. In contrast, SDN centralizes the control plane and enables network administrators to manage network traffic and configurations from a central location using software applications. These applications allow more efficient use of network resources, easier network management, and greater flexibility in adapting to changing network needs. Fig. 1 illustrates the legacy network and SDN architectures. Some of the key differences between SDN and legacy networks include [3], [4]:

1) Centralized control: In SDN, the control plane is centralized and managed by a software controller, whereas, in legacy networks, each network device has its own control plane.
2) Programmability: SDN networks are programmable, meaning network administrators can write applications to manage network traffic and configurations. In contrast, legacy networks are not as flexible and require manual configuration.
3) Virtualization: SDN enables network virtualization, which allows multiple virtual networks to run on a single physical network infrastructure. Network virtualization can improve network efficiency and reduce costs.
4) Open standards: SDN is based on open standards, meaning network devices from different vendors can work together seamlessly. The open standard is in contrast to legacy networks, which often require proprietary technologies that are specific to each vendor.

SDN is a rapidly evolving technology that is transforming how networks are managed and operated. Recent trends in SDN technology have focused on enhancing network performance, improving security, and increasing the scalability of networks. SDN is used to enhance network security by providing greater visibility into network traffic and enabling dynamic security policies that include using SDN to detect and mitigate distributed denial of service (DDoS) attacks and other types of cyber threats [4].

DDoS is a cyber attack involving overwhelming a network or server with traffic from multiple sources. A DDoS attack aims to disrupt or bring down the targeted network or server. DDoS attacks are typically carried out using a botnet, which is a network of compromised computers that are controlled by a single attacker. DDoS attacks can be very difficult to

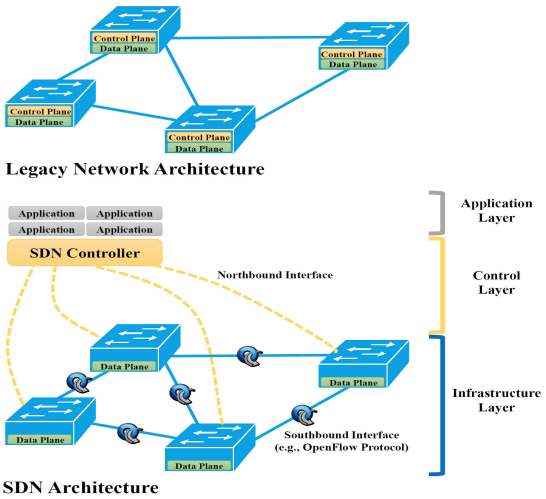


FIGURE 1. The legacy network and SDN architectures.

defend against and are a common threat to online services and businesses [5].

DDoS attacks significantly threaten network security, and SDN environments are not immune to them. In fact, SDN environments are particularly vulnerable to DDoS attacks because of their centralized control and programmability. A DDoS attack in an SDN environment can occur in several ways. Attackers may flood the network with a high volume of traffic to overwhelm the network and cause it to crash. Alternatively, they may exploit vulnerabilities in SDN controllers or switches to gain control of the network and launch DDoS attacks against other targets. The programmability of SDN also allows attackers to modify the network configuration to evade detection or launch more sophisticated attacks. For example, attackers may modify the flow rules in SDN switches to redirect traffic to a victim or bypass DDoS mitigation mechanisms. To prevent DDoS attacks in SDN environments, it is important to implement effective security measures [6].

Artificial Intelligence (AI) is a branch of computer science that focuses on creating intelligent machines that can learn and make decisions like humans. AI is used in various applications, from image recognition to robotics. Some of the popular AI technologies include machine learning (ML), deep learning (DL), and natural language processing (NLP) [7], [8]. AI, SDN, and DDoS are all related to the field of computer networking and security, and integrated them can be carried out in many ways [9], [10]:

- AI for DDoS detection and mitigation: AI can be used to detect and mitigate DDoS attacks by utilizing ML algorithms that can analyze network traffic patterns and identify abnormal traffic behavior that could indicate a DDoS attack. Once a DDoS attack is detected, an SDN controller can be used to reconfigure the network to block or mitigate the attack dynamically.
- SDN for AI-enabled network management: SDN can be used to provide a flexible and programmable network infrastructure that enables AI to manage

network resources. With SDN, network administrators can dynamically allocate network resources, prioritize traffic, and control network policies. AI can be used to optimize network performance, predict network failures, and automate network management tasks based on current traffic status.

- AI-enabled SDN for DDoS defense: AI can be used to enhance the effectiveness of SDN-based DDoS defense mechanisms. For example, AI algorithms can be used to automatically configure SDN-based DDoS mitigation policies based on real-time network traffic patterns. The automatic configuration can help to detect and mitigate DDoS attacks more quickly and efficiently without affecting the network performance.

The main contributions of this paper navigate the intricate intersection of DDoS attack mitigation and detection mechanisms within the context of SDN using AI techniques and can be summarized as follows:

1) Creating SDN simulated traffic dataset by utilizing mininet emulator. The created dataset launches DDoS flooding attacks for multiple protocols (i.e., user datagram protocol (UDP), Internet control message protocol (ICMP), and transmission control protocol (TCP)).
2) Developing a DL model based on one dimensional-convolutional neural network (1D-CNN) that employs a stacked convolution technique for improving the model's ability to extract complex features from the input data, handle variations, and prevent overfitting.
3) Optimizing the 1D-CNN model's hyperparameters using non-dominated sorting genetic algorithm II (NSGA-II) for achieving the best accuracy with minimum training time.

The rest of the paper is organized as follows: Section II reviews the recent state-of-the-art research that targeted the integration of AI to detect or mitigate DDoS attacks in the SDN environment. Section III summarizes the DDoS operation and its effects on network resources. While Section IV and Section V provide a clear explanation of the mininet emulator and the employed ryu controller. Section VI introduces the proposed approach steps in detail and highlights the main contributions of this work. On the other hand, Section VII discusses the obtained results from the developed approach. Finally, Section VIII concludes the presented work.

II. RELATED WORKS

The field of utilizing AI in the SDN environment has witnessed significant advancements in recent years, with numerous scholars and researchers investigating various aspects of the subject matter. This section provides a comprehensive review of the existing literature, highlighting the key studies, theories, and methodologies that have shaped the understanding of protecting the SDN environment from DDoS attacks. By examining the works of prominent researchers, this review aims to identify the gaps and limitations in the current knowledge, paving the way for the present study's contribution.

Swami et al. in [11] introduced a voting-based intrusion detection framework that could detect DDoS attacks and secure the SDN environment. The authors employed three ensemble classifiers called Voting-CKM, Voting-RKM, and Voting-CMN. While NSL-KDD, CICIDS2017, and UNSW-NB15 datasets were utilized for training and testing their developed models. The authors argued that their ensemble models had better performance when compared to other existing models with a classification accuracy of 99.68%, 97.77%, and 89.29% for Voting-CKM, Voting-RKM, and Voting-CMN, respectively.

Sahoo et al. in [12] utilized a support vector machine (SVM) as the main classifier for DDoS detection and mitigation in the SDN environment. The classifier was followed by kernel principal component analysis (KPCA) to perform feature selection. At the same time, a genetic algorithm (GA) was adopted to tune the SVM parameters to find their optimum values. The authors compared their developed approach to a single SVM classifier, and the results showed that their classifier achieved a high detection accuracy of 98.90%.

Zhijun et al. in [13] introduced a DDoS detection approach using factorization machines (FM) for low-rate DDoS attacks that targeted the data plane in SDN architecture. The obtained experimental results from their developed approach showed a moderate detection accuracy of 95.80% due to the fine-grained detection for low-rate DDoS attacks.

Ahuja et al. in [14] utilized ML techniques to classify the incoming traffic in an SDN environment into normal and DDoS attack traffic. The authors developed their hybrid model by integrating two classification algorithms: SVM and random forest (RF). The obtained results from their experimental scenario showed that the support vector classifier with random forest (SVC-RF) model achieved a high classification accuracy of 98.8% when a realistic SDN dataset was employed in training and testing their model.

Musumeci et al. in [15] developed an automated DDoS attacks detection (DAD) approach for SDN environment that aimed at detecting DDoS attacks in OpenFlow switch with the marginal intervention of SDN controllers. The authors introduced two versions of the DAD approach: standalone and correlated. These approaches were able to detect SYN flood attacks locally on the OpenFlow switch level to protect the SDN controller from DDoS attacks. The numerical results for the DAD model showed a classification accuracy of 98% for all tested ML algorithms (k-nearest neighbors (KNN), RF, SVM, and artificial neural network (ANN)).

Swami et al. in [16] presented a detailed analysis of various classifiers to detect the ingress DDoS attack traffic in an SDN environment. TCP-SYN flooding attacks were employed to study the effectiveness of the developed ML classifiers, such as decision tree (DT), multilayer perceptron (MLP), AdaBoost (AB), RF, and logistic regression (LR). The authors utilized a cross-validation technique to validate

the introduced classification models, and the obtained results from the conducted experiments showed that their introduced approach achieved high performance.

Sangodoyin et al. in [17] developed multiple ML models to detect and classify DDoS attacks against SDN. Their introduced SDN model had been emulated using mininet, while three DDoS flooding attacks were launched to study the effectiveness of their ML models. Four popular classifiers were employed by the authors (i.e., classification and regression tree (CART), KNN, Gaussian naïve Bayes (GNB), and quadratic discriminant analysis (QDA)), and the obtained results showed that the CART model outperformed other models with prediction accuracy of 98%.

