A Simulation-Based SDN Framework for Cyberattack Detection and Traffic Management

Aboubakr Bajenaid
Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
abajnaid@kau.edu.sa (corresponding author)

Maher Khemakhem
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
makhemakhem@kau.edu.sa

Fathy E. Eassa
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
fathy55@yahoo.com

Farid Bourennani
Department of Computer Science and Artificial Intelligence, Faculty of Computer Science and Engineering, University of Jeddah, Jeddah 23890, Saudi Arabia
fbourennani@uj.edu.sa

Junaid M. Qurashi
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
jqurashi0001@stu.kau.edu.sa

Abdulaziz A. Alsulami
Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
aaalsulami10@kau.edu.sa

Badraddin Alturki
Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
baalturki@kau.edu.sa

Received: 12 September 2025 | Revised: 12 October 2025 and 25 October 2025 | Accepted: 29 October 2025

 DOI: https://doi.org/10.48084/etasr.14738

ABSTRACT

Software-Defined Networking (SDN) enables centralized control and programmability, providing a promising foundation for intelligent traffic management and enhanced security. This paper proposes a simulation-based SDN framework that integrates multiple Machine Learning (ML) models to evaluate the effectiveness of cyberattack detection and to manage packet resilience through the analysis of metrics such as throughput, switch delay, controller response time, and security performance indicators. Experimental results demonstrate that the Decision Tree (DT) and Extreme Gradient Boosting (XGBoost) classifiers achieved higher throughput, whereas the Deep Neural Network (DNN) and Transformer models imposed greater computational overhead. In addition, the DT and Random Forest (RF) models showed an

accuracy–responsiveness trade-off, with better throughput and less processing time. Experimental results demonstrate high classification accuracy, with DT, RF, and XGBoost achieving near-perfect detection rates, whereas DNN and Transformer models maintained competitive performance despite certain class-specific limitations. The F1-scores validate the robustness of these classifiers. Complementary SDN simulations assessed packet management efficiency, yielding a consistent packet drop rate of approximately 31–32%, throughput ranging from 7.55 to 9.79 packets/s, and average switch delays around 12 ms across models. The findings confirm that the suggested hybrid approach not only detects cyberattacks effectively but also maintains acceptable SDN performance under diverse traffic conditions.

Keywords-Software-Defined Networking (SDN); Quality of Service (QoS); traffic management; load balancing; intrusion detection; Machine Learning (ML); classification performance; packet loss; throughput; latency

I. INTRODUCTION

Software-Defined Networking (SDN) is a significant advancement in network architecture that decouples the control plane from the data forwarding plane. This paradigm simplifies network administration and enhances flexibility. In SDN, the control plane is centralized in a controller that maintains a global view of the network, whereas the data forwarding plane comprises networking elements (e.g., switches and routers) that execute the controller's commands. The communication between the controller and networking devices is standardized through the OpenFlow protocol, providing a uniform programmable interface for SDN [1].

However, SDN's reliance on software-defined control introduces new security risks. Since network behavior is determined programmatically, vulnerabilities in the controller, Application Programming Interfaces (APIs), or switches can be exploited by attackers to compromise the system [2]. Historically, cyberattacks on conventional networks have shown that new technologies can introduce new vulnerabilities. In the context of SDN, the controllers and links of the control plane present new security issues exclusive to SDN, in addition to existing attack vectors in conventional networks. Due to its extensive deployment, SDN is also vulnerable to challenges related to data privacy and access control in network routing [3, 4].

Although SDN programmability provides significant benefits, enabling cloud services, traffic monitoring, and security applications, it also introduces several new threats [5]. Even a single vulnerability can lead to significant harm. For example, attackers can gain complete control over the network by compromising a single SDN controller. Therefore, security must be treated as a fundamental design feature of SDN [6]. The two most security susceptible SDN features are flow table restrictions and the centralization of control, both of which can degrade performance and endanger the entire network [7]. Furthermore, various SDN components are vulnerable to attacks. Switches, northbound and southbound APIs, and the application plane can be used to compromise the SDN controller. Man-in-the-middle and packet sniffing attacks can infect the APIs, whereas viruses and Trojan horses may infect the switches, intensifying network damage similarly to attacks on the controller. Unlike traditional networks, where a cyberattack typically affects only a portion of the network, compromised switches or end users in SDN can overwhelm the centralized controller with excessive traffic, leading to the disruption of the entire network [2, 8].

A common SDN vulnerability arises when a switch receives packets that do not match any flow rules. In such cases, the switch forwards the packets to the controller as packet-in messages using the OpenFlow protocol. Attackers can exploit this behavior by sending forged packets to flood the controller. The controller then processes these packets and returns flow instructions as packet-out messages, creating a feedback loop that exhausts resources [9].

To address these challenges, this paper proposes a simulation-based SDN framework that integrates multiple Machine Learning (ML) models for cyberattack detection and traffic management evaluation. By analyzing metrics such as throughput, switch delay, and controller response time, as well as security evaluation metrics, the proposed approach aims to balance security measures with packet-handling efficiency.

