International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I7P111

Volume 74 Issue 7, 160-178, July 2026
© 2026 Seventh Sense Research Group®

Original Article

Mitigation of DDoS Attacks in the Data Plane of Software-Defined Networking Using ML Techniques

Kamal Singh[1], Brijesh Kumar[2]

[1,2]*FET, Manav Rachna International Institute of Research and Studies, Faridabad, India.*

[1]*Corresponding Author : kamallohiaji@gmail.com*

Received: 27 February 2026 Revised: 12 June 2026 Accepted: 18 June 2026 Published: 28 July 2026

Abstract - *Distributed Denial-of-Service (DDoS) attacks remain one of the most significant cyber threats faced by Software-Defined Networking (SDN) architectures, essentially because of the salient decoupling of the control and data planes. This study examines the implications of DDoS attacks on the SDN data plane and evaluates the effectiveness of Machine Learning (ML) algorithms in detecting and addressing these attacks in real time. Using the Ryu controller, Mininet network emulator, and OpenFlow protocol, a realistic experimental environment was created to provide an accurate replica of the dynamic SDN behaviour under adverse circumstances. Empirical studies have demonstrated that distributed DDoS attacks, such as SYN, UDP, and ICMP flooding, substantially degrade network performance by reducing throughput, increasing packet loss, and exhausting switch flow table resources. To mitigate these effects, a suite of supervised machine learning classifiers, including Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naïve Bayes (NB), was instantiated and evaluated using traffic features captured on the emulated platform. The key performance indicators used to evaluate the classifiers included accuracy, precision, recall, and F1-score. The findings indicate that the Decision Tree and KNN models achieved detection rates above the 99% mark, with strong precision and recall scores, which in turn highlights their suitability for implementation in SDN-based security systems. This study provides experimental evidence that ML-based intrusion detection mechanisms can significantly enhance the resilience of SDNs to volumetric attacks. These results promote the implementation of adaptive and responsive security modules in SDN controllers, thereby increasing network resilience, particularly in large and dynamically programmable networks.*

Keywords - *SDN, DDoS Attacks, Machine Learning, OpenFlow, Ryu Controller, Intrusion Detection, Network Security.*

1. Introduction

SDN has become a widely adopted networking technology that separates the control plane from the data plane, thereby enabling centralised management and programmability of the network. This architectural shift has facilitated flexible traffic control, simplified configuration, and improved scalability. Consequently, SDN has been increasingly deployed across cloud computing environments, telecommunication systems, and enterprise networks.

Although SDN has its benefits, it is a distributed networking paradigm that depends on a logically centralised controller, creating a number of security vulnerabilities. The SDN controller maintains global flow tables and controls traffic forwarding by making control decisions centrally. Thus, if the functionality of the controller is disrupted, compromised, or overloaded, the overall network performance and reliability are significantly affected. This architectural feature makes SDN environments particularly vulnerable to DDoS attacks, as an adversary can attack the controller at a single point of failure.

DDoS attacks take advantage of the operational behaviour of SDN switches. If an OpenFlow switch receives a packet that does not have any entry in the flow table, it sends a packet-in message to the controller for processing. Attacks can be carried out by sending many packets with random header fields, causing switches to repeatedly query the controller.

This can overwhelm the controller resources, resulting in higher latency, lower throughput, and possible service interruption. DDoS attacks in the data plane can consume resources, such as OpenFlow switches, and make the network unresponsive in SDN. These effects may affect the performance of the network or render it unusable.

Although there is increasing research on the application of ML in DDoS detection for SDN, some gaps remain in the literature. Many studies have used obsolete benchmark datasets, such as KDD Cup 1999 and NSL-KDD [5, 21], which are not representative of modern attack traffic patterns. Others test their detection mechanisms on simplified or single-plane testbeds that are not fully representative of the

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

architectural separation between the data plane and the control plane, which is at the heart of SDN design [6, 15].

Although deep learning techniques are effective for detection, they are computationally intensive, making them impractical for use in SDN controllers with limited resources [9]. Moreover, few studies have assessed multiple ML classifiers under real-time conditions on a fully functional SDN testbed with live traffic captured from both SDN planes. These gaps hinder the use of existing approaches in real SDN deployments.

This study addresses these gaps by investigating DDoS attacks against the SDN data plane in a fully operational laboratory environment, in which the data and control planes are physically and logically separated as per the SDN architectural specification. Real-time traffic data were captured directly from the live testbed using the Ryu controller [2], Mininet emulator [3], and OpenFlow data-plane switches [4]. Unlike prior studies that depend on pre-recorded or synthetic traffic, the experimental data used in this study were produced under actual attack conditions on a functioning SDN. The classifiers were trained and evaluated on a realistic dataset of network configuration scenarios, CICDDoS2019, created using real network configurations and known attack tools [23].

The lab scenarios considered were SYN flood, UDP flood, and ICMP flood, where SYN flood uses the TCP handshake mechanism to consume connection resources, UDP flood targets hosts with high volumes of unsolicited packets, and ICMP flood consumes bandwidth by sending repeated echo request messages to the target hosts. Collectively, they address volumetric attacks, resource-exhaustion attacks, and provide a thorough evaluation of data-plane vulnerabilities in various attack scenarios.

In this context, the effectiveness of five machine learning classifiers in detecting and autonomously mitigating attacks at runtime in the SDN data plane was systematically evaluated.

The remainder of this paper is organised as follows: Section 2 presents the details of the SDN architecture, and Section 3 provides a review of related work. Section 4 presents a survey and taxonomy of ML approaches applied to DDoS mitigation in SDN.

The experimental setup and methodology are described in Section 5. Section 6 provides an overview of ML techniques for DDoS attack detection and mitigation. The data collection and preprocessing procedures are described in Section 7. The results and performance evaluation are provided in Section 8. Section 9 addresses the problems of deploying ML, such as scalability, latency, and retraining models. Section 10 summarises the study and proposes future research directions.

2. Background

2.1. SDN Architecture

The Open Networking Foundation (ONF) [1] defines the following layers in the SDN architecture:

2.1.1. Infrastructure Layer

This layer includes OpenFlow switches, which are responsible for forwarding packets following the flow-table entries provided by the controller. These flow tables are the matching criteria that enable packets to be directed in OpenFlow switches. In cases where the packets entering the switches do not have corresponding flow entries, the switches create packet-in messages to the controller, thus delegating the forwarding decision.

2.1.2. Control Layer

This layer contains the SDN controller, which serves as the network's centralised control function. The controller maintains a complete view of the network topology and manages the flow table entries of all OpenFlow switches. Controllers communicate with infrastructure devices via the southbound OpenFlow protocol, which manages network services to applications through northbound APIs, such as Representational State Transfer (REST) APIs.

The controller processes packet-in messages, computes optimal forwarding paths, and instals flow rules in switch flow tables. Popular controller implementations include Ryu, Open Daylight (ODL), and Open Network Operating System (ONOS).

2.1.3. Application Layer

This layer includes applications that interact with the controller via northbound APIs to request network services, such as traffic engineering policies and security controls, or to perform network analytics. Common SDN applications include load balancers, firewalls, traffic monitoring systems, and Quality of Service (QoS) provisioning modules.

2.2. SDN Controller

The SDN controller is a centralised network control system that provides network resource abstractions and centralised control logic. Controllers are used to control the network state, provide forwarding paths, and implement policies among distributed network devices. The Ryu controller, which is used in this study, is a Python framework that provides large-scale support for OpenFlow protocols and enables the quick development of network applications.

2.2.1. Southbound API

This interface enables communication between the controller and network infrastructure devices. The OpenFlow protocol serves as the main southbound API for the flow rules, topology discovery, statistical collection, and packet-forwarding instructions. Through the southbound interfaces,

controllers can program switch behaviour, monitor network state, and respond to network events.

2.2.2. Northbound API

This interface provides programmatic access to network services through applications located within the application layer. Northbound APIs offer RESTful web services that allow applications to query network topology, seek bandwidth assignment, issue security policies, and apply tailored traffic engineering policies.

2.2.3. East/Westbound API

These interfaces are used to allow inter-controller communication when deploying distributed controllers. In the case of scalable networks, single-controller networks have limited scalability and reliability. East/westbound interfaces provide state synchronisation and controller clustering, which provide horizontal scalability and fault tolerance.

2.2.4. Security Module

The SDN controller uses security modules to encrypt the communication channels of network devices before establishing control connections and to manage application privileges with role-based access control. It uses Transport Layer Security (TLS) to keep communication between network devices private and secure.

2.2.5. Management Module

This module provides statistics about the flow installation rate, controller CPU usage, memory usage, and network statistics. These features enable network operators to track the health of the controller and discover performance problems.