Maheshwari et al. in [18] introduced an optimized weighted voting ensemble model that employed a hybrid metaheuristic optimization algorithm (BHO) to find the optimal set of the model's weights. Their introduced ensemble classifier was based on a couple of each of the following classifiers: SVM, RF, and gradient-boosted machines (GBM). The author argued that the employed six classifiers were differentiated by their hyperparameter values and eliminated false negatives through the introduced dynamic fitness function. The obtained results from their developed ensemble model showed that their model had high classification accuracy of 99.35% and 99.41% for CAIDA-2007 and CIC-DDoS-2019 datasets, respectively.

Alanazi et al. in [19] proposed an efficient DDoS detection approach for SDN environment using DL. The authors employed the CIC-IDS2017 dataset to train their classifier using long-short-term memory (LSTM), gated recurrent unity (GRU), and CNN to enhance the traffic classification. The authors argued that their proposed approach achieved a detection accuracy of 99.77% while a few features were adopted.

Mbasuva et al. in [20] proposed a DDoS detection approach for SDN environment that utilized recurrent neural network (RNN), deep neural network (DNN), and CNN. The authors employed the CIC-IDS2017 dataset to train their developed ensemble model, and they argued that their experimental results outperformed other ensemble classifiers (i.e., ensemble voting, ensemble RNN, and ensemble CNN) by achieving a high detection accuracy of 99.05%.

Liu et al. in [21] introduced a DDoS attack detection approach comprising information entropy and DL. The information entropy detected the spoofed switch ports, while the CNN model was used to differentiate between suspicious traffic and normal traffic. The authors argued that their proposed approach exhibited high detection accuracy of 98.98% when detecting DDoS attacks.

The main motivation behind the conducted research was that DDoS attacks are becoming increasingly sophisticated, making them challenging to detect using traditional methods. DL techniques have shown promise in capturing and learning intricate patterns in data, making them suitable for detecting the complex and evolving nature of DDoS attacks. The potential research gaps for developing an AI model to detect DDoS attacks, relative to existing works, could be: (i) none of the papers cited above considered the imbalance of classes in the employed dataset, and (ii) some of the papers cited above did not utilize SDN emulated traffic dataset. However, publicly accessible datasets created for legacy networks were not applicable in the SDN context. This work utilizes 1D-CNN that aims to leverage the strengths of DL, specifically in capturing complex patterns, automatic feature extraction, and adaptability. By harnessing these capabilities, the developed 1D-CNN model will enhance the accuracy, efficiency, and effectiveness of DDoS detection systems, ultimately improving the resilience and security of network infrastructures.

To the best of our knowledge, this paper underscores the significant contributions that have been made to the field of SDN security. Through meticulous investigation and rigorous analysis, this paper shed new light on DDoS detection in the SDN environment, providing valuable insights and advancing our understanding of the benefits of optimizing the hyperparameters of the 1D-CNN model. The findings of this paper pave the way for future investigations in this area.

III. SDN ENVIRONMENT UNDER DDoS ATTACK

DDoS attack is one of the most regular and devastating attacks against SDN. This kind of attack has an impact on programmable networks performance and its behavior. It stops or degrades network services by depleting the available resources, and hence, valid hosts cannot communicate with the SDN controller or deliver packets across the network [22].

In an SDN environment, a DDoS attack is accomplished by generating several new flows that overwhelm the SDN controller, the OpenFlow switches, and the secure channel, causing the network to go down for valid hosts. It is important to note that while some attack vectors are typical of traditional networks, programmable networks (e.g., SDN) have their own distinct set of dangers. For example, an attacker may take advantage of low-volume traffic flows rather than high-volume traffic flows to create a large number of ingress messages, which, in turn, would overwhelm the ingress switch and the controller. To be more explicit, attackers will produce numerous new flows that have faked IP addresses but will send them from several sources. A table-miss has occurred because these faked addresses do not match any of the current flow rules in the flow table of the OpenFlow switch.

Fig. 2 illustrates how the attacker strikes the SDN environment from many potential points of entry and the impacts of a DDoS attack in the SDN environment are [23], [24]:

- Impact on OpenFlow switch: DDoS attacks generate a massive volume of malicious traffic that floods the network infrastructure. OpenFlow switches, responsible for forwarding and controlling network flows in the SDN environment, may become overwhelmed by excessive traffic. These large number of malicious flows may flood the switch, resulting in the exhaustion of flow table

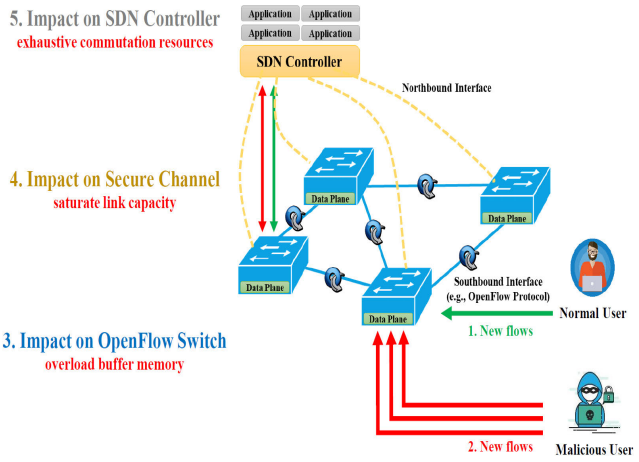


FIGURE 2. The impact of DDoS attack in SDN environment.

entries. Once the flow table reaches its capacity, the switch may either stop accepting new flows or evict existing flows, affecting the network's ability to process traffic effectively. Consequently, this can lead to network congestion, causing delays, packet loss, and overall performance degradation.

- Impact on the secure channel between SDN planes: The secure channel, which is responsible for transmitting control messages and instructions between the SDN planes, can be compromised or disrupted during a DDoS attack, leading to various consequences like (i) communication disruption between control and data planes, and (ii) increasing latency in the secure channel by consuming network resources that lead to increased packet loss and queuing delays.
- Impact on SDN controller: DDoS attacks can overwhelm the resources of the SDN controller, which is the central intelligence that manages and controls the network infrastructure, including CPU, memory, and network bandwidth. Massive malicious traffic can consume these resources, resulting in performance degradation. As a result, the controller may struggle to process control messages, make intelligent decisions, and effectively manage the network, leading to delays and unresponsiveness. Hence, critical network services and applications relying on the controller may experience downtime or degraded performance.

IV. MININET EMULATOR

Mininet is an open-source network emulator that allows developers and academic scholars to create a virtual network on their own computers or in the cloud. It enables developers to build, test, and debug complex network topologies and protocols without the need for physical hardware. Mininet is a powerful tool for network emulation and testing, with a wide range of features and capabilities that make it useful for research, education, and network development [25].

Various networks can be created with mininet using virtual hosts, switches, controllers, and links that behave just like a real network but without the expense or limitations of physical hardware. In addition, it provides an environment to run experiments, test different network topologies, and evaluate network performance under different conditions. Mininet supports various network protocols, including OpenFlow, and can be integrated with other network simulation tools and emulators. The architecture of mininet consists of the following components [25], [26], [27]:

- Hosts in mininet represent end devices such as computers or servers in a network. Each host is implemented as a Linux network namespace, providing a fully functional network stack. Hosts can run applications, generate traffic, and communicate with other hosts within the mininet network.
- Switches in mininet emulate OpenFlow switches, which are central to SDN. Each switch is implemented as a virtual Ethernet bridge and supports OpenFlow protocol for communication with the SDN controller. Switches in the mininet handle the forwarding of network traffic based on the flow table entries provided by the SDN controller.
- Controllers in mininet represent the SDN controller that is responsible for managing the network and providing instructions to the switches. Mininet supports various SDN controllers such as OpenDaylight, Ryu, and POX. The controller communicates with switches using the OpenFlow protocol to install flow table entries, handle network events, and control the behavior of the network.
- Links in mininet represent the virtual network connections between switches and hosts. They emulate network links and provide connectivity between different components within the mininet network. Links can have configurable characteristics such as bandwidth, delay, and packet loss, allowing users to simulate various network conditions.
- Custom network topologies in mininet can be created by users throughout the arrangement of switches, hosts, and links. Topologies range from simple linear or tree-like structures to more complex and realistic configurations. Mininet provides a flexible application programming interface (API) for defining topologies programmatically or using predefined topologies.
- Command line interface (CLI) and API in mininet allow users to manage and control the network interactively. Users can use the CLI to execute commands on hosts, view network configurations, and test network behavior. Additionally, mininet offers a Python API that enables programmatic control and automation of network setup and testing.

V. RYU CONTROLLER

An SDN controller is a software program that manages and directs network traffic flow in an SDN environment. The SDN controller manages the network and communicates with the switches that forward the traffic. It provides a single point of control for the network, allowing network

administrators to manage the network more easily and efficiently. The controller communicates with the switches using the OpenFlow protocol which is a standardized protocol used in SDN environments. The OpenFlow protocol allows the controller to program the switches to forward traffic in a specific way based on the network policies and rules defined by the administrator [28].