The key contributions of this work are:

- It unifies security and performance evaluation by integrating cyberattack detection with packet management analysis.
- It introduces a simulation-based model designed to emulate a realistic SDN environment and evaluate traffic management behavior under controlled experimental conditions.
- It provides a switch-level load distribution approach using a round-robin mechanism to balance traffic among switches, thereby mitigating flow table overload and improving resource utilization.
- It incorporates multiple ML and Deep Learning (DL) classifiers to evaluate their effectiveness in detecting attacks and their impact on overall network performance.
- It validates the feasibility of adopting SDN as both a secure and resilient architecture.

II. LITERATURE REVIEW

SDN has attracted significant interest due to its centralized control and programmability. However, ensuring both robust security and efficient traffic management remains a key challenge. The advantages of SDN have contributed to novel solutions for traditional network issues, including congestion management, load balancing, and Quality of Service (QoS) requirements. Optimizing QoS is essential for latency-sensitive communication, especially in real-time SDN traffic, which motivates the integration of QoS mechanisms in SDN environments [10]. Among the various optimization

techniques, intelligent load balancing is key to addressing congestion and scalability issues.

Data security remains a fundamental pillar in modern network infrastructures. Despite the benefits of SDN programmability, it introduces new security challenges. Managing security policies in an SDN environment can be complex due to the need to coordinate policies across multiple planes and components. This complexity requires consistent synchronization to maintain network integrity, and authors in [11] demonstrated how efficient synchronization mechanisms facilitate rapid policy dissemination. An exhaustive analysis of SDN security vulnerabilities was conducted by authors in [5], who identified potential security holes and suggested countermeasures to guide future research. In 2025, authors in [12] integrated Support Vector Machine (SVM) and Random Forest (RF) classifiers to detect anomalies in SDN traffic. Using the SDN-DDoS and NSL-KDD datasets, they introduced a feature selection method that achieved an accuracy of 99.2%.

Load balancing is crucial in SDN for enhancing availability and scalability, as well as minimizing response time. Distributing traffic across switches prevents congestion, ensures fair resource utilization, and improves overall network performance. Intelligent load balancers also enable predictive analytics, allowing administrators to identify traffic bottlenecks before they emerge [13, 14]. Unlike general traffic distribution methods, our contribution lies in achieving fairness and stability at the switch level. By employing a round-robin scheduling mechanism, the SDN controller distributes incoming flows fairly across available switches. Ensuring secure and efficient data distribution should depend on real-time threat detection. By simulating cyberattacks on SDNs and observing the behavior of data flow under different conditions, this paper aims to establish a framework in which data security becomes a prerequisite for efficient distribution. This approach improves scalability and maintains stable latency.

Several studies have proposed load-balancing mechanisms that rely on flow statistics to reroute flows toward underutilized paths. These mechanisms are considered proactive approaches to prevent congestion. The SDN controller can influence queuing behavior either directly through OpenFlow queuing configurations or indirectly by altering flow paths to avoid congested queues [14, 15]. Research on SDN traffic enhancement and priority queueing has revealed three primary approaches: heuristic, parametric, and model-based. Heuristic techniques use algorithms that discover queue models for traffic and regulate queue traffic using empirical methods and rules of thumb without considering any specific model. Authors in [16] proposed a heuristic approach to balance packet load among queues. Their research analyzed a partially implemented SDN network comprising both SDN and non-SDN devices, demonstrating how the centralized controller can achieve substantial enhancements in network usage while minimizing packet losses and delays. Authors in [17] suggested handling traffic at the cluster head through priority queues for prioritizing and aggregating incoming packets. This technique ensured that QoS standards were met in terms of delay and reliability. Authors in [18] proposed a heuristic approach to traffic engineering in SDNs that utilizes multipath forwarding and flow switching across paths. This strategy dynamically selects the optimal route based on network load. To manage incoming traffic, authors in [19] presented a scheduling method that enqueues packets into appropriate queues according to priority, showing that processing high-priority queues first mitigated packet loss.

Controller performance and load distribution in multi-controller SDNs have been addressed by authors in [20], who introduced a Multiple Domain Partition (MDP) method capable of matching controllers with switches based on load-balancing rates and node attraction ability, thus improving control plane reliability. Authors in [21] suggested a Reliability-Aware Flow Distribution Algorithm (RAFDA) and two complementary optimization methods for distributing flows according to traffic load and end-to-end latency. Compared to conventional queuing policies that prioritize packets within a single switch buffer, numerous flow distribution algorithms have been developed in SDN to enable load balancing across multiple switches. Random selection, hashing-based distribution, and round-robin scheduling are some of the approaches that fall under this category. In this study, a pure round-robin system is employed to allocate incoming flows to switches in a sequential manner. In addition to preserving simplicity and fairness, this strategy ensures that switch resources are used in a balanced manner and prevents congestion hotspots.