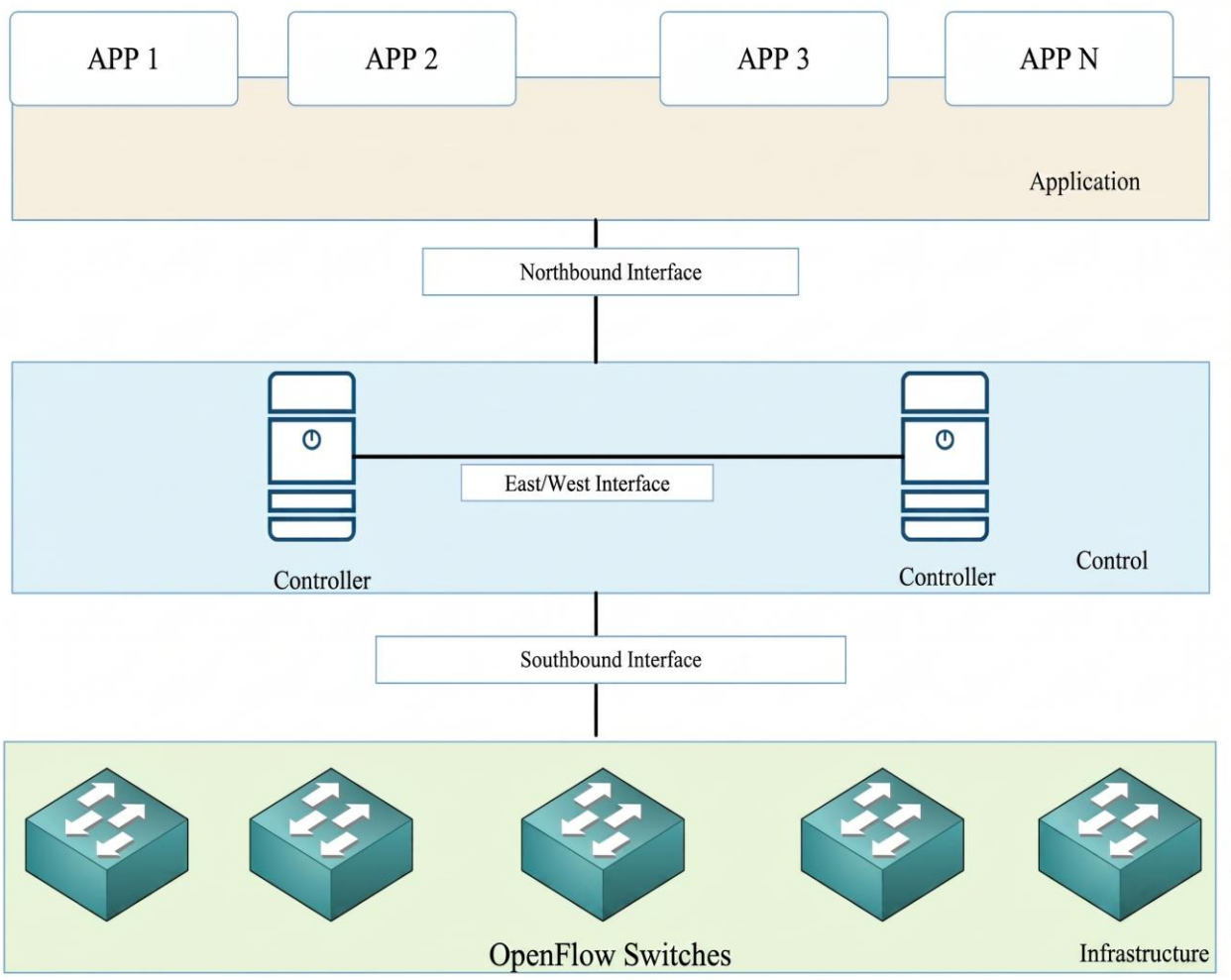

Fig. 1 SDN ONF architecture

3. Literature Review

SDN DDoS attacks have garnered considerable academic attention because of the distinct security challenges of centralised architectures. Modern DDoS detection and mitigation plans in SDN are critically evaluated, including machine-learning approaches, network intrusion systems, and hybrid approaches.

Etedali and Noorifard [5] also suggested a framework of supervised learning that uses CatBoost to identify DDoS incursions in an SDN environment controlled by the Ryu controller. It uses gradient-boosted decision trees, which are trained using the KDD Cup 1999 and NSL-KDD datasets, and uses features extracted from OpenFlow switch statistics to categorise network flows in real time. The framework has strong detection capabilities; however, its reliance on outdated datasets can limit its effectiveness against modern attack vectors. A detection system that uses machine learning, including feature preprocessing and ranking based on the importance measurements of Random Forest, was developed by Chaturvedi and Bishnoi [6]. Some of the classifiers considered in the system include logistic regression, support vector machines, and ensemble algorithms. Feature selection techniques minimise the amount of computation without affecting the detection performance. Although the system is satisfactory under diverse attack conditions, it should be evaluated further to determine its performance in high-traffic SDN settings.

Zheng et al., [27] proposed a multistage system that identifies low-rate denial of service attacks because such attacks do not leave a substantial signature when identified using typical detection systems, as they can be seen as a fine perturbation of network behaviour. They combined sliding-window monitoring with XGBoost-based classification and utilised dynamic time warping to localise malicious sources. The proposed system will solve the drawback of threshold-based mechanisms that do not often consider low-rate attacks, which can nonetheless hinder the performance of the network [16].

Keshari et al., [8] tested the behaviour of SDN controllers in the DDoS environment, especially comparing POX and RYU. The findings show that RYU provides better throughput and recovers faster after an attack. These results inform controller selection for security-sensitive applications, although architectural differences limit direct comparison. Pericherla et al., [18] demonstrated that early detection of DDoS attacks in SDN using ML techniques can significantly reduce the impact on network performance.

Yang et al. [9] examined a combination of convolutional neural networks and LightGBM optimisers to detect online DDoS. The method uses Borderline-SMOTE to counteract the imbalance of the classes. Although deep learning methods have high detection limits, they have much more resource-intensive computational requirements, which may limit their application in resource-limited settings.

Zakaria et al., [11] explored the advantages of intrusion detection with isolation forests and utilised unsupervised learning to detect irregular traffic patterns. Their model integrates scalable traffic gathering with feature selection to reduce the overhead of SDN controllers. Although unsupervised techniques are known to be efficient in the detection of novel threats, they might have a higher false-positive rate than supervised techniques. de Santos and Clayton [13] extended this with a stream-based technique to reduce false positives while enabling continuous monitoring, thereby improving deployment suitability on operational networks. Alhamami and Albermany [14] compared different statistical and machine learning approaches across heterogeneous SDN testbeds. Yang and Zhao [15] built a machine-learning application for campus-wide SDN implementations, using support-vector machines to classify attacks and apply adaptive flow policies. Their system has a practical feasibility of 99.8% in terms of the detection rate. However, the findings made in a campus setting cannot be expected to generalise to a broader or adversarial deployment environment. Similarly, Tahirou et al., [17] were able to show that SVMs achieve both high accuracy and low detection latency on the CICDDoS2019 dataset, which concentrated on control-plane attacks, a severe vulnerability in SDN, because controller congestion can cause network failure.

Sharma and Babbar [19] compared hybrid detection schemes that integrate random forest with logistic regression models to IoT-integrated SDN networks and used the Edge-IIoT dataset, attaining an accuracy of 99.10%. IoT-based scenarios are also more complex because of the abundance of connected devices and lack of resources. Sanapala et al., [20] also implemented decision-tree classifiers to improve IoT security in SDN models.

Kavitha et al., [21] compared a number of models such as KNN, logistic regression and decision trees using the KDD Cup 1999 data set; decision trees achieved more than 99% accuracy. Wan and Yang [22] have described that hybrid random-forest-svm models improve the level of detection resilience to a range of DDoS attack variants.

The studies considered cover a broad range of DDoS detection methods in SDN, including machine learning classifiers, deep learning models, and anomaly-based systems. Supervised learning often produces a high recovery rate; however, it requires the use of labelled datasets and can ignore new attacks. The opposite is true for unsupervised methods, which are more lenient in detecting unidentified threats with a higher false-positive rate. This study systematically evaluates numerous machine-learning algorithms in controlled circumstances, assesses their suitability for real-time SDN defence, and suggests practical countermeasure strategies. Gradient boosting methods have been investigated for the detection of DDoS attacks in SDNs to improve controller security.

A supervised learning approach is used to classify benign and malicious traffic using the CatBoost classifier in an SDN environment managed by the Ryu controller. The model was trained with benchmark datasets such as KDD Cup 1999 and

NSL-KDD, which allows it to deal with categorical features without extensive preprocessing and makes the model work effectively. Real-time traffic analysis is carried out by extracting flow-level attributes at the controller and dynamically categorising packets to trigger mitigation action. Experimental results show excellent detection accuracy in both offline and live SDN emulations, which proves the feasibility of embedding CatBoost-based intrusion detection into SDN security pipelines [5].

A hybrid deep learning approach using a circle search algorithm and a Convolutional Neural Network (CNN) is proposed to improve the ability of DDoS detection in SDN environments. The method consists of label encoding and feature selection with a circle search algorithm to reduce dimensionality and detect malicious traffic patterns, and classification with a CNN, which captures the spatial correlation between network flows. Evaluation on the CIC-DDoS2019 dataset shows that it achieves high precision, sensitivity, and overall detection accuracy compared with GRU-based and entropy-driven methods [12].

The framework can effectively detect known and unknown attacks via global–local optimisation using deep learning methods, allowing for scalable and real-time SDN security deployments [25]. There have been extensive and detailed security analyses of SDN environments that identify the vulnerabilities of the central control and programmable interfaces of the SDN. Previous studies have systemically identified vulnerabilities across the application, control, and data planes, including controller compromise, DDoS attacks, flow rule manipulation, and insecure northbound and southbound APIs. A range of secure communication protocols, controller redundancy, access control, and real-time monitoring techniques have been proposed to minimise these risks. Recent developments that enhance SDN resilience are based on machine learning anomaly detection and blockchain security architecture.

All these developments have been made; however, there are still open research directions to be explored, such as scalability, interlayer trust, and complete threat modelling [26]. Low-rate DoS attacks are a serious concern in SDN networks because they have stealthy traffic patterns and are difficult to detect by volume. A new defence framework using SDN has recently been proposed with a three-stage mechanism, which includes periodic traffic monitoring, sliding window analysis, and machine-learning-based attack detection, to accurately detect LDoS attacks. It uses the statistical features of both TCP and UDP traffic streams to classify traffic and employs dynamic time warping for localisation and blocking malicious traffic flows by controller-driven rule installation. Experimental results show that SDN-assisted intelligent defence mechanisms can effectively mitigate LDoS attacks within a few seconds under various traffic scenarios, which indicates their applicability to tackle sophisticated LDoS attacks [7, 27]. A packet-in-flooding attack has been proposed to be mitigated using a data-plane-centric defence model in SDN environments [10]. The idea is to keep track of the number of different destinations reached by a given source within a given time window and alert suspicious activity if a threshold is exceeded, thus providing direct switch-level detection of attacks.

This method moves the detection from the controller to the data plane, which significantly alleviates the workload of the controller and guarantees a fast response time. The solution ensures that network performance is not degraded during detection, and the round-trip time is under 1 ms even with the presence of packet-in-driven DDoS attacks, as shown in the experimental evaluation conducted in a Mininet-based setup [28] that proves the effectiveness of lightweight edge-based mitigation for packet-in-driven DDoS attacks in SDN networks.