Ryu is an open-source SDN controller that provides a programmable feature to network infrastructure for cloud computing and data center environments. Ryu is written in Python and supports the OpenFlow protocol, which is widely used in SDN environments. The Ryu controller architecture is designed to provide a flexible and modular framework for building SDN applications. It is based on a component-based architecture, where each component is responsible for a specific function or feature. The key components of the Ryu controller architecture are [29], [30]:

- Ryu application framework provides a programming framework for building SDN applications. It is based on the Python programming language and provides a simple and easy-to-use API for developing SDN applications.
- Event manager is responsible for receiving and handling network events, such as switch connection/disconnection events, packet-in events, and flow modification events. It uses an event-driven programming model to handle events in real-time.
- The OpenFlow library is responsible for communicating with OpenFlow switches and other network devices. It provides a high-level abstraction of the OpenFlow protocol, allowing developers to write SDN applications without worrying about the protocol's low-level details.
- Network application manager is responsible for managing the lifecycle of SDN applications. It provides an interface for registering and unregistering applications with the controller and managing application dependencies.
- Representational state transfer (REST) API server provides a RESTful interface for interacting with the Ryu controller. It allows developers to interact with the controller using hypertext transfer protocol (HTTP) requests and provides access to the controller's features and functionality.
- Database is responsible for storing the configuration and state information of the controller. It provides a persistent storage mechanism for the controller, allowing it to recover from failures and restarts.

Ryu has gained popularity in the SDN community due to its simplicity, extensibility, and Python-based programming model. Researchers and developers widely use Ryu controller to build applications and network management solutions.

VI. PROPOSED APPROACH

This section presents an innovative and optimized approach that leverages the integration of the mininet network emulator, the Ryu controller, and the 1D-CNN model to tackle the challenge of detecting and mitigating DDoS attacks in SDN environments. The developed approach aims to enhance the network's ability to identify and respond to malicious traffic patterns in a timely and automated manner. By combining the flexibility and programmability of SDN, the realistic network emulation capabilities of mininet, the centralized control provided by the Ryu controller, and the DL capabilities of the 1D-CNN model, this work envisions a comprehensive solution that addresses the limitations of traditional DDoS mitigation techniques.

A. DATASET COLLECTION AND PREPROCESSING

This work utilizes a simulated approach to construct the whole dataset specifically targeted to capture the DDoS attacks in an SDN-based environment. There are many available datasets, but they lack diversity in attack types and are unrealistic. Therefore, training DL models will be difficult to capture real-world scenarios' behavior effectively. To address this issue, this work tried to generate various benign and malicious network traffic for three different protocols using various tools. Simulating benign traffic and DDoS attacks in mininet involve creating a virtual network environment that mimics real-world communication patterns between hosts and the characteristics of a real DDoS attacks. Fig. 3 depicts the simulated network architecture and how the dataset was collected. The dataset collection steps were identical except the tools that utilized during network traffic generation:

Step 1: Create Topology: Define the network topology using mininet's Python API. This Python program creates hosts, switches, and links to simulate the architecture of the employed network. The mininet provides a virtual network with multiple hosts, switches, and controllers. The employed topology comprises a single remote Ryu controller connected to eight OpenFlow switches that are connected to three hosts.

Step 2: Traffic Generation: To simulate normal traffic, various communication patterns have been executed between hosts that mimic real-world communication patterns, and this can be achieved by having hosts send requests to each other using common protocols like web servers might receive HTTP requests from clients, and clients might fetch data from servers. On the other hand, to simulate DDoS traffic, a high volume of traffic from multiple hosts should be generated using various tools like "*iperf*" and "*hping3*." A custom script was used to simulate attack traffic for ICMP, UDP, and TCP protocols.

Step 3: Monitoring and Logging: A standalone application runs inside the Ryu controller that monitors the global topology construction and then logs different OpenFlow switch flows statistics on a regular basis. The OpenFlow switch flow details and requests to the SDN controller were written into a comma-separated values (CSV) file. The collected dataset was annotated automatically using a simple

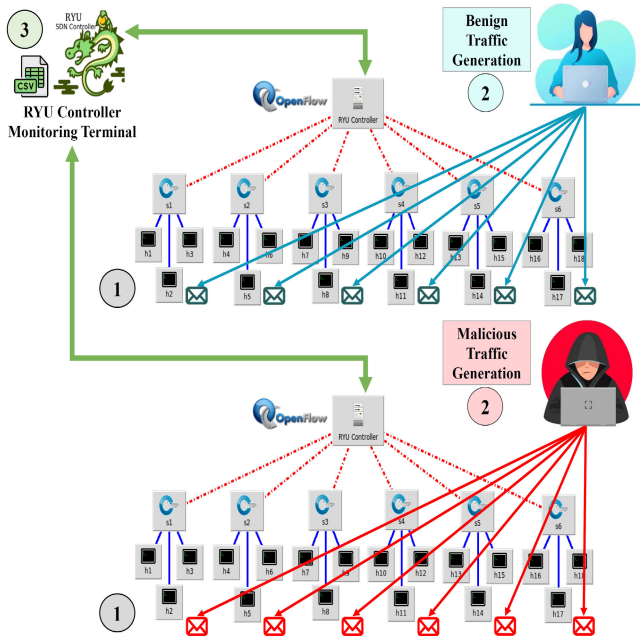


FIGURE 3. Dataset collection steps of the simulated network.

TABLE 1. The features of the employed dataset.

No.	Feature Name	Data Type
1	Switch ID	Numerical (integer)
2	Source IP	Categorical (object)
3	Destination IP	Categorical (object)
4	Packet Count	Numerical (integer)
5	Byte Count	Numerical (integer)
6	Duration	Numerical (integer)
7	Packet In	Numerical (integer)
8	Packet / Flow	Numerical (integer)
9	Byte / Flow	Numerical (integer)
10	Packet Rate	Numerical (integer)
11	Protocol	Categorical (object)
12	Port Number	Numerical (integer)
13	Transmitted Bytes	Numerical (integer)
14	Received Bytes	Numerical (integer)
15	Label	Numerical (integer)

code that set the label column of the dataset to "0" during the generation of benign traffic and set the label column of the dataset to "1" during the generation of malicious traffic.

Fig. 4 depicts the details of the proposed approach comprising three main phases: dataset collection, AI model training, and optimized model benchmarking.

Fig. 5(a) and Fig. 6(a) show the detailed statistics of the collected dataset in this work, with classes 0 and 1 corresponding to benign traffic and malicious traffic, respectively. Table 1 shows the features of the collected dataset that were employed to train the developed model.

The collected dataset should be preprocessed before the training phase of the optimized model begins because the raw data may have NaN/infinity values, errors, duplicate values, and missing values that certainly affect the accuracy of the AI model when classifying the ingress traffic. In this work, the below actions were performed during the preprocessing phase: (i) removing redundant entries, unchanged features, infinite values, and null; (ii) encoding categorical variables into a format that can be processed by the 1D-CNN model, preventing biases, and enhancing the model's ability to learn relevant patterns and features from the data; and (iii) normalizing the numeric features between 0 and 1 is a key preprocessing step that enhances the training efficiency, stability, and generalization capability of 1D-CNN models, leading to better overall performance.

The main observation from Fig. 5(a) and Fig. 6(a) is that the classes of the employed dataset are unbalanced. Therefore, the synthetic minority over-sampling technique (SMOTE) is utilized to improve the balance in dataset classes. The employed SMOTE algorithm consists of the following main steps [31]:

1) Split the dataset samples into minority class samples and majority class samples;
2) Find the center points of the minority sample distribution $\{O_1, O_2, \ldots, O_n\}$ for n regions that observed using k-clustering method, where $O_i = \frac{1}{m}\sum_{j=1}^{m} D_{ij}$ and D_{ij} is the jth point in ith region;
3) Calculate the closest distance d_i of all the majority sample points to the points O_i where $i = 1$ to $i = n$;
4) Calculate the distance dis for each minority class sample P to its corresponding center point;
5) Synthesis the new sample using $P_{new} = \eta P + (1-\eta)O_i$ when dis is less than its corresponding di and $0 < \eta < 1$. Otherwise, the point P is neglected;

In this work, two approaches have been implemented when applying SMOTE to the original dataset: (i) applying SMOTE on a protocol basis and (ii) applying SMOTE on a label basis. It is worth noting from Fig. 5(b) and Fig. 5(c) that both approaches above balanced the total number of classes in the dataset. However, applying SMOTE on a protocol basis achieves better balancing in the total number of class labels, as shown in Fig. 6(b) because each protocol will have equal value for benign and malicious traffic labels. On the other hand, applying SMOTE to the class label will not provide a balance at the protocol level as shown in Fig. 6(c) because the variance in protocol samples still exists as SMOTE did not pay any attention to the type of protocol when synthesis the new samples of the training dataset.

In summary, this paper adopted the utilization of the SMOTE approach on a protocol basis when generating the final dataset due to the following reasons:

- Increasing the representation of the minority class by generating synthetic samples because SMOTE balances the class distribution, allowing AI models to learn from both classes more effectively;
- Reducing bias towards the majority class and promoting fairer representation for both classes as a class imbalance can lead to biased models towards the majority class.