Despite the growth of ML research on SDN security, most studies focus either on detection accuracy [22] or on network performance [14, 15], but rarely on both, as summarized in Table I. Many studies analyze classifier accuracy and recall, emphasizing how well attacks are detected without considering their influence on SDN units such as flow table load or controller response time. Some initiatives have examined QoS features like load balancing and queuing but often ignore security. Few contributions holistically integrate sophisticated attack detection and real-time traffic optimization. To bridge this gap, this study proposes a unified framework that incorporates ML and DL classifiers in an SDN context and assesses their security performance and impact on network-level metrics, including throughput, switch latency, and computational overhead.

Load balancing in SDN-based Internet of Things (IoT) networks was surveyed by authors in [23] who compared relevant techniques for QoS improvement. Using the InSDN dataset [24], authors in [25] compared Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and Transformer models in SDN, achieving F1-scores of 99.01% and 99.02%, respectively. The Transformer is a fast DL model that processes input sequences in parallel and transforms them into corresponding vector representations, improving training and inference efficiency. This modern classifier utilizes the attention mechanism to focus on relevant features without relying on recurrent networks. In SDN-based traffic analysis, the Transformer can capture temporal dependencies in flow data more efficiently than traditional recurrent models. Currently, this architecture is widely applied in various fields, including time series prediction, computer vision, and Natural Language Processing (NLP) [26]. Despite using older datasets, authors in [27] employed a hybrid approach to address both

security and network performance with the CIC IDS 2017 dataset, although concerns about its construction remain [28]. Authors in [28] benchmarked 14 modern datasets, including InSDN, evaluating seven ML classifiers based on the macro F1-score. However, their study did not evaluate packet-level performance in SDN simulations, which is the focus of this work.

TABLE I. RECENT SDN-RELATED STUDIES

Ref.	Main focus	Techniques / approach	Metrics measured	Dataset used	Findings
[23]	Network performance	QoS-aware load balancing in SDN-IoT	Latency, throughput, resource utilization	Not available	Not available
[25]	Detection capabilities	CNN–LSTM hybrid & Transformer	Accuracy, precision, recall, F1-score	InSDN	Transformer: 99.02%. CNN–LSTM: 99.01%
[22]	Detection capabilities	ML & DL classifiers (DT, RF, XGBoost, KNN, CNN, GRU, LSTM)	Precision, recall, F1-score	IEC 60870-5-104	F1-score: 93.57%
[28]	Detection capabilities	ML & DL classifiers (NN, CNN, DBN, EFC, LCCDE, EGS, CFC)	Macro F1-score	14 datasets	NN: 62% (InSDN:0.727; UNSW-NB15: 0.405)
[27]	Hybrid (network performance and detection)	ML & DL classifiers (DT, KNN, RF, LSTM, CNN, LSTM)	CPU usage, CPU load, energy consumption, accuracy	CICIDS 2017	LSTM: 93.86%; CNN: 94.74%; EIDM: 99.56%

The integration of SDN programmability with OpenFlow can facilitate the creation of more effective and adaptable load-balancing systems. Network administrators can adjust resource distribution to accommodate fluctuating network traffic and changing requirements. Moreover, SDN architecture offers developers APIs to customize load-balancing algorithms according to specific business needs, hence enhancing network performance and reliability. Prior studies have primarily focused on improving network efficiency and reducing latency through such mechanisms. However, few have jointly considered these aspects together with real-time security detection. This gap highlights the need for hybrid approaches, such as the one proposed in this paper, that address both traffic management and intrusion detection while maintaining scalability and stability. For example, by utilizing real-time data and predefined policies, autonomous reconfiguration of network paths and bandwidth distribution can improve device efficiency and reduce latency.

III. METHODOLOGY

This section outlines the methodology of the proposed simulation-based SDN framework designed for cyberattack detection and traffic management. It begins by describing the overall simulation framework, followed by details of the dataset used for evaluation and an overview of the ML classifiers employed in the study. The implementation of the SDN controller is then explained, and finally, the performance metrics used to assess the framework's effectiveness are presented.

A. Proposed Simulation Framework

The proposed framework simulates the lifecycle of a packet within an SDN environment, integrating both traffic forwarding and cyberattack detection techniques. As shown in Figure 1, the process begins by loading a packet from a pre-labeled dataset row, which a simulated host then sends to the corresponding switch. Upon receiving the packet, the switch invokes an ML classifier to determine whether the packet is associated with a cyberattack.