The surveyed literature has not explored other emerging paradigms of DDoS mitigation in SDN, such as federated learning or online learning. In multi-domain deployments, federated learning allows the distributed training of detection models without centralising raw traffic data to resolve privacy issues. Online learning techniques enable classifiers to learn incrementally with each arrival of a new traffic sample; therefore, they are less reliant on periodic full retraining. These approaches are more relevant to large-scale SDN deployments with dynamic traffic patterns and limited data sharing between administrative domains. In the present work, supervised batch learning is used on a representative labelled dataset, and federated and online learning extensions are suggested for future research.

4. Survey and Taxonomy of Ml-Based DDoS Mitigation in SDN

Machine learning approaches applied to DDoS detection and mitigation in SDN environments can be broadly classified into three categories: supervised, unsupervised, and deep and hybrid learning. Each category differs in its training paradigm, data requirements, and suitability for real-time SDN deployment. Table 1 summarises the taxonomy of ML approaches identified across the surveyed literature.

4.1. Supervised Learning

Supervised learning is the most common paradigm used for DDoS detection in SDN environments. They rely on labelled training data, which include benign and attack traffic samples, and learn decision boundaries between input flow features and output class labels. All of these classifiers are part of this category, which are the classifiers used in this study: Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, and Naive Bayes. Supervised approaches such as Gradient Boosting (CatBoost [5] and XGBoost [27]) have also been used in this context, and they perform well on benchmark datasets for DDoS detection.

To make the detection more resilient to different types of attacks, hybrid classifiers that combine multiple supervised classifiers have been investigated, such as Random Forest and logistic regression [19] and Random Forest and SVM [22]. The key benefit of supervised learning is its ability to achieve high detection accuracy with representative labelled data; however, it is limited by the need for labelled data and can be less effective in detecting new attack patterns.

4.2. Unsupervised Learning

Unsupervised learning techniques use no labelled training data and instead detect anomalous traffic patterns using statistical variations from baseline traffic. Zakaria et al., [11] studied isolation forests for intrusion detection in SDN, which uses unsupervised anomaly detection to identify irregular traffic without the need for attack signatures. Other unsupervised methods based on streams have been studied to allow for continuous monitoring at lower false-positive rates [13].

The main benefit of unsupervised methods is their ability to identify new or unknown attack types that are not included in the labelled training sets. As mentioned in the surveyed literature, unsupervised approaches have higher false-positive rates than supervised approaches, which can cause an operational burden for security analysts and SDN controller resources. Unsupervised methods are most appropriate when labelled data are unavailable or when detecting zero-day attacks is a priority.

4.3. Deep Learning and Hybrid Approaches

Deep learning techniques, such as CNN and gradient-boosted ensemble models, have been used to detect DDoS in SDN with high accuracy. The CNN-LightGBM model with Borderline-SMOTE was used to solve the class imbalance problem and achieved good detection performance [9].

Padmakala et al., [25] presented a hybrid approach, which is a combination of the circle search algorithm and CNN, to select features and classify the CICDDoS2019 dataset. Deep learning methods can achieve high detection limits; however, their computational requirements are much higher than those of traditional supervised classifiers, which can limit their usage in real-time SDN controllers with limited resources. A new trend in the SDN security field is the use of hybrid approaches that combine deep learning with traditional ML classifiers or optimisation algorithms, which seek to combine accuracy with deployment feasibility.

4.4. Taxonomy Summary

This study is based on supervised learning and utilises five classical classifiers that are tested in real-time SDN lab environments. The taxonomy shows that unsupervised and deep learning approaches complement each other in detection capabilities; however, supervised classifiers are more suitable for real-time deployment in resource-constrained SDN data plane environments owing to their low inference latency and high detection accuracy on representative labelled datasets, such as CICDDoS2019, and their interpretability.

Table 1. Taxonomy of ML approaches for DDoS detection in SDN

Category	Methods / Algorithms	Key References	Advantages	Limitations
Supervised Learning	DT, RF, KNN, SVM, NB, CatBoost, XGBoost	[5, 6, 15, 19, 21, 22, 27]	High accuracy; interpretable; fast inference	Requires labelled data; limited against novel attacks
Unsupervised Learning	Isolation Forest, Anomaly Detection, Stream-based	[11, 13]	No labelled data needed; detects novel attacks	Higher false positive rate; harder to tune
Deep / Hybrid Learning	CNN, LightGBM, CSA-CNN, En-CNN + OptiLGBM	[9, 25]	High accuracy; captures complex patterns	High computational cost; resource-constrained SDN controllers

5. Methodology and Experimental Setup

5.1. Experimental Environment

This testbed was designed to emulate realistic SDN deployments and attack scenarios on a data plane in real-time. The testbed is based on the Ryu controller framework version 4.34, which is installed on Ubuntu Linux 20.04 LTS and supports a wide range of OpenFlow 1.3 protocol features, as well as the development of network applications. The network topology consists of six OpenFlow switches interconnected in the network to support multipath routing and the simulation of distributed attack vectors. There were three hosts per OpenFlow switch and 18 hosts across the various OpenFlow switches. In this topology, multiple sources can be attacked simultaneously, and network connectivity can be maintained to observe the legitimate traffic. The Mininet network emulator version 2.3.0 is a lightweight virtualisation of network components, allowing repeatable experimental conditions without the need to deploy physical hardware. System resource utilisation was continuously monitored during the attack scenarios, and controller performance degradation was also monitored. The collected metrics include controller CPU utilisation percentage, memory consumption

in megabytes, packet processing throughput measured in packets per second, and the flow table of all switches. Data were collected from the Ryu controller via OpenFlow protocol messages and from the controller itself using system utilities.

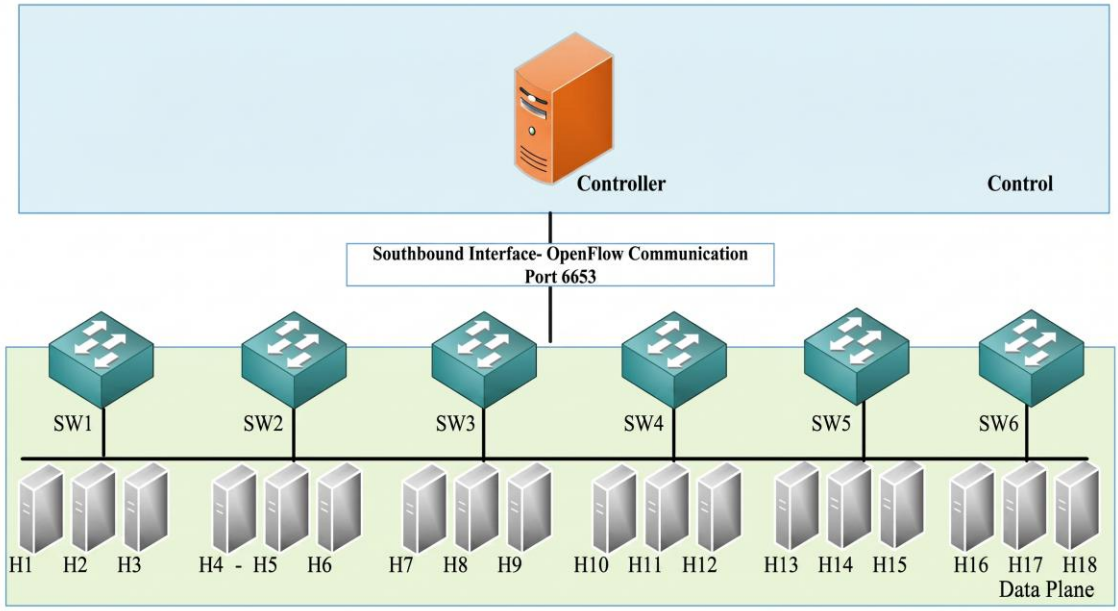

Fig. 2 Network topology

5.2. Attack Scenarios

Three separate DDoS attack scenarios were designed to assess various aspects of vulnerability in SDN architectures. Each scenario utilised source address randomisation, implemented via the hping3 [24] network testing tool and its --rand-source option, so that every malicious packet is attributed a unique source address. This technique effectively circumvents basic source-based filtering mechanisms and compels the SDN controller to process each incoming packet individually. The SYN flood attack exploits the TCP handshake protocol by overwhelming the victim with a large number of SYN packets and failing to send ACK packets. This leaves many half-open (half-closed) TCP connections that remain unresolved, eventually saturating the victim's network connection resources and complicating the flow of legitimate data.

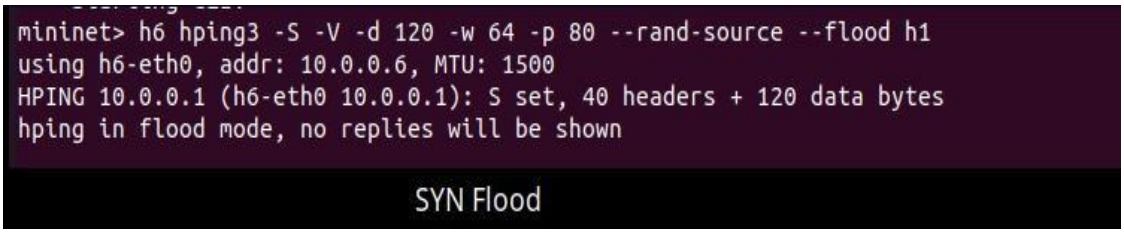

Fig. 3 SYN flood

A UDP flood attack is fairly simple; it creates a high volume of random UDP packets aimed at the network to consume network resources.

The traffic consequently overfills the capacity of the available bandwidth, causing denial of service for legitimate users.

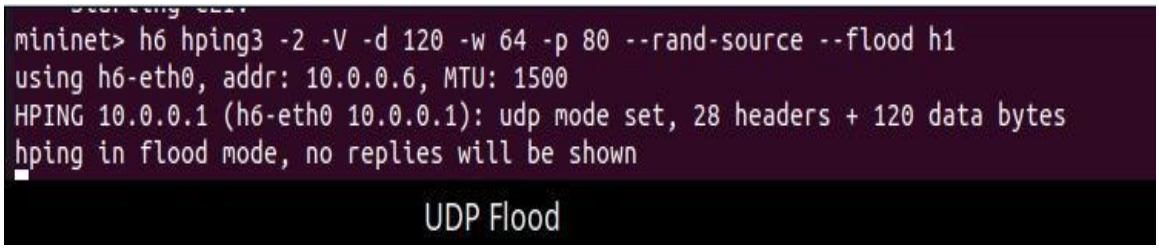

Fig. 4 UDP flood

5.2.1. Controller Saturation Attack

This attack focuses on the core operation of SDN, which overloads the controller with a large number of packet-in messages. Malicious hosts target flow tables systematically by sending packets with different and random header attributes, such as source and destination IP addresses or source and destination ports. These packets are sent to the controller for decision-making in the OpenFlow switch upon receipt.