B. OPTIMIZED AI MODEL TRAINING

The dataset will be divided into 70% as a training dataset, and 30% will be divided evenly between validation and testing datasets. TensorFlow is an open-source ML framework

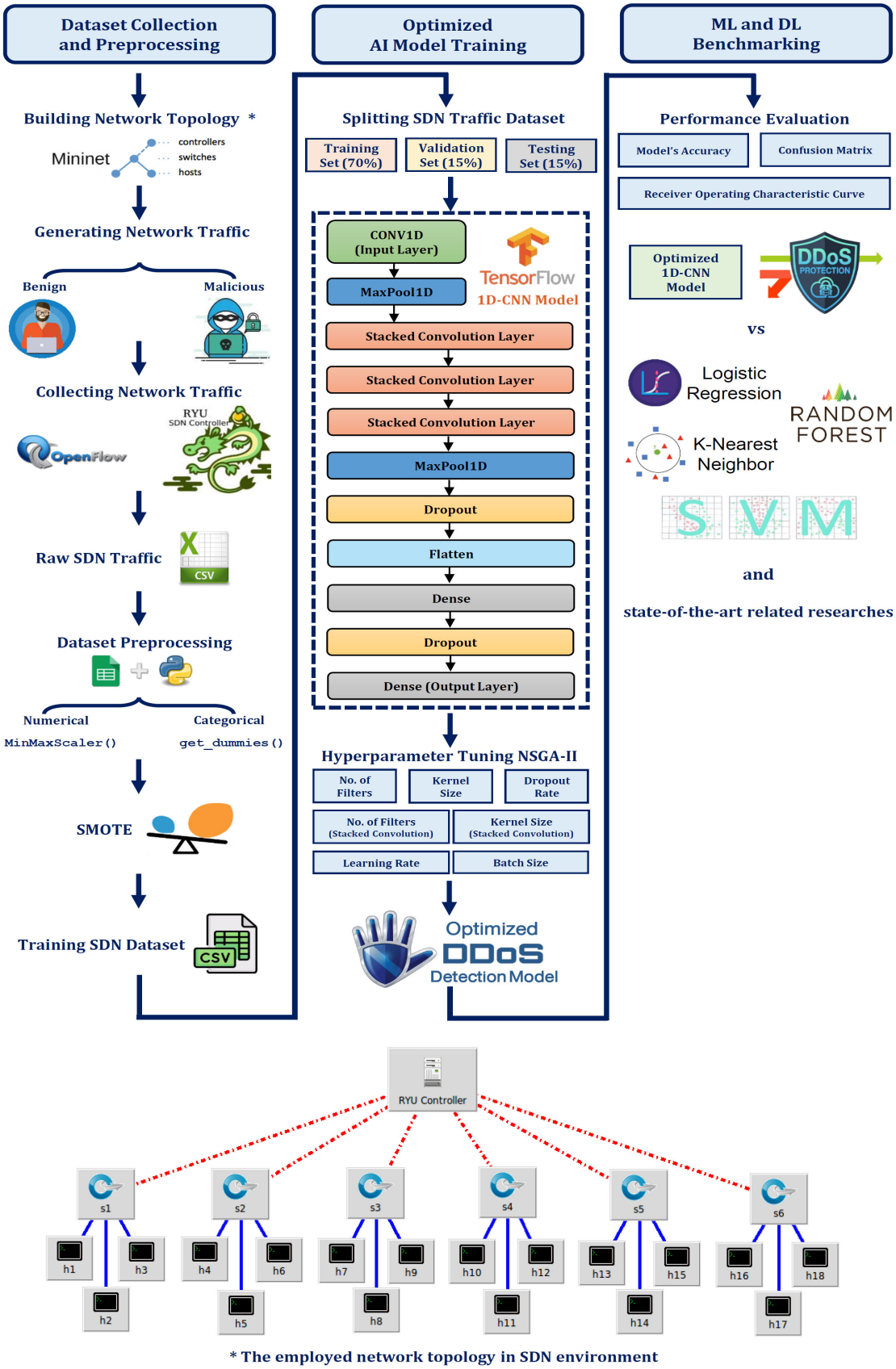
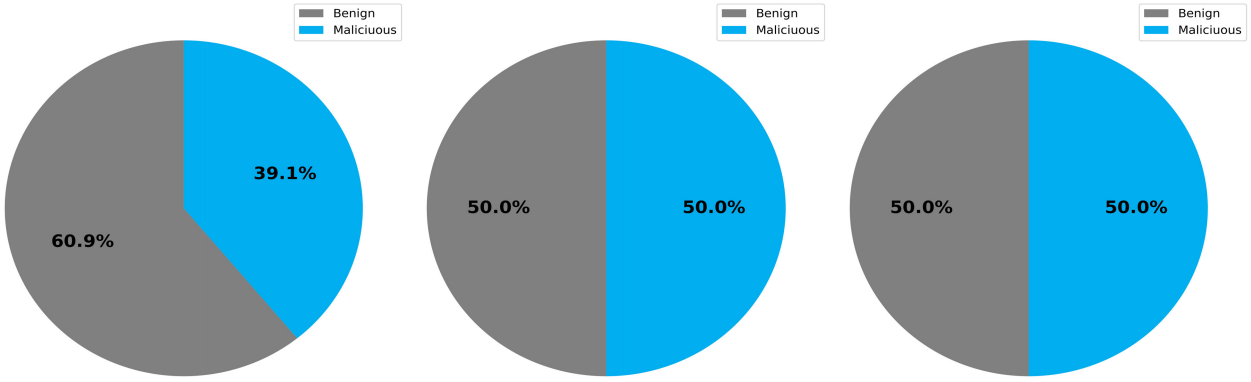
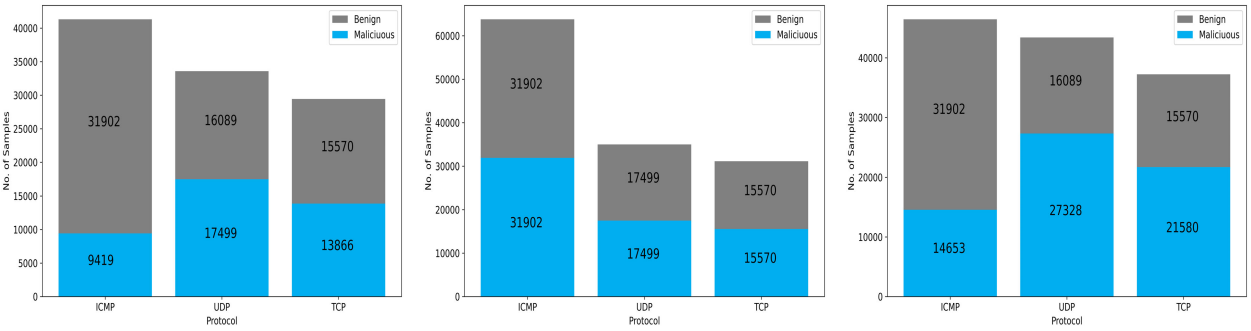


FIGURE 4. The proposed approach phases to develop the optimized DDoS detection model.



(a) Original dataset (b) SMOTE to individual protocol (c) SMOTE to whole dataset

FIGURE 5. The percentage of benign and malicious traffic in the collected dataset.



(a) Original dataset (b) SMOTE to individual protocol (c) SMOTE to whole dataset

FIGURE 6. The sample count for benign and malicious traffic in the collected dataset.

developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying various ML models. The 1D-CNN is a neural network architecture commonly used for analyzing sequential or time-series data. It is particularly effective in capturing local patterns and dependencies within the data. When using TensorFlow for 1D-CNN, developers can leverage the TensorFlow library's extensive capabilities to build, train, and deploy models efficiently. TensorFlow provides a high-level API called Keras, which simplifies the process of building neural networks, including 1D-CNNs. Keras provides a user-friendly interface for defining the model architecture, specifying the layers, and configuring hyperparameters. Once the 1D-CNN model is trained, TensorFlow offers tools for model evaluation, visualization, and deployment. Developers can export the trained model in a format compatible with different platforms and frameworks, allowing seamless integration into various applications.

1D-CNNs can capture local patterns and dependencies in the temporal dimension that are particularly relevant for DDoS detection, as certain attack patterns may manifest over time in the network traffic. By applying convolutional operations in the 1D-CNN, the model can extract relevant features from the temporal data, allowing it to identify suspicious patterns that indicate DDoS attacks. In an SDN environment, there is no need to reshape or transform the data into a one-dimensional format to reduce the input dimensionality and simplify the data preprocessing steps, making the detection process more efficient.

Two techniques were adopted in the developed model to improve its accuracy: stacked convolutional neural network (SCNN) [32] and early stopping (ES) [33]. SCNN involves stacking multiple convolutional layers in a 1D-CNN model that can improve model accuracy in several ways, including increased model capacity, non-linear transformations, and feature reusability. At the same time, ES involves monitoring the model's performance on a validation set and stopping the training early when the performance starts to deteriorate. The effects of ES on accuracy include preventing overfitting, finding optimal generalization point, and efficient model training.