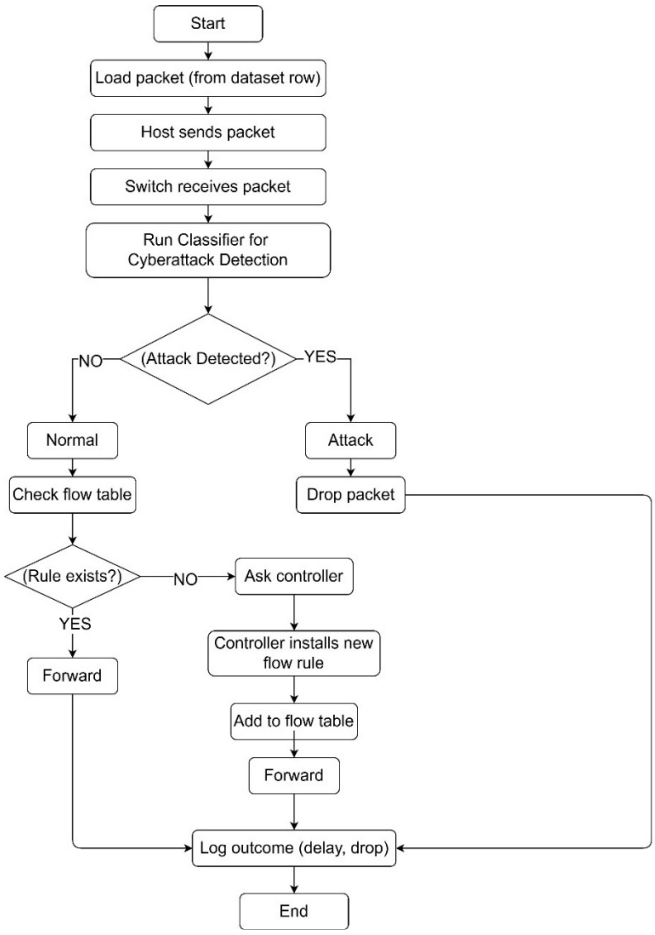

Fig. 1. Proposed packet handling process in SDN with integrated cyberattack detection.

If an attack is detected, the packet is immediately dropped to prevent potential damage or propagation. Conversely, if no threat is identified, the packet is treated as normal and forwarded for further processing. The switch then checks its flow table to determine if a forwarding rule for the destination exists. If a matching rule is found, the packet is forwarded directly according to the existing instructions. However, if the rule is missing, the switch sends a request to the SDN controller.

The controller dynamically installs a new flow rule and sends it back to the switch. This rule is then added to the switch's flow table to optimize future forwarding decisions. The packet is forwarded according to the newly installed rule. Finally, the simulation logs critical metrics such as packet delay and drop events, which are then used for QoS assessment and classifier performance analysis.

Traffic was streamed through the network to evaluate processing under realistic load conditions. In total, over 100,000 packets were processed, with specific switches handling higher traffic loads to reflect varying stress levels across the network. This configuration facilitated precise performance testing under dynamic and imbalanced traffic conditions. The simulation-based SDN environment was designed with 10 switches and multiple hosts. The SDN controller orchestrates all forwarding decisions and implements round-robin-based load balancing to distribute traffic across the network elements.

To train the ML and DL classifiers and evaluate their performance, this work applies several classifiers by splitting the dataset into two subsets: training (70%) and testing (30%).

B. Dataset

The selected public dataset, InSDN [24, 29], was developed to address difficulties associated with traditional datasets by including distinctive attack features specific to SDN environments. Selecting a dataset relevant to SDN is essential for a comprehensive assessment of SDN-oriented attack detection methodologies. Employing datasets not specific to SDN may cause compatibility issues, as the attack vector implementation must consider the network's changing architecture [30, 31]. This publicly available dataset involves both normal traffic and various labeled malicious attacks, including DoS, Distributed Denial of Service (DDoS), Brute Force Attack (BFA), Probe, Botnet, Exploits (User-to-Root, U2R), and Web attacks. Each flow contains 82 features, including Flow-id, Protocol, Duration, Src-IP, Dst-IP, TotLen-Fwd-Pkts, TotLen-Bwd-Pkts, Flow-Byts/s, and the number of packets.

C. Machine Learning Classifiers

ML and DL classifiers are important technologies for pattern recognition and prediction. In this work, we trained five popular models, including Decision Tree (DT), RF, Extreme Gradient Boosting (XGBoost), Transformer, and Deep Neural Network (DNN). The selected classifiers have diverse attributes and are suitable for anomaly detection tasks. DT and RF are interpretable models that effectively process structured data and help uncover key decision patterns. XGBoost is particularly adept at managing imbalanced datasets. Transformer-based classifiers are well-suited for capturing sequential relationships in network traffic, making them effective for identifying advanced attacks. DNNs, as start-of-the-art models, utilize layered feature extraction to recognize intricate patterns in complex data. The combination of these models provides a balance between conventional tree-based techniques and modern DL architectures, enhancing both interpretability and predictive performance.