5.2.2. Bandwidth Saturation Attack

This type of DDoS attack is characterised by attackers sending packets as large as possible and at the highest possible rate to rapidly fill the available bandwidth between attacking hosts and attacked nodes within the SDN network.

5.2.3. ICMP Flood Attack

This attack floods the target hosts with a large volume of ICMP Echo Request messages. The ICMP flood attack allocates processing resources and memory to the victim to process incomplete protocol interactions, similar to SYN flood attacks on TCP/IP networks, because the attackers do not return the proper Echo Reply packets. The backlog of these requests can significantly affect the system performance and normal network operations over time.

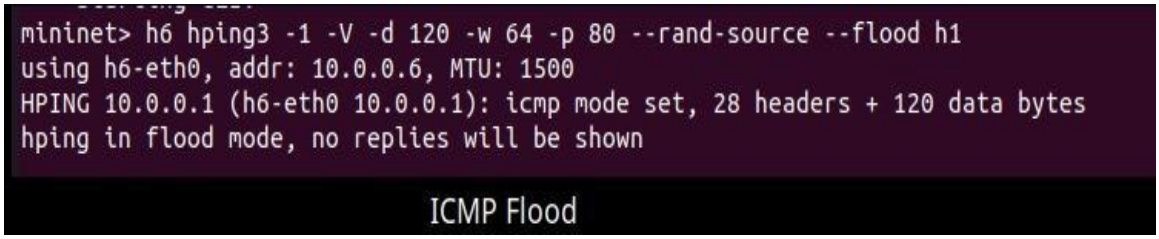

Fig. 5 ICMP flood

The more requests that come in, the more the victim cannot keep up and deny services to authorised users. Moreover, if there are no ICMP handling rules in the flow tables, ICMP traffic can overload the network links, causing packet-in messages to be sent to the controller.

6. Machine Learning Techniques for DDoS Detection

This section presents a brief description of the machine learning algorithms employed for detection and mitigation of data-plane attacks. The selected methodologies cover various algorithmic paradigms such as distance-based models, probabilistic classifiers, kernel-driven approaches and ensemble learning techniques.

6.1. Classification Algorithms

6.1.1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors is an instance-based learning algorithm that classifies data points with a label based on how closely they resemble labelled sample points in a feature space. The basic idea is that similar cases will be located at the same place, and KNN is especially useful in pattern-recognition problems where the boundary of classes can be chosen as a local similarity measure.

The classification process begins by calculating the distance between the target instance and each example in the training set. Provided that the target feature vector is referred to as x (x 1,x 2), then the Euclidean distance between x and a training sample, x i, is as follows:

$$d(x, x_i) = \sqrt{[\Sigma_{j=1}^{n} (x_j - x_{ij})^2]}$$

Once all pairwise distances have been computed, the algorithm selects the k smallest training instances that are closest to the target. The local neighbourhood is constituted by these k nearest neighbors, and a majority voting of their class labels determines the prediction; when the majority are in the attack class, the target is classified as malicious, but when the majority are benign traffic, the target is considered benign traffic. The hyperparameter, k, has a decisive impact on the classification behaviour. When the value of k is small (e.g., when k=1 or k=3), the boundary between decisions closely follows the distribution of training data, and thus, the probability of overfitting and sensitivity to noise is high. The larger the value of k, the smoother the decision surfaces, and the more resistant they are to outliers, but they can also be too large to make the lines between classes indistinct and to decrease the overall accuracy. The choice of the optimal value of k is typically based on cross-validation methods, which compromise these conflicting trade-offs. KNN has several strengths in the context of DDoS detection in SDN setups: it does not require a distinct training phase, it intuitively scales with the changing pattern of threats, and it returns interpretable classifications based on similarity to known examples. However, the computational complexity of the algorithm is directly proportional to the size of the training sample, because to compute the distance, distance computations must be performed between all pairs. This computational cost can be a bottleneck in detecting flows in real time in high-throughput network environments, where many flows are created, run, or processed each second. Moreover, in high-dimensional feature spaces, there is a curse of dimensionality, which causes decreased performance of algorithms, and thus requires consideration in feature selection or dimensionality-reduction preprocessing.

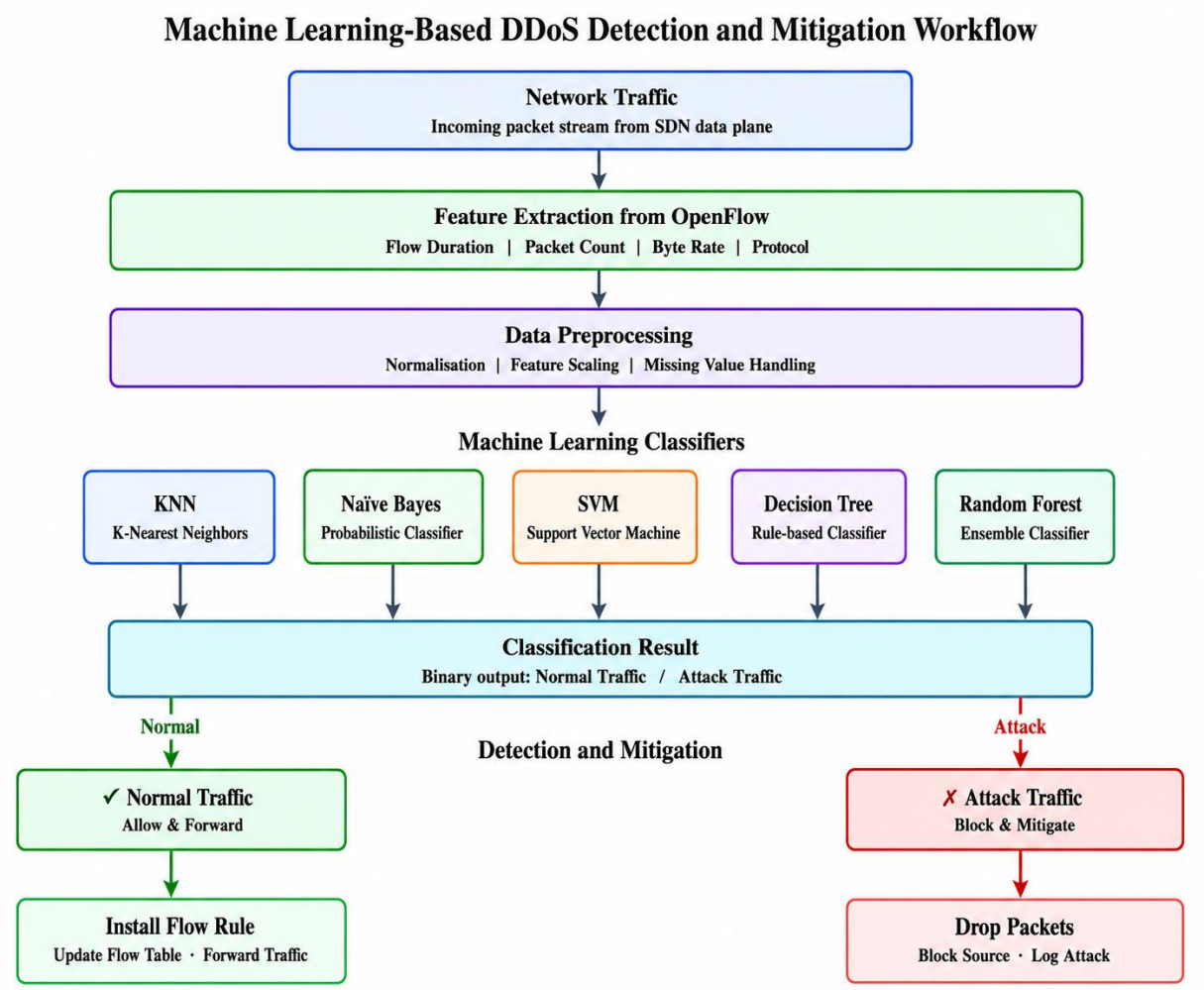

Fig. 6 Machine Learning-Based DDoS detection and mitigation workflow

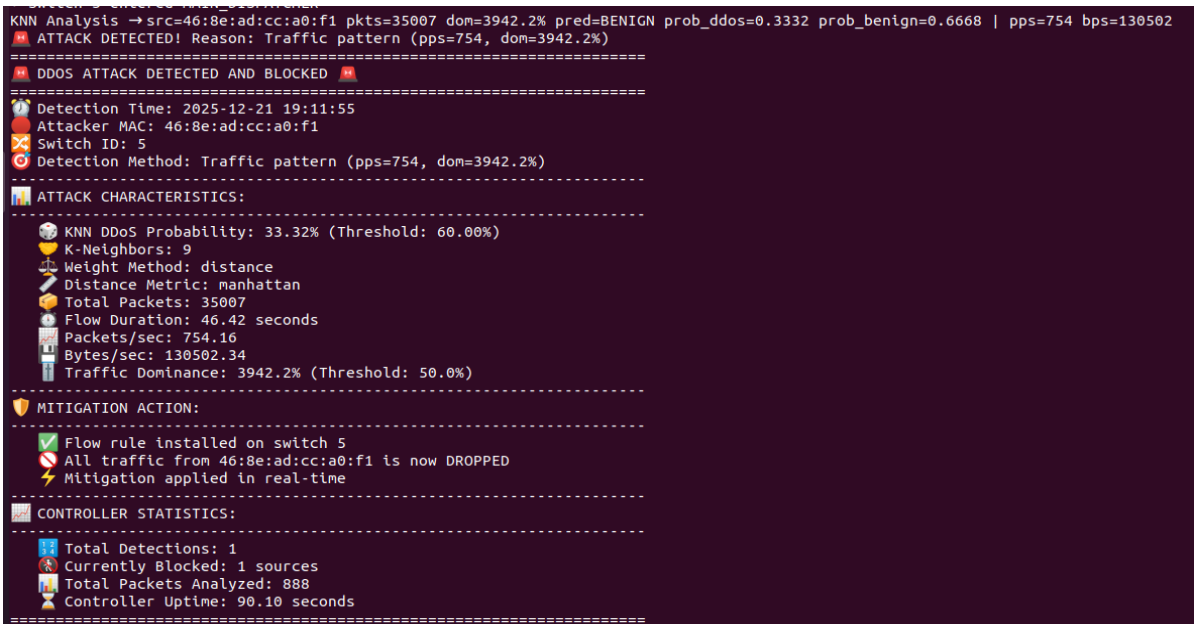

Fig. 7 Attack analysis using KNN ML technique

6.1.2. Naive Bayes

Naïve Bayes is a probabilistic classification model based on Bayes' theorem, which outlines the interaction between conditional and marginal events. The algorithm determines the posterior probability of each set of hypotheses under the observed feature vector and classifies the instance as having the highest posterior probability. The term naive is derived from the conditional independence assumption among the features; each feature is assumed to contribute independently to the class probability.

