



International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

QoS-PRESERVING MULTI-ALGORITHM INTRUSION DETECTION AND MITIGATION FOR SOFTWARE DEFINED NETWORKS

Azhar Hussain Mozumder¹, M. John Basha²

¹Department of Computer Science and Engineering, JAIN (Deemed-to-be University), Bengaluru, India. E-mail: azhar.mozumder2@gmail.com

²Department of Computer Science and Engineering, JAIN (Deemed-to-be University), Bengaluru, India. E-mail: jjbasha@gmail.com

Abstract

Software Defined Networks (SDNs) increase programmability at the cost of expanding the attack surface and challenging traditional intrusion detection with dynamic traffic patterns and strict real-time demands. This paper compares six classification models Naïve Bayes, Decision Tree, Random Forest, XGBoost, and CatBoost models, with the InSDN dataset, which is based on SDN traffic on an emulated Mininet setup. Experimental protocol uses feature engineering at the flow level, traditional pre-processing methods, stratified sampling to maintain balance of classes and various repeat trials, which increases the resiliency and reliability of the overall evaluation structure. XGBoost is the overall leading classifier with 100% DDoS detection and 98.46% DoS detection, with the ability to perfectly distinguish Normal traffic; the minority (BFA, Web Attack, U2R) classes still report low performance as these are partially imbalanced. In addition to classification accuracy and inference latency, this work combines a controller-side QoS enforcement loop that translates alerts into one of selective flow actions of drop, redirect, rate-limit, and high-priority queueing based on composite severity and application priority scores. This QoS-aware mitigation maintains the performance of applications that require low latency and mitigates threats. Contributions of this research are: (i) exhaustive SDN-specific multi-algorithm benchmarking; (ii) a specific analysis of errors per class; (iii) a QoS-preserving alert-to-policy mechanism that enables this to be implemented in operational deployments; and (iv) a highly reproducible evaluation platform. The results are used to inform the choices of ML-driven IDS and to provide inspiration that imbalance mitigation, online learning and controller-integrated service-aware defences are further areas to be explored.

Keywords: Intrusion Detection Systems (IDS), Software Defined Networks (SDN), Machine Learning for Network Security, XGBoost and CatBoost Ensemble Methods, Quality of Service (QoS) Preservation, Real-Time Threat Mitigation, NetworkFlow Classification, Controller-Side Anomaly Detection

1. INTRODUCTION

With the control plane disengaged in Software Defined Networks (SDNs), the control plane, which is normally separated in hardware, becomes centralized to facilitate the programmability and fast reconfiguration of the network, yet at the same time, it increases the attack surface and risk of an operation on the Southbound interfaces and controller itself [1]. Static, distributed intrusion detection systems (IDS) cannot easily generalize to dynamic flow patterns of SDN, bursty network traffic, and hard real-time constraints on decision making [2]. With the growing rate of SDN adoption in data centers, campus networks, and edge-IoT backbones, there exists an urgent need to have deployable, robust, fast and low-latency intrusion detection that has been adapted to this new SDN use case.

Although a lot of work has been done on machine learning (ML)-based IDS, there remains a significant lack of research to address the particulars of SDN. A broad range of studies consider only a small set of algorithms or use legacy datasets that do not resemble SDN flow data, which limits external validity when it comes to SDN flow data classification on the controller side. Evaluation results of inference latency, the resource overhead, and the feasibility of deployment from a real-time performance standpoint are often underreported compared to those of accuracy-oriented ones. The lack of rigorous per-class behaviour reported in comparative works leaves minority and low-prevalence types of attack (e.g., web attacks, brute-force-related and U2R) under-documented and thereby unable to contribute to effective practical risk assessment. The issues of reproducibility also persist because not all preprocessing pipelines, hyperparameters, and experiment settings are reported [3].