D. Controller

In our simulation setup, the SDN controller, which acts as the brain of the SDN, is implemented as a Python class and modeled using a pure round-robin scheduling mechanism to distribute incoming flow rules among the available switches. All registered switches are stored in an ordered list, and an index pointer cycles through them sequentially. Each time a switch requires a new flow rule for a unique (Src-IP, Dst-IP) pair, the controller assigns it to the next switch in the list, incrementing the pointer by one and returning to the first switch when reaching the end. This systematic rotation ensures fair utilization of switch resources, prevents any single switch or flow table from becoming overloaded, and maintains balanced traffic distribution. This design simulates realistic SDN behavior and facilitates accurate evaluation of controller delay, switch load, and flow table growth in a balanced network environment.

E. Performance Metrics Statistics

Several key evaluation metrics were computed at the end of the SDN simulation to assess the network's behavior and the controller's performance. These metrics provide insight into network efficiency and reliability, as described below:

- Total packets processed: This metric represents the total number of data packets that traversed the network during the simulation. It reflects the overall load and traffic volume handled by the SDN environment.
- Packet drop rate: This is the percentage of packets that were lost or discarded during transmission. It is computed by dividing the number of dropped packets by the total number of packets and multiplying by 100. A high drop rate can indicate congestion, buffer overflow, or inefficiencies in routing policies.
- Throughput: This metric indicates the average rate at which packets are successfully transmitted across the network during the simulation. It is calculated as follows:

$$\text{Throughput} = \frac{P_{\text{received}}}{T_{\text{total}}} \quad (1)$$

  where P_{received} denotes the total number of successfully received packets, and T_{total} refers to the simulation time in seconds.
- Switch delay: This refers to the time a packet spends within a switch before being forwarded. Its values can identify bottlenecks within switches.
- Average controller time: This metric evaluates the average time taken by the SDN controller to respond to packet-in messages from switches. It reflects the responsiveness of the control plane and the efficiency of decision-making in dynamic conditions.
- Packets per switch: This metric helps understand the traffic distribution across different switches. It indicates how balanced the network load is, which can affect performance.
- Flow table size per switch: This represents the number of flow entries installed in each switch's table. Monitoring

flow table sizes guarantees that switches do not reach their capacity limits, which could degrade performance.

IV. RESULTS AND DISCUSSION

This section briefly presents the experimental findings and illustrates the benchmark results of several ML-based approaches evaluated under comparable conditions.

A. Experimental Setup

The experiments in this study were conducted on a personal computer with the hardware specifications listed in Table II. Regarding software, the Python programming language (version 3.11.9) running on the Jupyter platform was used. Table III lists the main parameters used in the simulation and evaluation process. Furthermore, the configuration parameters of the classifiers used in this study are summarized in Table IV. These values were selected based on widely adopted defaults to ensure a fair evaluation.

TABLE II. HARDWARE SPECIFICATIONS

Component	Specification
GPU	NVIDIA® GeForce RTX™ 4090, 24 GB GDDR6X
CPU	Intel® CoreTM i9-14900K, 3.20 GHz
RAM	32 GB DDR4
Storage	1 TB HDD

TABLE III. SIMULATION AND CONTROLLER PARAMETERS WITH DESCRIPTIONS

Parameter name	Description
dataset_path	Path to the file (cleaned_data2_B.csv)
packet_delays	List storing switch processing delays (ms)
packet_drops	Counter for dropped packets (classified as attacks)
controller_times	List of controller processing times (ms)
switch_packet_counts	Number of packets handled by each switch
switch_flow_table_sizes	Flow table size per switch (tracks the number of flow rules per switch)
feature_cols	List of features used as model input
max_depth=10	Maximum depth of the decision tree
min_samples_split	Minimum number of samples required to split an internal node
test_size=0.3	Proportion of the dataset used for testing
random_state=42	Ensures reproducibility by setting a random seed
round_robin_index	Index used to implement the round-robin switch selection in the controller
time.sleep(…)	Random sleep calls to simulate real-world delays in switches and the controller

B. Performance of Packet Management Simulation

This subsection evaluates the performance of packet management and security mechanisms. Prevalent metrics, including accuracy, precision, recall, and F1-score, were used to assess all classifiers, as defined by the following equations:

$$\text{Test Accuracy} = \left(\frac{TP+TN}{TP+FP+TN+FN}\right) \times 100 \quad (2)$$

$$\text{Precision} = \left(\frac{TP}{TP+FP}\right) \times 100 \quad (3)$$

$$\text{Recall} = \left(\frac{TP}{TP+FN}\right) \times 100 \quad (4)$$

$$\text{F1} - \text{score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right) \times 100 \quad (5)$$

where TP denotes true positives, TN denotes true negatives, FP denotes false positives, and FN denotes false negatives.