Tuning the hyperparameters of a 1D-CNN is an important step in optimizing its performance and achieving the best possible results. Hyperparameters are configuration settings that define the behavior of a model and affect its performance. In the context of a 1D-CNN, the key hyperparameters include the number of employed filters and size of the employed kernel, the learning rate, batch size, activation functions, dropout rates, and the number of dense layers. Start by defining a search space for each hyperparameter, representing the range of values explored during the tuning process. The NSGA-II

approach was utilized in tuning seven hyperparameters of the developed 1D-CNN model: The number of employed filters in both convolutional and stacked convolutional layers, the size of employed filters in both convolutional and stacked convolutional layer, the learning rate, the dropout rate, and the batch size. The NSGA-II is a multi-objective optimization algorithm that uses a GA to optimize a set of objectives while ensuring that the solutions found are non-dominated, meaning no solution is better than another in all objectives simultaneously. NSGA-II is a favored choice for multi-criteria optimization due to its capacity to identify a diverse range of non-dominated solutions that capture trade-offs between conflicting objectives, enabling decision-makers to navigate complex decision landscapes and select from a variety of Pareto-optimal solutions tailored to their preferences and requirements. NSGA-II algorithm works by performing a series of steps in each generation [34]:

1) Initialization: Create an initial population of candidate solutions.
2) Evaluation: Evaluate each candidate's solution using the objectives of the problem.
3) Non-dominated sorting: sort the population into fronts based on the non-dominated relationships between solutions.
4) Crowding distance assignment: Assign a crowding distance value to each solution in each front, representing the density of solutions around that solution.
5) Selection: Select parents for the next generation using a combination of the non-dominated sorting and crowding distance assignment.
6) Crossover: Create new candidate solutions by recombining genetic information from the selected parents.
7) Mutation: Introduce small changes to the genetic information of the new candidate solutions to add diversity to the population.
8) Replacement: Replace the current population with the newly created candidate solutions.

The tournament selection was employed in the utilized NSGA-II. A random subset of individuals was selected in a tournament, and the one with the best fitness value (based on non-dominated sorting and crowding distance) was chosen as a parent. Tournament selection helped maintain diversity in the population and selected solutions with different trade-offs. At the same time, simulated binary crossover (SBX) was employed in the utilized NSGA-II. SBX emulated binary crossover in the continuous domain by mixing the genetic information of two parents based on a probability distribution. It created offspring that explore the search space around the parents, encouraging diversity and exploration with a crossover rate of 0.8. Finally, the polynomial mutation was adopted to perturb a decision variable by adding a small random value while ensuring the new value stayed within the variable's defined range. The polynomial mutation with a mutation rate of 0.1 was important for maintaining diversity and escaping local optima.

Multi-objective optimization problems, such as optimizing the hyperparameters of the ID-CNN model using NSGA-II, pose interesting challenges for researchers and academic scholars. These problems involve juggling multiple objectives that often compete with each other. This paper's proposed approach strives to maximize model accuracy while minimizing training time. However, finding the right balance between these objectives is not a simple task. It requires careful exploration of trade-offs and identifying the best set of hyperparameters that can deliver the desired performance.

Table 2 summarizes the decision variables of the introduced optimization problem and their search space with the obtained optimum value from the employed NSGA-II that systematically explores the search space by evaluating different combinations of values for the decision variables and adjusting them iteratively to find the best solution.

The first objective function is the model accuracy (MA). It aims to push the 1D-CNN model to perform at its best by training it with various hyperparameters and measuring the resulting validation accuracy. On the other hand, the second objective function is the training time (TT). It focuses on reducing the time it takes to train the 1D-CNN model. This time was measured by determining the time it takes to build the model using the chosen hyperparameters.

A fitness function (F) is formulated as a weighted sum to merge the above objective functions. It calculates the fitness value of a potential solution, represented by a set of hyperparameters, using the below equation:

$$F(x) = w_1 \times MA(x) + w_2 \times TT(x). \tag{1}$$

where $F(x)$ represents the fitness value of a candidate solution with hyperparameters x. By adjusting the weights w_1 and w_2, the right trade-off between maximizing model accuracy and minimizing training time can be achieved. It's like finding the sweet spot that meets our specific requirements.

The optimization process utilizing NSGA-II involves creating a population of potential solutions with different sets of hyperparameters presented in Table 2. They were then evaluating these solutions based on their fitness values. NSGA-II employs evolutionary operations such as selection, crossover, and mutation to evolve the population and uncover a diverse range of non-dominated solutions. These solutions represent different trade-offs between the objective functions.

C. ML AND DL BENCHMARKING

This benchmarking section provides a comprehensive evaluation and comparison of various ML techniques in the context of DDoS detection in the SDN environment. By conducting an extensive analysis, this paper aims to assess the performance and effectiveness of these techniques in relation to the existing body of work in the literature. The benchmarking process involves examining diverse state-of-the-art algorithms, including LR, RF, SVM, and KNN, by comparing their performance metrics, such as accuracy, sensitivity, specificity, precision, and F1-Score.

TABLE 2. The tuned hyperparameters (decision variable *x*).

No.	Hyperparameter	Search Space	Optimized Value
1	No. of Filters (Convolutional Layer)	16, 32, 64, 128, 256	64
2	Kernel Size (Convolutional Layer)	8, 16, 32, 64, 128	8
3	No. of Filters (Stack Convolutional Layer)	8, 16, 32, 64, 128	128
4	Kernel Size (Stack Convolutional Layer)	4, 8, 16, 32, 64	32
5	Dropout Rate	0.1, 0.2, 0.3, 0.4, 0.5	0.5
6	Learning Rate	0.0001, 0.001, 0.01, 0.1	0.001
7	Batch Size	100, 200, 300, 400, 500	200

Furthermore, by contrasting our findings with prior research, this paper aims to contribute to the ongoing efforts in advancing the field and identifying the most promising directions for future developments in DDoS detection and mitigation in the SDN environment.

Four classifiers have been adopted to study the effectiveness of the optimized model developed in this work. These classifiers have different strengths and are suitable for different data types and problem domains. The employed ML classifiers are:

- LR is a classification algorithm used to predict the probability of a binary or categorical outcome. It models the relationship between a set of input features and the probability of a particular class using a logistic function. LR is a linear model that applies a sigmoid function to the linear combination of input features to estimate the probability of the positive class [35].
- RF is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree is built on a random subset of the training data and a random subset of the input features. RF improves the accuracy and reduces overfitting by aggregating the predictions of multiple decision trees. It can handle both classification and regression tasks [36].
- SVM is a powerful classification algorithm that finds the best hyperplane to separate classes in a high-dimensional feature space. It works by mapping the input data into a higher-dimensional space and finding the hyperplane that maximizes the margin between the closest data points of different classes. It can handle linear and nonlinear classification tasks using different kernel functions [37].
- KNN is a simple and intuitive classification algorithm that classifies new data points based on the majority vote of their k-nearest neighbors in the feature space. It does not build a model but stores all the training instances as reference points. KNN determines the class of a new instance by comparing it to the k closest training instances based on Euclidean distance [38].

The system performance that utilized the optimized model has been evaluated using the dataset created in Section VI-A. The confusion matrix (CM) is a table that is commonly used to evaluate the performance of a classification model. It provides a summary of the predictions made by the model on a set of data points, comparing them to the actual labels of those data points. Receiver operating characteristic (ROC) is a graphical representation and evaluation metric used in ML to assess the performance of binary classification models. It plots the false positive rate (1-specificity) against the true positive rate (sensitivity) at various classification thresholds. ROC provides a way to observe how well the classifier separates the positive and negative classes across different thresholds.

VII. RESULTS AND DISCUSSIONS

The experimental scenarios of the developed approach were carried out using an HP All-in-One 24-df1xxx machine that equipped with Intel® Core™ i5-1135G7 processor and 8GB of random access memory (RAM). The HP machine runs Windows 10 as an operating system while the mininet software runs under a Linux virtual machine. The proposed approach tried to develop an optimized 1D-CNN model for SDN traffic classification to prevent DDoS attacks.

The optimized 1D-CNN model developed in this work utilized NSGA-II to tune its hyperparameters. The tuned hyperparameters are the number of employed filters in both convolution and stacked convolution layers, the size of the employed kernel in both convolution and stacked convolution layers, batch size, learning rate, and dropout. Fig. 7 shows the importance of hyperparameters to the optimized model accuracy. In summary, hyperparameters play a crucial role in determining the final accuracy of the trained model. Optimizing these hyperparameters through careful tuning and experimentation can lead to improved performance. Understanding the relationship between specific hyperparameters and accuracy and their impact on model complexity and generalization is essential for achieving the best possible model performance.

Fig. 8 shows the relationship between the model's hyperparameters and fitness function. The NSGA-II employed a multi-objective function that aimed at simultaneously optimizing multiple objectives or criteria, considering the trade-offs between them: the model's accuracy and the model's computation time (training time). The optimization process explores the hyperparameters search space to find diverse solutions that balance multiple decision variables. Once the experimental run finishes, the optimized model will have a set of hyperparameters with the optimized value that provides the best accuracy with minimum training time. It is clear that the NSGA-II maintains diversity in the population due to the adoption of the crowding distance technique as

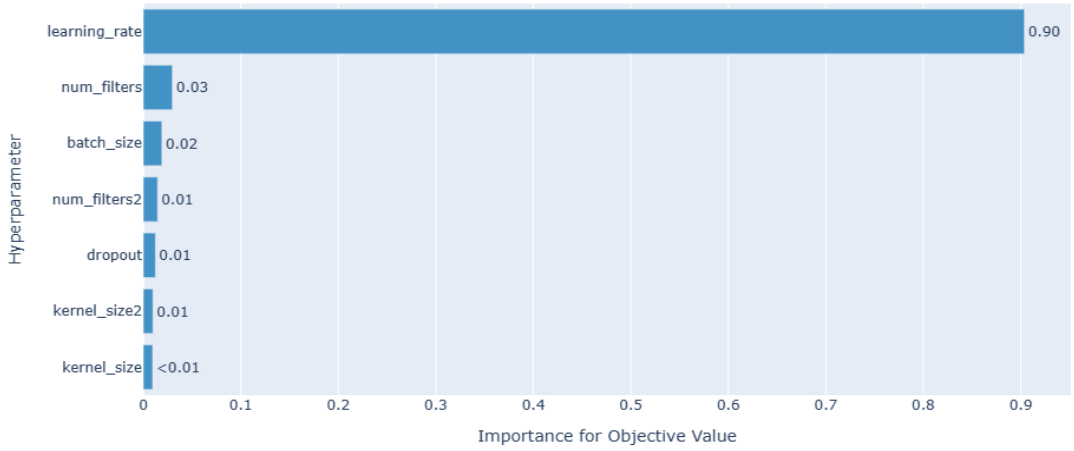


FIGURE 7. The importance of hyperparameters to the optimized model accuracy.