For a feature vector $x = (x_1, x_2, ..., x_n)$ and class c, Bayes' theorem states that,

$$P(c \mid x) = [P(x \mid c) \times P(c)] / P(x)$$

and because the probability of a fixed instance is the same across classes, the decision rule can be simplified to the maximisation of the product of the probability of an instance and the probability of class, that is, P(x j c)P(c). Under the conditional independence assumption, the likelihood is split into the joint likelihood as follows:

$$P(x \mid c) = \prod_{i=1}^{n} P(x_i \mid c)$$

In the case of continuous attributes, such as packet counts, bit rates, or flow times, which are common in network traffic analysis, the naive Bayes model assumes that each attribute is a Gaussian distribution conditional on the class. The probability density function associated with this is

$$P(x_i \mid c) = (1 / \sqrt{(2\pi\sigma^2)}) \times \exp[-(x_i - \mu)^2 / (2\sigma^2)]$$

Where μ and σ^2 represent the mean and variance of attribute I in class c, respectively, and are estimated based on the training data. The probability distribution of the preceding probability, P(c), is normally obtained as the frequency of class c in the training set.

In the classification phase, the algorithm computes the product of the form $P(c) \times \prod_i P(x_i \mid c)$ of each candidate class and selects the class with the largest value. In practice, log probabilities are used to avoid underflow problems by summing several small-probability terms.

Naïve Bayes is computationally efficient and can be highly effective even in the presence of violations of the conditional independence assumption, as long as the inter-feature dependencies do not essentially alter the decision limits.

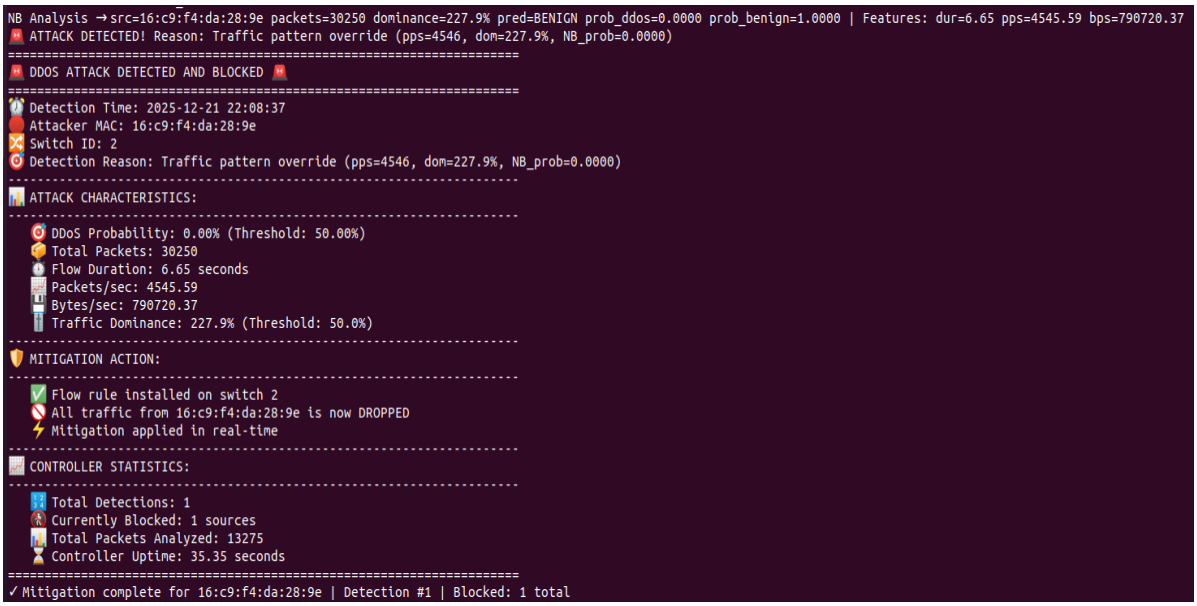

Fig. 8 Attack analysis using Naïve Bayes

The algorithm has linear scaling properties with respect to both the feature dimension and size of the training set, making it applicable to high-throughput real-time detection systems. However, its performance could worsen when features exhibit high levels of correlation because the assumption of independence would not reflect such relationships. The number of packets and bytes are two attributes that are correlated in nature in the case of SDN-based DDoS detection and may limit the accuracy of Naïve Bayes compared to algorithmic approaches with explicit feature interactions.

6.1.3. Support Vector Machine (SVM)

Support Vector Machine is a learning algorithm based on the kernel and is used to build optimal separating hyperplanes in high-dimensional feature spaces. The main goal is to define

a boundary of the decision that can maximise the margin, which is the difference between the decision boundary and the closest training examples of both classes. The principle of maximum margin improves the performance of generalisation by creating a wide gap between the classes. In datasets that can be separated linearly, SVM approximates a hyperplane of the form w top $w \cdot x + b = 0$, which maximises the distance to the nearest training data points. These are called support vectors, and they play a crucial role in the location of the hyperplane; all other instances of training do not influence the decision line. The formulated optimisation problem is as follows:

Minimise: $(1/2)\|w\|^2$ subject to: $y_i(w \cdot x_i + b) \geq 1$ for all training instances i, where y_i represents the class labels.

This constrained maximisation guarantees proper classification of all training cases and reduces the squares of w, which can also give a maximum margin of $2/\|w\|$.

To fit the misclassified training cases (soft margin), a regularisation parameter C is traded off between margin maximisation and training error minimisation. Small values of C yield longer margins at the cost of increased training misclassifications, which may be better for generalisation. Large values of C make the training data stricter and can lead to overfitting. The best choice for C is generally made using cross-validation.

For SDN DDoS detection, SVM shows strong results in detecting attacks and benign traffic when the maximum-margin principle is used because it provides resilience to both measurement noise and feature variability.

Nonetheless, the complexity of SVM training is between $O(n^2)$ and $O(n^3)$, with a training set size of n, potentially constraining its applicability to large datasets. In addition, hyperparameter optimisation, which includes the selection of kernels, C value, and kernel parameters, requires considerable computing power. Nevertheless, SVM often performs well on security-related tasks with high classification accuracy when the geometries of the dividing classes are complex.

6.1.4. Decision Tree

A decision tree is a recursive partitioning algorithm that builds decision rules in a tree structure using information-theory measures, recursively partitioning the feature space. The resulting tree has internal nodes as feature tests, branches as the result of the test, and terminal nodes as class labels. The architecture is intuitive to read since the classification logic can be expressed as a series of if-then statements. The tree is constructed using a greedy top-down approach that selects the feature that maximises the information gain at each node. The information gain measures the decrease in entropy, that is, uncertainty, obtained by dividing the data in proportion to a given property. Given a dataset, S, with a distribution of classes, $S = \{c_1, c_2, ..., c_k\}$, entropy is defined as:

$$H(S) = -\Sigma_{i=1}^{k} p_i \log_2(p_i)$$

Where p_i is the proportion of instances that belong to class c_i. Given that S is partitioned into subsets $\{S_1, S_2, ..., S_m\}$ on feature A, the information gain is

$$IG(S, A) = H(S) - \Sigma_{j=1}^{m} (|S_j| / |S|) H(S_j)$$

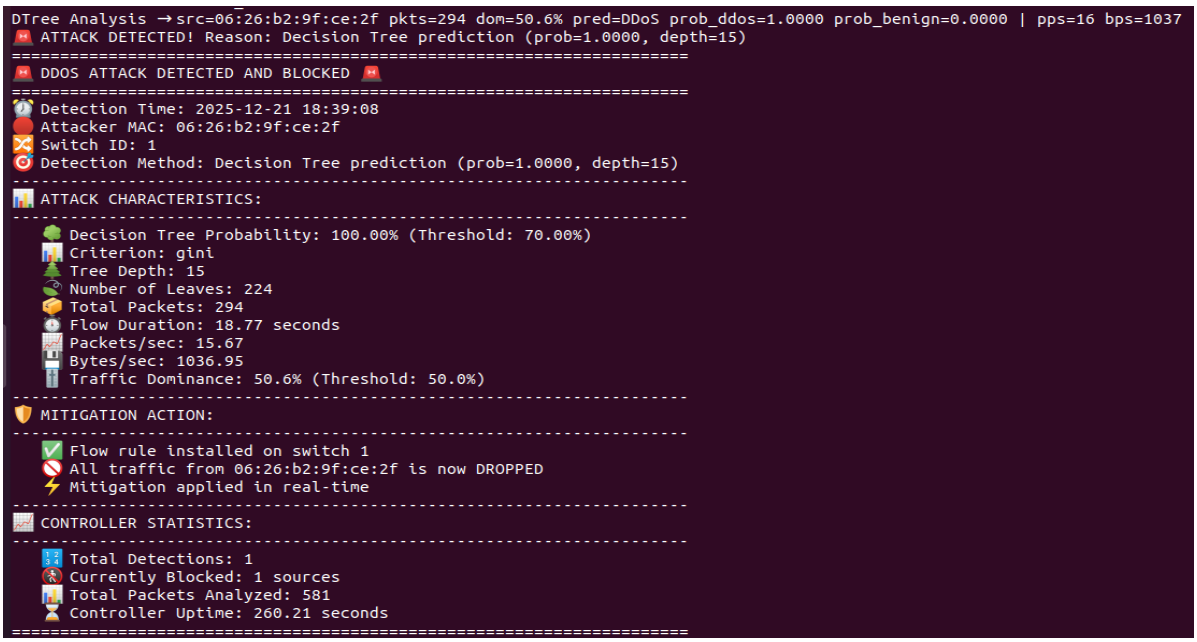

Fig. 9 Attack analysis using DTree ML techniques

Decision trees can be used to automatically find salient feature thresholds in the context of SDN DDoS detection. For example, the model can provide the following rules: when the flow duration is less than 0.5 s and the packet numbers exceed 1000, the flow is an attack. Such explicit rules allow network security analysts to gain an understanding of the logic of detection and confirm that the patterns learned by the tree reflect the legitimate attributes of attacks.