TABLE IV. CLASSIFIER CONFIGURATIONS AND PARAMETER SETTINGS FOR SDN TRAFFIC ANALYSIS

Classifier	Key parameters	Description
DT	max_depth = 10, min_samples_split = 10, min_samples_leaf = 5, random_state = 42	Baseline single-tree classifier used for comparison
RF	n_estimators = 100, max_depth = 10, min_samples_split = 10, min_samples_leaf = 5, max_features = 'sqrt', random_state = 42	An ensemble of decision trees designed to enhance model stability and reduce overfitting
XGBoost	n_estimators = 50, max_depth = 6, learning_rate = 0.1, subsample = 0.8, colsample_bytree = 0.8, eval_metric = 'mlogloss'	Gradient boosting model optimized for structured SDN flow data
Transformer	n_layers = 3, hidden_dim = 128, n_heads = 8, feedforward_dim = 512, optimizer = 'Adam', batch_size = 32, epochs = 10, learning_rate = 0.001	Attention-based DL model that captures temporal dependencies in SDN traffic data more efficiently than recurrent models
DNN	input_layer = Dense(128, activation = 'ReLU'); hidden_layer1 = Dense(64, activation = 'ReLU'); dropout1 = 0.3; hidden_layer2 = Dense(64, ReLU) + dropout2 = 0.2; output_layer = Dense(y_train.shape[1], activation = 'softmax'); optimizer = 'Adam'	Fully connected DNN designed for nonlinear pattern recognition and multi-class classification

The simulation environment consisted of ten switches to evaluate the performance of packet management and security mechanisms. For an initial overview, we aggregated packet counts and flow table sizes across all classifiers to investigate the general relationship between switch traffic and controller load. The results indicated a moderately strong positive correlation (0.73). As shown in Tables V and VI, the simulation processed over 100,000 packets, achieving an average throughput of 9.79 pkts/s. The recorded packet drop rate was 31.69%, whereas the average switch delay remained stable around 12 ms, although congestion occurred at specific points. Controller processing time ranged from 67.5 to 68.14 ms, influencing overall SDN latency. Throughput varied across classifiers, with the highest reaching 9.79 pkts/s in the DT configuration and the lowest being 7.00 pkts/s in the Transformer setup. These findings confirm the system's ability to handle a moderate traffic load effectively while maintaining acceptable latency levels across switches and controller communication.

TABLE V. PERFORMANCE METRICS FOR PACKET MANAGEMENT IN SDN

Metric	Value
Total packets processed	100,201
Throughput (pkts/s)	9.79
Avg switch delay (ms)	12.00
Min switch delay (ms)	0.00
Max switch delay (ms)	112.52
Avg controller time (ms)	68.14

TABLE VI. PERFORMANCE EVALUATION OF ML AND DL MODELS IN SDN

Classifier	Accuracy	Weighted average	Packet drop rate (%)	Throughput (pkts/s)
DT	1.00	1.00	31.69	9.79
RF	1.00	1.00	31.70	9.44
XGBoost	1.00	1.00	31.70	9.44
Transformer	0.98	0.98	31.88	7.00
DNN	0.99	0.96	31.81	7.56

C. Security Evaluation

DT, RF, XGBoost, Transformer, and DNN classifiers were implemented to detect attacks within the SDN traffic. The DT model achieved near-perfect results with 100% accuracy, and F1-scores close to 1 for most attack types. In contrast, the DNN classifier exhibited lower precision for certain classes, such as BFA, but maintained high performance in detecting DDoS and Botnet attacks. These results demonstrate that classifiers can be effective for real-time SDN-based attack detection with minimal impact on QoS. The Transformer classifier, however, showed significantly reduced effectiveness against rare attacks, such as BFA and U2R, resulting in a macro F1-score of 65%, despite maintaining an overall accuracy of 98% due to the dominance of normal traffic.

From the SDN simulation perspective, all classifiers processed the same number of packets, which was 100,201. However, flow table sizes and the distribution of packets varied across switches. The DT and XGBoost classifiers demonstrated better throughput (above 9 pkts/sec), whereas DNN and Transformer models showed reduced throughput, which can be attributed to higher classification complexity and increased controller involvement. Moreover, statistics at the switch level indicated that some classifiers imposed heavier loads on specific switches, as evidenced by the larger flow table sizes and uneven packet counts, which may impact scalability in real-world SDN deployments. Although ML and DL classifiers demonstrated higher detection accuracy, they required significantly more processing time, potentially limiting real-time deployment.

The integration of performance metrics and attack detection results reveals that intelligent load distribution and scheduling have a direct impact on both QoS and security detection efficacy. Classifiers with higher throughput and lower delay (e.g., DT and XGBoost) yielded better attack classification precision and more balanced load handling. Meanwhile, the Transformer classifier struggled in the SDN simulation, showing the importance of latency-aware classifiers. These insights highlight the effectiveness of using simulation-based SDN testbeds for QoS evaluation.

The findings were compared with prior research that used conventional datasets like NSL-KDD and CICIDS 2017 for SDN attack detection. In contrast to prior research that primarily emphasized classification accuracy, the suggested system integrates security detection with scalability. The XGBoost and DNN classifiers in our configuration achieved superior detection accuracy (99.2%) while ensuring consistent throughput and reduced controller overhead, surpassing comparable ML-based methodologies reported in [5] and [12]. Moreover, using the InSDN dataset, specifically designed for SDN traffic patterns, the framework attained more realistic outcomes in switch-level scenarios, overcoming a barrier present in previous works that used traditional datasets. These comparisons validate the novelty and efficacy of the suggested SDN-based approach in balancing detection accuracy, scalability, and real-time performance.