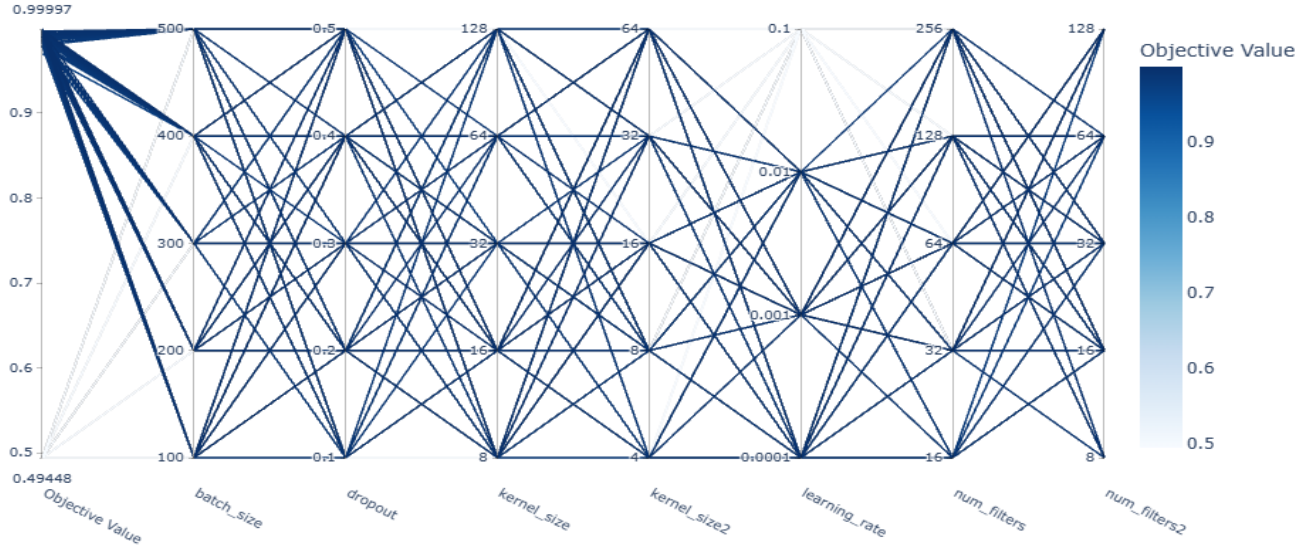


FIGURE 8. The relationship between the hyperparameters and the optimized model accuracy.

a selection criterion. As a result, the NSGA-II will select solutions that are not too similar to each other in the objective space.

Selecting the NSGA-II technique for hyperparameter tuning of 1D-CNN models can offer several compelling advantages that make it a suitable choice for the work presented in this paper: (i) NSGA-II excels in multi-objective optimization scenarios, enabling the exploration of trade-offs between conflicting objectives to find a set of Pareto-optimal solutions, (ii) NSGA-II explores a wide range of solutions across the Pareto front providing various options that cater to different priorities, (iii) NSGA-II's navigates complex and high-dimensional hyperparameter spaces makes it well-suited for optimizing hyperparameters in ML and DL scenarios, (iv) NSGA-II employs techniques like crowding distance to maintain diversity among solutions, striking a balance between exploration of the search space and exploitation of promising regions, and (v) NSGA-II's relatively straightforward implementation and intuitive parameter settings make it accessible to researchers from different backgrounds.

Fig. 9 depicts the visual representation of how the multi-objective function value evolves throughout the optimization process. Over 100 trials, the multi-objective function curve exhibits a steady improvement with minimum fluctuation around the initial value (reference point). It is clearly noted that the optimum values for the optimized 1D-CNN model's hyperparameters were reached in trial 33, and the multi-objective function value did not improve after that point.

Fig. 10 illustrates the learning curve in terms of the model's accuracy for the optimized 1D-CNN model. The training curve shows the optimized model's performance on the training data as the amount of training data increases. While the validation curve represents the optimized model's performance on a separate validation dataset, which contains samples not seen during training. The purpose of the

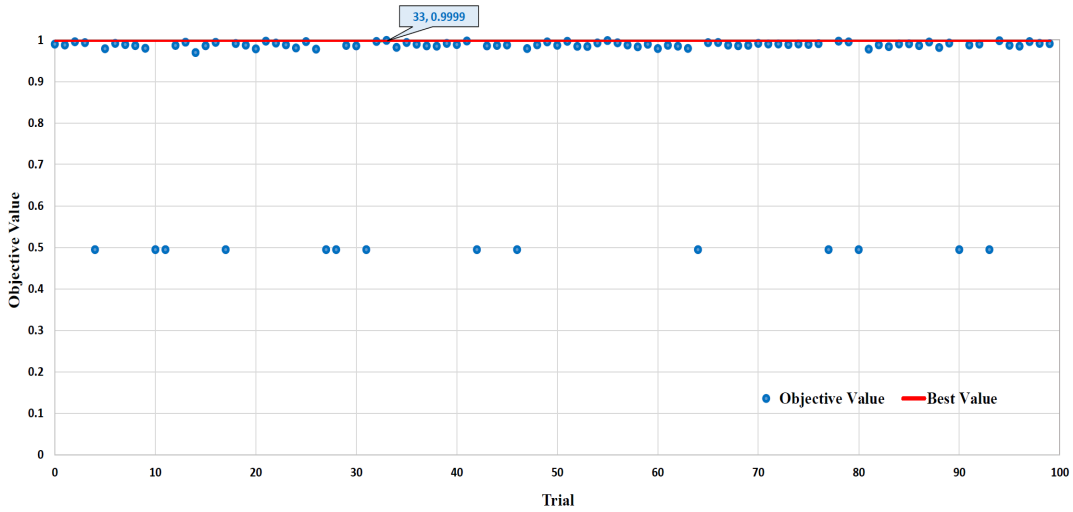


FIGURE 9. The objective function value vs optimization trials.

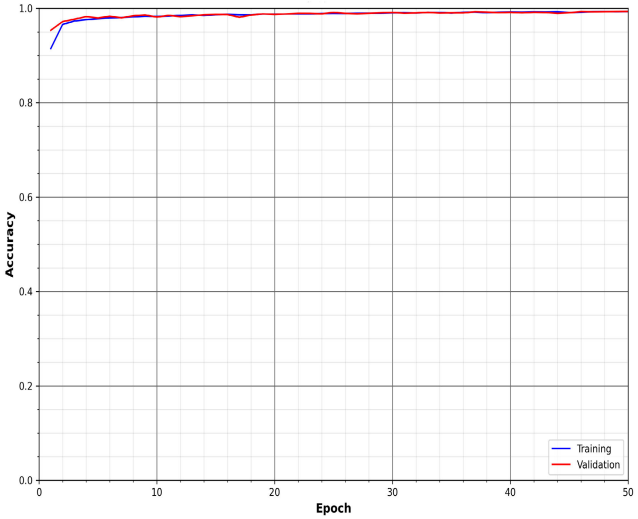


FIGURE 10. The accuracy of the optimized 1D-CNN model.