6.1.5. Random Forest

Random Forest is an ensemble learning algorithm that builds several decision trees during training and combines their predictions by majority voting on classification tasks. This ensemble technique reduces the high variance of single decision trees, resulting in models that are more robust and generalisable.

This algorithm provides two standards of randomisation: bagging (randomly sampling training data) and random selection of a subset of features at every node of the tree.

This training process produces a set of decision trees that are independent of each other (i.e. the number of trees is M, a hyperparameter that is normally set to an interval of 50–500 trees). In each tree, the algorithm starts with a bootstrap sample, which is a random sample of size n, by drawing training instances with replacements from the original dataset of size n.

The sampling process led to approximately 63.2% of the incidences being observed in a bootstrap sample, with the remaining 36.8% (out-of-bag samples) being accessible to validation.

In tree construction, random sampling is performed at each node, and a subset of features (usually d features, which is the dimensionality of all features in total) is considered eligible for splitting. This randomised feature selection randomises the trees such that they do not have any dominant features in all trees. This subset of features is then split, with the best split being selected using information gain or Gini impurity metrics.

The classification of new instances was performed by running the instance against all M trees and aggregating the individual predictions of each tree via majority voting.

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), ..., h_m(x)\}$$

Where $h_i(x)$ is the prediction of tree number i; this aggregation lowers the prediction variance compared to that of individual trees; however, the bias remains low. The out-of-bag error provides an independent measure of the capabilities of the generalisation performance without the need for a separate validation set: in each case, only the trees in which that case did not feature appear in the bootstrap sample and contribute to the classification of that case. Random Forest assumes the benefits of decision trees: it is computationally efficient, feature scaling is not required, and feature importance measurements allow its interpretation, although it is also more accurate and robust.

The importance of features can be calculated by finding the average reduction in impurity (or increase in out-of-bag error) of randomising a given feature across all trees. This makes it possible to identify the most discriminative features in the detection of a DDoS.

For SDN security applications, Random Forest is an excellent ensemble, giving it good performance in various attack scenarios. The algorithm is immune to noisy data and outliers, can deal with missing values in feature values, and scales large datasets by building trees in parallel. The number of trees increases the training time linearly, and the prediction is fast (milliseconds per instance). The biggest drawback is that it is not as easy to interpret as a single decision tree because it is difficult to interpret hundreds of trees.

7. Data Collection, Preprocessing, and Model Training

7.1. Dataset Description and Labelling

The dataset utilised for this experiment is CICDDoS2019, which contains network traffic traces of both normal traffic and different DDoS attacks. It is based on a realistic setup of networks and known attack tools; thus, it is a realistic example of the present DDoS techniques. In this study, two types of attack were selected: UDP flood (UDP.csv) and SYN flood (SYN.csv). These are typical volumetric and protocol-attack methods.

Benign traffic samples were collected from Friday-working-hours-mornings. pcap_iscx.csv dataset file, which records typical network activities during business hours. These samples included web browsing, email, file transfers, and other generic traffic. A binary labelling system was used: label 0 for benign traffic and label 1 for attack traffic.

This approach enables focus on detection performance without the added complexity of multiclass classification. All network flow records from the SDN experimental platform were labelled based on their context at the time: standard activity was given label 0, and traffic from attack scenarios was given label 1.

7.2. Feature Engineering

OpenFlow protocol-based network flow statistics are the basis for machine learning-based attack detection. These features were chosen because of their discriminative ability to differentiate between attacks and benign flows.

Table 2. OpenFlow feature descriptions for DDoS detection

Feature	Description
Flow Duration	The maximum time at which the flow was active in the network was calculated. DDoS attacks have abnormally short times because connection attempts are made quickly or for long periods in the case of persistent flooding.
Total Forward Packets	The number of packets transmitted between the source and destination was counted. Attack traffic is likely to generate a high number of forward packets, and an attacker will flood victim hosts with attack packets.
Total Backward Packets	In DDoS attacks, this number is typically very low (or highly asymmetric) because the victim hosts cannot handle the volume of incoming requests. Packets sent back to the source are counted as Total Backward Packets.
Flow Bytes per Second	Flow Bytes per Second The speed at which data are sent in terms of the number of bytes per second is monitored; volumetric DDoS attacks have typical peaks or constant high rates of bytes sent, which are significantly above normal traffic rates.
Additional Statistical Features	Other discriminative statistical features of packets, such as the mean packet size, inter-arrival time statistics, protocol types, and TCP flag distributions, were also added to the classification algorithms.

7.3. Data Preprocessing

The overall preprocessing involved class balancing, feature scaling, and partitioning of datasets, which are described below.

7.3.1. Class Imbalance Handling

The original datasets contain more attack samples than benign samples. This imbalanced distribution may lead to a skewed model in favour of the majority class, which may exaggerate the overall accuracy but reduce the minority detection performance. To counterbalance this influence, the number of benign samples was boosted using random oversampling until the classes were equally represented. The last dataset consisted of 200,000 samples, including 100,000 benign and 100,000 instances of attacks. This equal distribution of classes allows the model to train both classes without bias during the training process.

7.3.2. Feature Scaling and Normalisation:

The features based on network traffic have a significant amount of different scale variation; for example, the number of packets may have very small values as well as thousands, whereas flow duration may have microsecond- to minutes-long values. 1. Distance-based methods, such as KNN, are particularly sensitive to these differences between the feature magnitudes, as are kernel-based methods, such as Support Vector Machines (SVM). To eliminate this, each numerical aspect was standardised to have a mean of 0 and a variance of 1, and the transformation was as follows:

$$x' = (x - \mu) / \sigma$$

Where μ and σ are the mean and standard deviation of the training data, respectively. This normalisation guarantees that all features have an equal contribution to the distance calculations and optimisation, consequently leading to better convergence and algorithm robustness.

7.3.3. Dataset Partitioning

A random sample was selected to divide the dataset into a training set (75%) and test set (25%). The proportion of classes remained balanced in both cases. A set of 150,000 samples was used for model development, and a set of 50,000 samples was used to undertake an objective generalisation assessment. Stratified sampling was performed to provide the same representation for both the attack and benign classes in both subsets, thus minimising the bias created by the imbalance in classes.

7.4. Model Training and Evaluation Methodology

The training of each machine-learning algorithm was performed using the preprocessed training data with the relevant procedure protocols. The decision tree classifier uses an entropy-based splitting criterion supplemented by pruning mechanisms to reduce overfitting. The random forest ensemble consisted of 100 decision trees produced through bootstrap aggregation, and a random subset of features was chosen for each split. The support vector machine used a Radial Basis Function (RBF) kernel, and the hyperparameters were optimised using a systematic grid search. The K-nearest neighbors algorithm used Euclidean distance, where the number of neighbors (k=5) was obtained by cross-validation.

The naive Bayes learner made an assumption of the Gaussian distributions of values of the features depending on each class. The classifiers were then tested on a separate hold-out test set after training to determine their generalisation skills. A set of metrics that most effectively summed up the accuracy of the detection, false-alarm rate, and ability to recall instances of attacks was used to conduct a performance assessment. These measures can be used to provide complementary perspectives because the improvement of one measure can give rise to the worsening of others. As an illustration, a degenerate classifier that identifies all instances as attacks would achieve perfect recall and zero precision.

7.5. Evaluation Metrics

The classification models were evaluated based on their effectiveness using a confusion matrix that divided the predictions into four exclusive categories. True Negatives (TN) refer to correctly determined benign flows, and False Positives (FP) refer to benign flows that are incorrectly determined to be attacks. False Negatives (FN) are attacks that are classified as benign, and True Positives (TP) are those attacks that are detected correctly. The derived performance indicators were calculated using the following foundational counts:

Table 3. Confusion matrix structure for binary classification

	Predicted Normal	**Predicted Attack**
Actual Normal	True Negative (TN)	False Positive (FP)
Actual Attack	False Negative (FN)	True Positive (TP)

Table 4. Performance metrics for DDoS detection evaluation

Metric	**Formula**	**Interpretation**
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall proportion of correct predictions across all instances
Precision	TP / (TP + FP)	Proportion of predicted attacks that are genuine attacks; inversely related to false alarm rate
Recall (Sensitivity)	TP / (TP + FN)	Proportion of actual attacks correctly detected; critical for security applications where missed attacks have severe consequences
F1-Score	2 × (Precision × Recall) / (Precision + Recall)	Harmonic mean of precision and recall; provides a balanced assessment when both false positives and false negatives are important

8. Experimental Results and Discussion

In this section, the experimental results related to the preprocessed CICDDoS2019 dataset, which was used to evaluate the five machine learning algorithms, are presented. The algorithms were evaluated using a variety of metrics to measure the performance of each algorithm and its applicability to real-time DDoS detection in SDN environments.