V. CONCLUSION

Monitoring of network data has become simpler through Software-Defined Networking (SDN), which also allows more flexible network programmability, enhancing network security. As a result, researchers have increasingly focused on SDN to implement Machine Learning (ML) and Deep Learning (DL) strategies for intrusion detection. In this study, we used the InSDN dataset to evaluate five ML/DL models. Our Python-based simulation successfully demonstrated how SDN can integrate ML techniques to detect cyberattacks in real time, providing a strong foundation for performance optimization in real-world environments.

The results showed that Extreme Gradient Boosting (XGBoost) and Deep Neural Networks (DNNs) achieved superior detection accuracy, albeit with increased processing overhead. In contrast, Decision Trees (DT) and Random Forests (RFs) demonstrated higher throughput and lower processing times, highlighting the trade-offs between detection accuracy and system responsiveness.

Compared to previous works that primarily relied on traditional datasets, our work demonstrates improved adaptability to SDN-specific network traffic, making the suggested framework more realistic for modern programmable networks. The novelty of this study lies in combining cyberattack detection with switch-level performance analysis, enabling the simultaneous assessment of security and scalability under diverse network conditions.

Furthermore, the analysis of flow table sizes and packet distribution highlighted potential scalability concerns, particularly for resource-intensive models. Overall, the findings emphasize the trade-offs between detection accuracy and system responsiveness, offering valuable insights into the feasibility of deploying intelligent security mechanisms in real-time SDN environments.

REFERENCES

[1] A. H. Abdi *et al.*, "Security Control and Data Planes of SDN: A Comprehensive Review of Traditional, AI, and MTD Approaches to Security Solutions," *IEEE Access*, vol. 12, pp. 69941–69980, 2024, https://doi.org/10.1109/ACCESS.2024.3393548.

[2] W. G. Gadallah, N. M. Omar, and H. M. Ibrahim, "Machine Learning-based Distributed Denial of Service Attacks Detection Technique using New Features in Software-defined Networks," *International Journal of Computer Network and Information Security*, vol. 13, no. 3, pp. 15–27, June 2021, https://doi.org/10.5815/ijcnis.2021.03.02.

[3] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "SDN Security Review: Threat Taxonomy, Implications, and Open Challenges," *IEEE Access*, vol. 10, pp. 45820–45854, 2022, https://doi.org/10.1109/ACCESS.2022.3168972.

[4] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey,"

Future Internet, vol. 12, no. 9, Sept. 2020, Art. no. 147, https://doi.org/10.3390/fi12090147.

[5] J. Kim *et al.*, "Enhancing security in SDN: Systematizing attacks and defenses from a penetration perspective," *Computer Networks*, vol. 241, Mar. 2024, Art. no. 110203, https://doi.org/10.1016/j.comnet.2024.110203.

[6] A. AlEroud and I. Alsmadi, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152–164, Feb. 2017, https://doi.org/10.1016/j.jnca.2016.12.024.

[7] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, Mar. 2019, https://doi.org/10.1007/s12083-017-0630-0.

[8] A. Kamisiński and C. Fung, "FlowMon: Detecting Malicious Switches in Software-Defined Networks," in *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, Denver, CO, USA, 2015, pp. 39–45, https://doi.org/10.1145/2809826.2809833.

[9] C. S. Khin, A. T. Kyaw, M. M. Maw, and M. Z. Oo, "Reducing Packet-In Messages in OpenFlow Networks," *ECTI Transactions on Electrical Engineering, Electronics, and Communications*, vol. 20, no. 1, pp. 1–9, Feb. 2022, https://doi.org/10.37936/ecti-eec.2022201.244944.

[10] I. Ali, S. Hong, and T. Cheung, "Quality of Service and Congestion Control in Software-Defined Networking Using Policy-Based Routing," *Applied Sciences*, vol. 14, no. 19, Oct. 2024, Art. no. 9066, https://doi.org/10.3390/app14199066.

[11] S. Badotra and M. Gurusamy, "SecuNet 4D a comprehensive framework for distributed SDN security and resilience," *Scientific Reports*, vol. 15, no. 1, May 2025, Art. no. 15996, https://doi.org/10.1038/s41598-025-98649-x.

[12] A. Hirsi *et al.*, "HSF: A Hybrid SVM-RF Machine Learning Framework for Dual-Plane DDoS Detection and Mitigation in Software-Defined Networks," *IEEE Access*, vol. 13, pp. 112303–112323, 2025, https://doi.org/10.1109/ACCESS.2025.3583712.