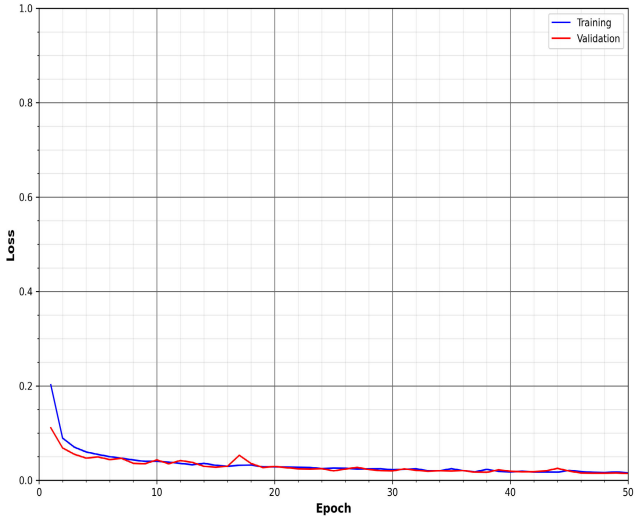
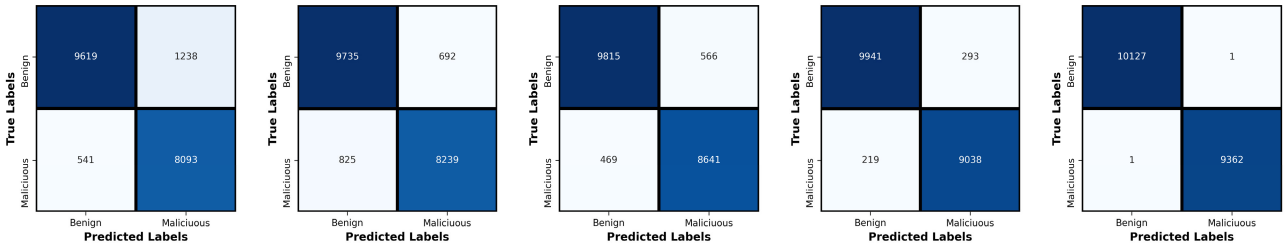


FIGURE 11. The loss of the optimized 1D-CNN model.

validation set is to evaluate the model's generalization ability. It is worth noting from Fig. 10 that the training and validation curves increased together as more data was added. This indicates that the model learns from the data and generalizes well to unseen samples. The narrow generalization gap indicates that the optimized model has a good ability to generalize its learning from the training data to unseen samples. In other words, the optimized model captures the underlying patterns of the data rather than memorizing specific instances. In addition, the narrow generalization gap indicates that the model has reached its optimal capacity with the available training data. Adding more data may not significantly improve its performance, suggesting that the model has learned the essential patterns and features necessary for accurate predictions.

Fig. 11 illustrates the learning curve regarding the model's loss for the optimized 1D-CNN model. The narrow generalization gap in the loss training curve indicates that the optimized model is able to minimize its loss not only on the training data but also on the validation data, indicating its ability to capture the underlying patterns and make accurate predictions. This implies that the model is not overfitting or underfitting but is achieving a reliable level of performance on unseen data. In summary, a loss training curve with a narrow generalization gap is a positive sign, reflecting the model's ability to be deployed in real-world scenarios.

In order to have a fair comparison, the benchmarking ML models (e.g., LR, RF, SVM, and KNN optimized models) have their hyperparameters tuned using NSGA-II. Fig. 12 visually represents the CM comparisons, highlighting the contrasts between the approach developed in this paper and other optimized ML models. Concurrently, Table 3 encapsulates a comprehensive summary of intricate comparisons among diverse optimized classifiers. Despite the LR classifier being widely employed in the literature of ML, however, it did not provide adequate accuracy as



(a) LR model (b) RF model (c) SVM model (d) K-NN model (e) Optimized model

FIGURE 12. The CM of the testing dataset to benchmark the developed optimized model vs other ML models.

TABLE 3. The detailed performance evaluation metrics for different classifiers vs the developed optimized model.

Performance Metric	LR Model	RF Model	SVM Model	KNN Model	Optimized Model
Accuracy (%)	90.87	92.22	94.69	97.37	99.99
Sensitivity	0.9468	0.9219	0.9544	0.9784	0.9999
Specificity	0.8673	0.9225	0.9385	0.9686	0.9999
Precision	0.8860	0.9336	0.9455	0.9714	0.9999
False Positive Rate	0.1327	0.0775	0.0615	0.0314	0.0001
False Negative Rate	0.0532	0.0781	0.0456	0.0216	0.0001
F1-Score	0.9154	0.9277	0.9499	0.9749	0.9999

shown in Table 3 because the LR model failed to capture the exact relationship between dataset features and dataset labels. Hence, LR could not classify the incoming traffic correctly. While various decision trees were adopted by the RF classifier in order to classify the class labels. Correct decisions made by other trees can balance out an incorrect decision resulting from one decision tree. The final class label can be obtained from the maximum votes provided by each decision tree. Therefore, the RF classifier provides better accuracy than the LR classifier. On the other hand, the SVM classifier provided moderate accuracy because it utilized support vectors to properly find a decision plane that had the greatest degree of separation between the two classes. The dataset features are strongly correlated; therefore, it may not always be able to get a perfect decision boundary without overfitting. Finally, the KNN classifier provided better accuracy than the above-mentioned ML classifiers as it employed a straightforward approach to classifying the ingress traffic using just elementary math. It is worth noting from Fig. 12 and Table 3 that the optimized 1D-CNN model outperformed all the optimized LR, RF, SVM and KNN models because the dataset has complex patterns and automatic feature extraction is important. The improvements in model accuracy were achieved by the NSGA-II approach that was employed to tune the hyperparameters of the 1D-CNN model and reach the optimum values for them. In addition, employing stacked convolution layers in the optimized 1D-CNN model can improve the model's ability to extract complex features from the input data, handle variations, and prevent overfitting during model training.

Fig. 13 depicted the ROC for the developed model versus other ML models. It is clear that the ROC of the optimized 1D-CNN model is closer to the top-left corner, indicating a better classifier with higher specificity and lower sensitivity when compared to optimized LR, RF, SVM, and KNN models.

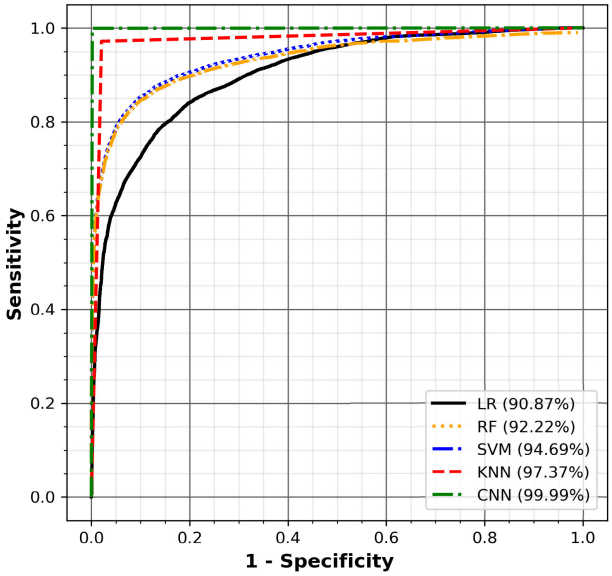


FIGURE 13. The ROC for the optimized model versus other ML models.

In the landscape of DDoS detection and mitigation within SDN environments, the proposed approach in this paper stands as an innovative stride, leveraging the power of DL techniques. The introduced approach addresses the evolving challenges of DDoS attacks and redefines the boundaries of efficacy in SDN security. A distinct differentiation emerges by juxtaposing the introduced method against the state-of-the-art solutions prevalent in the literature. In order to evaluate the effectiveness of the proposed approach, a comparison is made between the model developed in this work (optimized 1D-CNN) and the previous works that have already been done in the field of DDOS attack detection using a simulated dataset described in Section VI-A. As seen from Table 4, the optimized 1D-CNN model gets the maximum accuracy, which is 99.99%. The developed

TABLE 4. The comparisons with state-of-the-art works.

No.	Author Name	Year	Testing Accuracy
1	Sahoo *et al.* in [12]	2020	98.90%
2	Ahuja *et al.* in [14]	2021	98.80%
3	Musumeci *et al.* in [15]	2021	98.00%
4	Alanazi *et al.* in [19]	2022	99.77%
5	Liu *et al.* in [21]	2022	98.98%
6	Optimized Model	2023	99.99%

DL-based approach capitalizes on the inherent capacity of CNN to discern intricate patterns and anomalies that might elude conventional methods.

VIII. CONCLUSION

This paper presented a comprehensive literature review covering the state-of-the-art solutions to address DDoS attacks in the SDN environment. Simple network administration facilities could be achieved when utilizing SDN technology with more flexible tools as compared to legacy network technology. Various security concerns still exist when deploying SDN controller, and these issues need to be solved to improve network performance and increase security. This paper aimed to develop an optimized 1D-CNN model to classify the incoming traffic in an SDN environment and prevent DDoS attacks. Ryu controller was employed to monitor the network traffic generated in the mininet emulator. Extensive simulation scenarios were run through the utilization of OpenFlow switches and hosts. Once the simulation scenarios finished, the raw dataset was saved in a CSV file for further processing before training the developed model.

The developed approach enhanced the dataset through SMOTE in order to eliminate the effect of unbalanced classes in the collected dataset. The SMOTE was carried out at the protocol level rather than the label level to ensure that each type of traffic from different protocols had equal labels (benign and malicious). Then, the dataset was divided into 70% training, and 30% was divided evenly between validation and testing datasets. The developed 1D-CNN model was trained using NSGA-II to tune its hyperparameters for achieving the best accuracy with minimum training time.

Various ML algorithms were adopted to compare the developed approach among them, and other works existed in the literature to study the effectiveness of the optimized 1D-CNN model. The obtained results showed that the optimized 1D-CNN model outperformed the other ML models by 9.5%, 8%, 5.4%, and 2.6% when compared to LR, RF, SVM, and KNN models, respectively.

The developed methods of detecting and mitigating DDoS attacks in an SDN environment allow researchers and network security experts to test new detection and mitigation strategies without risking actual network infrastructure. This is particularly beneficial for fine-tuning algorithms, exploring different attack scenarios, and assessing the effectiveness of novel techniques. However, the simulated environments might not capture the complexities of real-world networks. The performance of detection and mitigation strategies in a controlled simulation might not directly translate to real-world effectiveness due to variations in network dynamics, traffic patterns, and the adaptability of attackers.