8.1. Quantitative Performance Analysis

Table 5. Machine learning model performance comparison

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	0.9980	0.9981	0.9979	0.9980
Random Forest	0.9981	0.9982	0.9980	0.9981
SVM	0.9436	0.9396	0.9481	0.9438
KNN	0.9979	0.9985	0.9972	0.9979
Naive Bayes	0.8437	0.8272	0.8690	0.8476

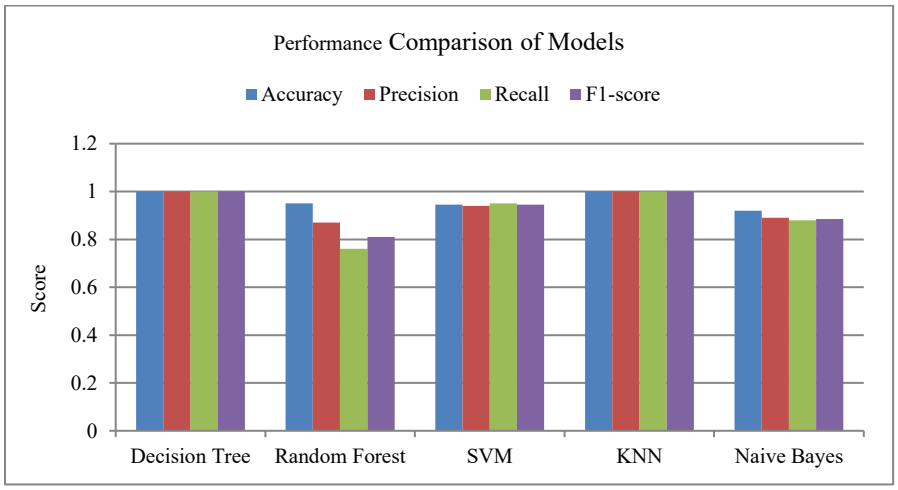

Fig. 10 Model Comparison - Accuracy, precision, recall

8.2. Comparative Analysis and Discussion

In the experiment, it was found that decision trees and K-nearest neighbors are better with respect to all measures taken in the assessment, with an accuracy of over 99.7%. The decision tree showed the best F1-score (0.9980), which indicates an optimum balance between precision and recall.

This performance is based on the capacity of the algorithm to find hierarchical decision rules that are effective in partitioning attacks and benign traffic using discriminative feature thresholds. In addition, decision trees have minimal computational costs at the training and prediction stages, which makes them particularly appropriate for real-time deployment in resource-deprived SDN controller systems.

K-NN achieved a similar level of performance (accuracy = 0.9979, F1-score = 0.9979), with an AUC of 0.9991 and an operating point of FPR = 0.0015, TPR = 0.9984. The confusion matrix confirmed 29,956 true negatives, 44 false positives, 49 false negatives, and 29,951 true positives out of the 60,000 test samples. Very high precision (0.9985) leads to very low false-positive rates.

This feature is beneficial in operational conditions where a false alarm may affect legitimate traffic and congest the security workforce. However, the computational complexity of KNN is linear with the size of the training set, which may be limiting in large network systems where thousands of flows are processed per second. This limitation can be countered by optimisation, such as KD-tree data structures or approximate nearest-neighbor algorithms.

The machine learning SVM provided strong performance (accuracy = 0.9436, F1-score = 0.9438) with balanced accuracy and recall. The RBF kernel allows the SVM to capture complex nonlinear decision boundaries, which offers resistance to adversarial traffic patterns. However, hyperparameter optimisation (C value, kernel parameter γ) is expensive to compute, and domain knowledge is required. The complexity of the training model might be too high to be used regularly in retraining on changing patterns of attacks.

Random Forest achieved strong performance across all evaluation metrics (accuracy = 0.9981, precision = 0.9982, recall = 0.9980, F1-score = 0.9981), with an AUC of 0.9999 and an operating point of FPR = 0.0018, TPR = 0.9980. The confusion matrix confirmed 29,946 true negatives, 54 false positives, 60 false negatives, and 29,940 true positives out of the 60,000 test samples.

The outcomes show that under a controlled SDN lab environment, Random Forest, trained with a balanced and representative dataset, has almost 100% detection performance, similar to that of Decision Tree and KNN. The ensemble-like model makes this approach more robust to noisy traffic features, but also causes this model to be more costly to train, being proportional to the number of trees.

Among the five classifiers, Naive Bayes had the lowest accuracy, precision, recall, and F1 score (0.8437, 0.8272, 0.8690, and 0.8476, respectively), lowest AUC (0.9000), and a significantly higher false positive rate (0.1816). The confusion matrix shows that the classifier has 24,553 true negatives, 5,447 false positives, 3,930 false negatives, and 26,070 true positives.

This implies that a large percentage of attacks and benign flows were misclassified. When network traffic features are correlated, the conditional independence assumption is violated. For example, the number of packets and bytes are highly correlated, reducing the accuracy of the model. Despite this, Naive Bayes is a fast and efficient algorithm and is thus capable of acting as a simple first-stage filter in multistage detector systems.

All algorithms have a consistency of precision better than 0.87, which implies that the false-positive rate is also at an acceptable level. The accuracy of the results is essential to reduce the amount of operational load related to investigating false alarms.

The recall values have a larger difference (0.8690 to 0.9985), which represents different sensitivities to attack traffic. Security applications usually give high recall importance to achieving low missed attacks, regardless of the minor sacrifice in precision.

The experimental outcomes revealed that machine learning-based detection can be effective in attaining an accuracy level that is suitable for production. The decision Tree and KNN become the best options because they have better accuracy, faster computational capability, and balanced precision-recall properties. The algorithms of SDN controllers enable the classification of flows in real time, as well as taking mitigation measures before attacks seriously degrade network performance.

8.3. Comparison with State-of-the-Art

A systematic comparison of the ML classifiers analysed in this study is presented in Table 6, with selected state-of-the-art classifiers reported in the literature. Benchmarking was performed based on the reported accuracy, dataset used, SDN environment, and type of classifier used.

If the metric values were not provided in the cited work, they were marked as such. It is important to recognise that direct comparison of numbers between studies is limited by the differences between the datasets, testbed setup, type of attack, and conditions of evaluation. However, the comparison provides some background indication of the competitiveness of the results of this study.

Table 6. Comparison with State-of-the-Art ML-Based DDoS detection in SDN

Study	Classifier	Dataset	Accuracy (%)	Limitation Noted
Yang and Zhao [15]	SVM	Campus SDN	99.80	Campus only; not generalisable
Sharma and Babbar [19]	RF + Logistic Regression	Edge-IIoT (IoT-SDN)	99.10	IoT-specific; limited generalisability
Kavitha et al., [21]	Decision Tree	KDD Cup 1999	>99.00	Outdated dataset; no real-time testbed
Tahirou et al., [17]	SVM	CICDDoS2019	Not reported	Control-plane focus only
This Study – Decision Tree	Decision Tree	CICDDoS2019 (Real-time SDN Lab)	99.80	Full data/control plane separation
This Study – KNN	KNN	CICDDoS2019 (Real-time SDN Lab)	99.79	Full data/control plane separation
This Study – Random Forest	Random Forest	CICDDoS2019 (Real-time SDN Lab)	99.81	Full data/control plane separation

The results show that the classifiers tested in this study perform at least as well as several recently published classifiers. Most importantly, Yang and Zhao [15] reported a 99.8% detection rate using SVM in a campus SDN environment, whereas the 99.80% and 99.81% rates achieved in this study were in a fully operational testbed with separate data and control planes, real-time traffic capture, and three attack types. Kavitha et al., [21] achieved over 99% accuracy by applying Decision Tree to the KDD Cup 1999 dataset, whereas Sharma and Babbar [19] achieved 99.10% accuracy with a hybrid RF and logistic regression model on the dataset of IoT-integrated SDN. Both these studies were outdone or tied with the current study and with a more representative and modern dataset (CICDDoS2019), as well as a more realistic experimental environment. The most important difference between this study and others is that the classification performance was measured in the actual environment of the SDN network in real-time data capture from the live SDN network with full plane separation, which increases the practical validity of the results reported in this study.

8.4. Mitigation Verification

After applying mitigation measures, connectivity measurements using ICMP (echo request) were conducted between malicious and legitimate machines in the network to assess the effectiveness of the detection mitigation strategy against attacks. Upon detecting traffic with features typical of DDoS attacks, for example, packet rates exceeding configured limits or malicious classification by means of machine learning, the SDN controller programs flow rules on OpenFlow switches to discard packets sent by the identified attack sources.

The post-mitigation connectivity tests confirm that attack hosts are disconnected and cannot reach the victim host, or the victim hosts cannot reach the attack sources. It blocks attack sources and protects network resources, which may help keep the network uninterrupted and protected from attackers. Network connectivity is not lost, and no disruption in service is observed, demonstrating that the mitigation measures are selective to malicious flows and do not affect benign traffic.

The mitigation mechanism is automated and does not require manual action by network administrators. When an attack is detected, the controller automatically derives the correct flow rules, pushes them to the relevant OpenFlow switches via the southbound API, and verifies installation success. Such an automated response enables quick mitigation, which usually occurs within several seconds of an attack, significantly reducing the vulnerability window compared with manual response procedures.

9. ML Deployment Challenges in SDN Data Plane

Although classification performance is an important factor to consider, the deployment of ML-based DDoS detection mechanisms into the SDN data plane introduces a set of practical issues that are beyond the scope of classification. The main deployment issues observed in the experimental part of this study and their implications for real-life SDN deployment are discussed in this section, including scalability, detection latency, and model retraining.

9.1. Scalability

This study was conducted on an experimental testbed that was fully built and tested using limited laboratory resources, including six OpenFlow switches, 18 host nodes, and a single Ryu controller. All features and detection mechanisms were fully tested in this configuration, and the framework of detection and mitigation using ML demonstrated functional correctness. The topology for the testbed is limited by the resources of the lab but is aligned with the architectural rules of engagement in real-world SDN deployments. A multi-controller architecture supports scalability to larger production environments by deploying extra Ryu or

equivalent controllers and managing them in accordance with network traffic and network size. As described in the SDN architectural specification, east/west-bound inter-controller communication interfaces support state synchronisation and load distribution across multiple geographically distributed controller clusters, allowing for horizontal scaling of the controller without redesigning the architecture. The ML classification modules embedded in the controller can be replicated and are scalable for an expanding network, enabling the detection capacity to grow as the network expands.

9.2. Detection Latency

In the experimental environment of this study, the ML classification module was embedded in the Ryu controller, which fetched flow statistics using the OpenFlow protocol. Because classification was performed as part of the flow-processing pipeline in the controller and not as a separate process that added measurable latency to the flow, no attempt was made to measure the latency as a standalone metric.

However, for typical SDN implementations, detection latency is a key metric because of the latency-sensitive nature of most environments, including telecommunications networks and industrial control systems. For switches with multiple distributed components, feature extraction, model inference, and flow rule installation can add a measurable delay under a high throughput. Highly dedicated latency profiling under different traffic conditions and controller configurations is cited as an important area for evaluation in future work, particularly as the deployment scale expands beyond the laboratory testbed.