[13] O. M. A. Alssaheli, Z. Z. Abidin, N. A. Zakaria, and Z. A. Abas, "Software Defined Network based Load Balancing for Network Performance Evaluation," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, pp. 117–124, Apr. 2022, https://doi.org/10.14569/IJACSA.2022.0130414.

[14] T. Semong *et al.*, "Intelligent Load Balancing Techniques in Software Defined Networks: A Survey," *Electronics*, vol. 9, no. 7, July 2020, Art. no. 1091, https://doi.org/10.3390/electronics9071091.

[15] E. Reticcioli, G. D. Di Girolamo, F. Smarra, A. Torzi, F. Graziosi, and A. D'innocenzo, "Modeling and Control of Priority Queueing in Software Defined Networks via Machine Learning," *IEEE Access*, vol. 10, pp. 91481–91496, 2022, https://doi.org/10.1109/ACCESS.2022.3201823.

[16] L. Boero, M. Cello, C. Garibotto, M. Marchese, and M. Mongelli, "BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch," *Computer Networks*, vol. 106, pp. 161–170, Sept. 2016, https://doi.org/10.1016/j.comnet.2016.06.025.

[17] S. Bhandari, S. K. Sharma, and X. Wang, "Latency Minimization in Wireless IoT Using Prioritized Channel Access and Data Aggregation," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, 2017, pp. 1–6, https://doi.org/10.1109/GLOCOM.2017.8255038.

[18] K. Truong Dinh, S. Kukliński, T. Osiński, and J. Wytrębowicz, "Heuristic traffic engineering for SDN," *Journal of Information and Telecommunication*, vol. 4, no. 3, pp. 251–266, July 2020, https://doi.org/10.1080/24751839.2020.1755528.

[19] K. S. Umadevi, M. S. S. Pranay, and K. Rachana, "Multilevel queue scheduling in software defined networks," in *2017 Innovations in Power and Advanced Computing Technologies*, Vellore, India, 2017, pp. 1–4, https://doi.org/10.1109/IPACT.2017.8245144.

[20] T. Hu, P. Yi, J. Zhang, and J. Lan, "Reliable and load balance-aware multi-controller deployment in SDN," *China Communications*, vol. 15, no. 11, pp. 184–198, Nov. 2018, https://doi.org/10.1109/CC.2018.8543099.

[21] M. Ibrar, L. Wang, N. Shah, O. Rottenstreich, G.-M. Muntean, and A. Akbar, "Reliability-Aware Flow Distribution Algorithm in SDN-Enabled Fog Computing for Smart Cities," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 573–588, Jan. 2023, https://doi.org/10.1109/TVT.2022.3202195.

[22] A. Alharthi, M. Alaryani, and S. Kaddoura, "A comparative study of machine learning and deep learning models in binary and multiclass classification for intrusion detection systems," *Array*, vol. 26, July 2025, Art. no. 100406, https://doi.org/10.1016/j.array.2025.100406.

[23] M. Rostami and S. Goli-Bidgoli, "An overview of QoS-aware load balancing techniques in SDN-based IoT networks," *Journal of Cloud Computing*, vol. 13, no. 1, Apr. 2024, Art. no. 89, https://doi.org/10.1186/s13677-024-00651-7.

[24] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A Novel SDN Intrusion Dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020, https://doi.org/10.1109/ACCESS.2020.3022633.

[25] M. S. Ataa, E. E. Sanad, and R. A. El-khoribi, "Intrusion detection in software defined network using deep learning approaches," *Scientific Reports*, vol. 14, no. 1, Nov. 2024, Art. no. 29159, https://doi.org/10.1038/s41598-024-79001-1.

[26] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System," *IEEE Access*, vol. 10, pp. 64375–64387, 2022, https://doi.org/10.1109/ACCESS.2022.3182333.

[27] S. Jamshidi, K. W. Nafi, A. Nikanjam, and F. Khomh, "Evaluating machine learning-driven intrusion detection systems in IoT: Performance and energy consumption," *Computers & Industrial Engineering*, vol. 204, June 2025, Art. no. 111103, https://doi.org/10.1016/j.cie.2025.111103.

[28] J. C. Mondragon, P. Branco, G.-V. Jourdan, A. E. Gutierrez-Rodriguez, and R. R. Biswal, "Advanced IDS: a comparative study of datasets and machine learning algorithms for network flow-based intrusion detection systems," *Applied Intelligence*, vol. 55, no. 7, Apr. 2025, Art. no. 608, https://doi.org/10.1007/s10489-025-06422-4.

[29] "InSDN dataset." Kaggle. [Online]. Available: https://www.kaggle.com/datasets/badcodebuilder/insdn-dataset.

[30] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, Vienna, Austria, 2021, pp. 1–7, https://doi.org/10.1145/3465481.3469190.

[31] H. Y. I. Khalid and N. B. I. Aldabagh, "A Survey on the Latest Intrusion Detection Datasets for Software Defined Networking Environments," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13190–13200, Apr. 2024, https://doi.org/10.48084/etasr.6756.