The employed dataset in this work provides controlled environments where the characteristics of attacks can be precisely defined. They are useful for benchmarking and comparing the performance of various detection and mitigation methods. They can also help in understanding how different attacks affect network behavior. However, the utilized simulated datasets aim to replicate real-world attack patterns; they might not fully capture the diversity and sophistication of actual DDoS attacks. Real-world DDoS attacks can evolve rapidly, employing new techniques that the simulated dataset might not cover. Additionally, the scale of real attacks can vary widely, from small-scale attacks to large and complex botnet-driven assaults, which might not be adequately represented in the simulated data.

REFERENCES

[1] W. Rafique, A. S. Hafid, and S. Cherkaoui, "Complementing IoT services using software-defined information centric networks: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23545–23569, Dec. 2022.

[2] Y. Maleh, M. Shojafar, A. Darwish, and A. Haqiq, Eds., *Cybersecurity and Privacy in Cyber-Physical Systems*. Boca Raton, FL, USA: CRC Press, May 2019.

[3] Y. Al Mtawa, A. Haque, and H. Lutfiyya, "Migrating from legacy to software defined networks: A network reliability perspective," *IEEE Trans. Rel.*, vol. 70, no. 4, pp. 1525–1541, Dec. 2021.

[4] T. Alharbi and M. Portmann, "The (In)Security of virtualization in software defined networks," *IEEE Access*, vol. 7, pp. 66584–66594, 2019.

[5] R. Jmal, W. Ghabri, R. Guesmi, B. M. Alshammari, A. S. Alshammari, and H. Alsaif, "Distributed blockchain-SDN secure IoT system based on ANN to mitigate DDoS attacks," *Appl. Sci.*, vol. 13, no. 8, p. 4953, Apr. 2023.

[6] L. F. Eliyan and R. Di Pietro, "DoS and DDoS attacks in software defined networks: A survey of existing solutions and research challenges," *Future Gener. Comput. Syst.*, vol. 122, pp. 149–171, Sep. 2021.

[7] O. Campesato, *Artificial Intelligence, Machine Learning, and Deep Learning*. Herndon, VA, USA: Mercury Learning and Information, 2020.

[8] R. Olusegun, T. Oladunni, H. Audu, Y. Houkpati, and S. Bengesi, "Text mining and emotion classification on monkeypox Twitter dataset: A deep learning-natural language processing (NLP) approach," *IEEE Access*, vol. 11, pp. 49882–49894, 2023.

[9] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3559–3570, Apr. 2020.

[10] J. Jiang, C. Lin, G. Han, A. M. Abu-Mahfouz, S. B. H. Shah, and M. Martínez-García, "How AI-enabled SDN technologies improve the security and functionality of industrial IoT network: Architectures, enabling technologies, and opportunities," *Digit. Commun. Netw.*, Jul. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822001420?via%3Dihub

[11] R. Swami, M. Dave, and V. Ranga, "Voting-based intrusion detection framework for securing software-defined networks," *Concurrency Comput., Pract. Exp.*, vol. 32, no. 24, Dec. 2020, Art. no. e5927.

[12] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, "An evolutionary SVM model for DDOS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020.

[13] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate DDoS attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17404–17418, 2020.

[14] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *J. Netw. Comput. Appl.*, vol. 187, Aug. 2021, Art. no. 103108.

[15] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled DDoS attacks detection in P4 programmable networks," *J. Netw. Syst. Manage.*, vol. 30, no. 1, p. 21, Nov. 2021.
[16] R. Swami, M. Dave, and V. Ranga, "Detection and analysis of TCP-SYN DDoS attack in software-defined networking," *Wireless Pers. Commun.*, vol. 118, no. 4, pp. 2295–2317, Feb. 2021.
[17] A. O. Sangodoyin, M. O. Akinsolu, P. Pillai, and V. Grout, "Detection and classification of DDoS flooding attacks on software-defined networks: A case study for the application of machine learning," *IEEE Access*, vol. 9, pp. 122495–122508, 2021.
[18] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, "An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment," *Microprocessors Microsyst.*, vol. 89, Mar. 2022, Art. no. 104412.
[19] F. Alanazi, K. Jambi, F. Eassa, M. Khemakhem, A. Basuhail, and K. Alsubhi, "Ensemble deep learning models for mitigating DDoS attack in software-defined network," *Intell. Autom. Soft Comput.*, vol. 33, no. 2, pp. 923–938, 2022.
[20] U. Mbasuva and G. L. Zodi, "Designing ensemble deep learning intrusion detection system for DDoS attacks in software defined networks," in *Proc. 16th Int. Conf. Ubiquitous Inf. Manage. Commun. (IMCOM)*, Jan. 2022, pp. 1–8.
[21] Y. Liu, T. Zhi, M. Shen, L. Wang, Y. Li, and M. Wan, "Software-defined DDoS detection with information entropy analysis and optimized deep learning," *Future Gener. Comput. Syst.*, vol. 129, pp. 99–114, Apr. 2022.
[22] P. M. Ombase, N. P. Kulkarni, S. T. Bagade, and A. V. Bagade, "Survey on DoS attack challenges in software defined networking," *Int. J. Comput. Appl.*, vol. 173, no. 2, pp. 19–25, Sep. 2017.
[23] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: Pros and cons," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 73–79, Jun. 2015.
[24] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.
[25] Mininet Project. (2012). *Mininet: An Instant Virtual Network on Your Laptop (or Other PC)*. Accessed: Jun. 2023. [Online]. Available: http://mininet.org/
[26] F. Keti and S. Askar, "Emulation of software defined networks using mininet in different simulation environments," in *Proc. 6th Int. Conf. Intell. Syst., Modelling Simulation*, Feb. 2015, pp. 205–210.
[27] J. Yan and D. Jin, "VT-Mininet: Virtual-time-enabled mininet for scalable and accurate software-define network emulation," in *Proc. 1st ACM Symp. Softw. Defined Netw. Res. (SIGCOMM)*, Jun. 2015, pp. 1–7.
[28] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN controllers: Benchmarking & performance evaluation," 2019, *arXiv:1902.04491*.
[29] Ryu Controller. (2012). *Ryu SDN Framework Community*. Accessed: Jun. 2023. [Online]. Available: https://ryu-sdn.org/
[30] S. Bhardwaj and S. N. Panda, "Performance evaluation using RYU SDN controller in software-defined networking environment," *Wireless Pers. Commun.*, vol. 122, no. 1, pp. 701–723, Aug. 2021.
[31] Y. Bao and S. Yang, "Two novel SMOTE methods for solving imbalanced classification problems," *IEEE Access*, vol. 11, pp. 5816–5823, 2023.
[32] S. Madichetty and M. Sridevi, "A stacked convolutional neural network for detecting the resource tweets during a disaster," *Multimedia Tools Appl.*, vol. 80, no. 3, pp. 3927–3949, Sep. 2020.
[33] M. V. Ferro, Y. D. Mosquera, F. J. R. Pena, and V. M. D. Bilbao, "Early stopping by correlating online indicators in neural networks," *Neural Netw.*, vol. 159, pp. 109–124, Feb. 2023.
[34] S. Verma, M. Pant, and V. Snasel, "A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems," *IEEE Access*, vol. 9, pp. 57757–57791, 2021.
[35] K. Yeturu, "Machine learning algorithms, applications, and practices in data science," in *Principles and Methods for Data Science* (Handbook of Statistics), vol. 43, A. S. S. Rao and C. Rao, Eds. Amsterdam, The Netherlands: Elsevier, 2020, ch. 3, pp. 81–206.
[36] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103160.
[37] H.-T. Thai, "Machine learning for structural engineering: A state-of-the-art review," *Structures*, vol. 38, pp. 448–491, Apr. 2022.
[38] Ö. Tonkal, H. Polat, E. Başaran, Z. Cömert, and R. Kocaoğlu, "Machine learning approach equipped with neighbourhood component analysis for DDoS attack detection in software-defined networking," *Electronics*, vol. 10, no. 11, p. 1227, May 2021.

YOUSIF AL-DUNAINAWI received the B.Sc. and M.Sc. degrees in mechatronics engineering from Baghdad University, Baghdad, Iraq, in 2004 and 2007, respectively, and the Ph.D. degree in intelligent and control systems from Brunel University London, U.K., in 2017. His current research interests include fuzzy logic, neural networks, and evolutionary based algorithms.

BILAL R. AL-KASEEM (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from Nahrain University, Baghdad, Iraq, in 2006 and 2010, respectively, and the Ph.D. degree in telecommunication and network engineering from Brunel University London, U.K., in 2017. His current research interests include wireless sensor networks, the design and implementation of smart sensors, software-defined networking, network functioning virtualization, M2M communication, cloud computing, network security, and the intelligent IoT applications.

HAMED S. AL-RAWESHIDY (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees from the University of Technology, Baghdad, Iraq, in 1977 and 1980, respectively, the master's Diploma degree from Glasgow University, Glasgow, U.K., in 1987, and the Ph.D. degree from Strathclyde University, Glasgow, in 1991. He was with the Space and Astronomy Research Centre, Baghdad, PerkinElmer, Waltham, MA, USA, British Telecom, London, U.K., Oxford University, Oxford, U.K., Manchester Metropolitan University, Manchester, U.K., and Kent University, Canterbury, U.K. He is currently the Director of the Wireless Network Communications Centre, Brunel University London.