9.3. Model Retraining

This study involved classifiers trained on a predetermined labelled dataset that did not include any means of online or incremental learning. When applied to a production SDN environment, DDoS attack patterns change with time, as attackers continue to refine their attack methods to avoid detection.

Therefore, the issue of when to retrain the ML models and what events should trigger retraining is an important operational consideration that is beyond the scope of the current experimental work. This is recognised as a problem that needs attention in the future. These might involve regular periodic retraining on new traffic samples, trigger-based retraining after the classification drift is detected, or incremental learning techniques that update the parameters of the model without having to retrain the entire model. These strategies are elaborated in the conclusion and future directions section of this paper.

10. Conclusion and Future Directions

This study presented a comprehensive study of DDoS attacks in a SDN environment with a controlled experimental setup comprising a Ryu controller, the Mininet network emulator, and OpenFlow switches. Three representative attack scenarios were examined: controller saturation through packet-in message flooding, bandwidth exhaustion through volumetric traffic flooding, and ICMP flood attacks. The experimental results demonstrated that these attacks substantially degraded network performance by increasing flow table occupancy, reducing throughput, and causing service disruption.

To enhance the resilience of SDN against DDoS threats, multiple machine learning algorithms were evaluated for real-time data-plane attack detection and automated mitigation. The highest classification accuracy and the most favorable balance between detection performance and computational efficiency were achieved by decision trees and KNN, both yielding optimal detection accuracy (exceeding 99.7%). SVM exhibited strong performance in terms of balanced accuracy and recall; however, its computational overhead may constrain its applicability in real-time scenarios. Random Forest achieved performance comparable to decision trees and KNN (accuracy = 0.9981, AUC = 0.9999). In contrast, Naive Bayes showed the lowest performance (accuracy = 0.8437, AUC = 0.9000) and a comparatively higher false positive rate, making it less suitable for standalone deployment in security-critical SDN environments.

The findings indicate that machine-learning-based detection systems, together with SDN programmability, are an effective means of preventing DDoS attacks in real time. The centralised control plane of an SDN enables the rapid deployment of mitigation policies across a distributed network infrastructure. Automated Detection and Response (D&R) systems significantly reduce the impact of attacks on network operations.

However, these results are not generalisable because of several limitations. The emulated network topology was carefully controlled in a Mininet environment to ensure repeatable and reproducible conditions, but it does not fully reflect a production SDN deployment. Although the testbed represents a full-fledged SDN system, with physically and logically separated data and control planes, real-time attack traffic injected using hping3, and live traffic captured from the running network, these characteristics distinguish it from evaluations based purely on synthetic or pre-recorded data. In practice, production SDN deployments exhibit more diverse application traffic, more complex, evolving topologies, and more advanced adversarial behaviour that is not fully represented in the CICDDoS2019 dataset. In addition, the current study focuses on volumetric and protocol-based attacks, while application-layer and low-rate DDoS variants remain unexplored. These limitations will be overcome in future studies that will test the suggested detection and mitigation framework in larger SDN testbeds with realistic traffic patterns and network dynamics. The study of multi-controller architecture will involve fault tolerance and

resilience under distributed control conditions. In addition, such complicated temporal attack patterns can be modelled by convolutional neural networks, and recurrent neural networks should be considered. Transfer learning techniques can assist detection models in transferring their application capabilities to other network environments without massive retraining.

It is important to highlight that adversarial ML robustness is a separate future research direction not explored in the current study because the objective of this study is to detect and mitigate DDoS attacks. This includes assessing how well the trained classifiers can withstand traffic that is designed to be more difficult for these classifiers to detect, such as feature manipulation or low-rate mimicry attacks.

There are other approaches as well, such as incremental learning methods, which allow for continuous adaptation of the model as new attack patterns arise without the need for complete retraining cycles. This study was not aimed at analysing controller resources and performance, such as CPU utilisation, memory usage, flow table occupancy, and degradation in throughput when attacked, and it is recommended to investigate this in the future. The need for statistical significance testing and performing ablation studies of individual system components to ensure greater empirical rigor in the evaluation is also noted in future work. Finally, defence-in-depth techniques through integration with other complementary techniques, such as network access control, traffic filtering, and anomaly-based intrusion detection, would further reinforce SDN security.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The research was conducted using the authors' own institutional resources.

Acknowledgements

Not applicable.

References

[1] Open Networking Foundation, SDN Architecture Overview, Open Networking Foundation, 2014. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/02/SDN-architecture-overview-1.0.pdf

[2] RYU, Ryu SDN Framework, RYU, 2017. [Online]. Available: https://ryu-sdn.org/

[3] Mininet, Mininet an Instant Virtual Network on your Laptop, 2022. [Online]. Available: https://mininet.org/

[4] Open Networking Foundation, OpenFlow Switch Specification Version 1.5.1, Open Networking Foundation, 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[5] Yazdan Etedali, and Mohamadreza Noorifard, "CatBoost Classifier for DDoS Detection in SDN using Ryu Controller," *2025 33rd International Conference on Electrical Engineering (ICEE)*, Isfahan, Iran, Islamic Republic of, pp. 1022-1026, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[6] Pooja Chaturvedi, and Deepika Bishnoi, "Detection and Classification Approach of Denial of Service Attack in SDN," *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)*, Greater Noida, India, pp. 551-554, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hasen Alomin, Amir Gargouri, and Mohamed Ali Ghorbel, "Enhancing Network Security in SDN: Detecting Low-Rate DDoS Attacks using Decision Trees," *2024 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, Hammamet, Tunisia, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[8] Shruti Keshari et al., "Enhancing SDN Security: Performance Analysis of POX and RYU Controllers Under DDoS Attacks," *2025 12th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida NCR, India, pp. 1-5, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[9] Guoqing Yang et al., "Real-Time DDoS Attack Detection in SDN based on En-CNN and OptiLGBM," *IEEE Access*, vol. 13, pp. 161453-161471, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Rong Cao, "Research on DDoS Attack and Defence System based on SDN Controller," *2025 5th International Symposium on Computer Technology and Information Science (ISCTIS)*, Xi'an, China, pp. 594-598, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Zakaria Abou El Houda, Abdelhakim Senhaji Hafid, and Lyes Khoukhi, "A Novel Machine Learning Framework for Advanced Attack Detection using SDN," *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Kishansmaran Puranik et al., "A Two-Level DDoS Attack Detection using Entropy and Machine Learning in SDN," *2023 3rd International Conference on Intelligent Technologies (CONIT)*, Hubli, India, pp. 1-7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Admilson de Ribamar Lima Ribeiro, Reneilson Yves Carvalho Santos, and Anderson Clayton Alves Nascimento, "Anomaly Detection Technique for Intrusion Detection in SDN Environment using Continuous Data Stream Machine Learning Algorithms," *2021 IEEE International Systems Conference (SysCon)*, Vancouver, BC, Canada, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Karrar Alhamami, and Salah Albermany, "DDoS Attack Detection using Machine Learning Algorithm in SDN Network," *2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT)*, Al-Muthana, Iraq, pp. 97-102, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lingfeng Yang, and Hui Zhao, "DDoS Attack Identification and Defence using SDN based on Machine Learning Method," *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, Yichang, China, pp. 174-178, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Abimbola O. Sangodoyin et al., "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. [CrossRef] [Google Scholar] [Publisher Link]

[17] Abdoul Karim Tahirou, Karim Konate, and Moussa Moindze Soidridine, "Detection and Mitigation of DDoS Attacks in SDN using Machine Learning," *2023 International Conference on Digital Age and Technological Advances for Sustainable Development (ICDATA)*, Casablanca, Morocco, pp. 52-59, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Parvathi Sanjana Pericherla et al., "Early Detection of DDoS Attacks in SDN using Machine Learning Techniques," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India, pp. 1-7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Anshika Sharma, and Himanshi Babbar, "Enhancing DDoS Attack Detection Deploying Machine Learning in Software Defined Networking Environment," *2024 4th International Conference on Intelligent Technologies (CONIT)*, Bangalore, India, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Srinuvasarao Sanapala et al., "Machine Learning based DDoS Attack Detection in Software Defined Networks (SDN)," *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, Namakkal, India, pp. 1124-1126, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] M. Kavitha et al., "Machine Learning Techniques for Detecting DDoS Attacks in SDN," *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Pudukkottai, India, pp. 634-638, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Lipeng Wan, and Guiqin Yang, "Research on DDoS Attack with Learning Ability Detection in SDN Environment," *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Chongqing, China, pp. 1136-1140, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Mohammed Ibrahim Kareem, and Mahdi Nsaif Jasim, "The Current Trends of DDoS Detection in SDN Environment," *2021 2nd Information Technology to Enhance e-Learning and other Application (IT-ELA)*, Baghdad, Iraq, pp. 29-34, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] Kali, Hping3, Kali, 2025. [Online]. Available: https://www.kali.org/tools/hping3/

[25] S. Padmakala et al., "Circle Search Algorithm with Convolutional Neural Network for Distributed Denial-of-Service Attack Detection in Software-Defined Networks," *2024 International Conference on Distributed Systems, Computer Networks and Cybersecurity (ICDSCNC)*, Bengaluru, India, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ajit Karki, and Raktim Deb, "Comprehensive Security Analysis and Threat Mitigation Strategies for Software-Defined Networking (SDN) Environments," *2025 3rd International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC)*, Silchar, India, pp. 623-628, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[27] Zining Zheng, Qi Zhong, and Uno Fang, "Enhancing Network Security: An SDN-Enabled Defence Mechanism Against LDoS Attacks," *2025 3rd International Conference on Mobile Internet, Cloud Computing and Information Security (MICCIS)*, Dongguan, China, pp. 1-6, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[28] W. Aïda Ouedraogo Rakissaga, P. Justin Kouraogo, and T. Fréderic Ouedraogo, "Preventing DDoS Attacks in SDN Networks: A Model of Defence Against Packet-in Flooding," *2025 IEEE World AI IoT Congress (AIIoT)*, Seattle, WA, USA, pp. 1025-1030, 2025. [CrossRef] [Google Scholar] [Publisher Link]