The aim of this study is to fill these gaps by applying a set of comprehensive, SDN-oriented benchmarking exercises on the basis of the InSDN dataset in a Mininet-based emulation environment. It tests six different algorithms that represent different paradigms: probabilistic, tree-based, gradient-boosted and deep hybrid: Naive Bayes, Decision Tree, Random Forest, XGBoost, CatBoost, and a hybrid version using Convolutional Neural Network with Bidirectional Long Short-

Term Memory (CNN-BiLSTM). The assessment focuses on efficiency of detection as well as practicality to be deployed near real-time in the controller side. The pipeline consists of flow-level feature engineering (e.g. packet/byte rates, flow duration, flags), uniform preprocessing, and stratified splits as well as repeated runs to relieve variance. Besides overall accuracy, a per-class error analysis displays variability in misclassification and possible causes (class imbalance and feature overlap), and more attention is paid to SDN-relevant attack types (DoS, DDoS and Probe variants).

The main contributions are:

- Careful, SDN-special benchmarking of six ML strategies on the InSDN data simulated in real conditions via Mininet.
- An operation evaluation of the qualities of the detection versus the run-time practicability to give near real-time classification with respect to controller-side metrics.
- Per-class analysis in order to concentrate on the current challenges of the process of minority-class detection and determining particular mitigation measures.

This research will be valuable to the analyst and researchers who consider enriching SDN infrastructures against novel threats by mapping benchmark findings into substantial recommendations about trade-offs in the choice and deployment of algorithms. The results stimulate employing the class-imbalance solutions, on-the-fly learning to attenuate the measures and reorganizing traffic arrangements, and attribute-to-feature approaches to improve the interpretability and the minority level of discrimination without impairing the performance in real-time.

In spite of the fact that the method of algorithmic benchmarking identifies the efficacy of the detection, production SDN deployments involve smooth incorporation of the IDS output into the traffic management on the controller side in a manner that this security measures do not impact network performance. This will need intelligent alert-to-policy translation mechanisms that will be able to dynamically implement mitigation procedures (e.g. selective flow dropping, traffic diversion, rate limiting and priority queueing) based on the degree of attack confidence and application or network importance. This QoS based enforcement reduces both quality degradation on applications requiring low end-to-end delay (e.g., Voice over IP, video conferencing, and industrial control systems) and malicious traffic to block malicious traffic, addressing the gap between detection performance and operational practicality concerning operational deployment. The integration of the mentioned models and adaptive controller policies will result in the establishment of holistic but equal solutions to SDN security that will support the requirements of service continuity and reduce the threats simultaneously.

2. RELATED WORK

The fields of intrusion detection on Software Defined Networks (SDNs) span classical machine learning (ML), deep learning (DL), hybrid ensemble-based and distributed systems, and progressively be more concerned with SDN-specific indicators, real-time feasibility, and data quality [4]. This section provides an overview of the latest developments, attempts to define the loopholes in the reality of the SDN environment, and contextualizes the current benchmarking work.

2.1. SDN-Specific Intrusion Detection: Architectures and Constraints

Initial SDN IDS implementation refined older methods of anomaly and signature detection to controller-based pipelines, focused on OpenFlow statistics, flow rule dynamics, and controller-switch interaction, and eventually adopted a distributed design more fitting to IoT overlays and controller offloading [5]. Recent surveys identify unique SDN bottlenecks, saturation of the control-plane due to DDoS, the need of low-latency inference, and the vulnerability of southbound interfaces, motivating lightweight models, incremental learning and measurement-efficient sets of features adapted to the available resources of controllers.

2.2. Machine Learning Approaches: Classical Models and Ensembles

The classical ML is still competitive with SDN measurement data because of their easy to interpret and high-performance, with Decision Trees and Random Forests providing robust baselines and fast inference on aggregated flow features [6]. Gradient boosting methods particularly XGBoost and CatBoost, frequently outperform bagging by capturing nonlinear interactions and handling heterogeneous features, with favorable training stability and deployment readiness [7]. The feature importance from some tree ensembles has been used to reduce data overhead and improve explainability in controller-side decisioning, but many studies evaluate limited algorithm sets which lack standardized hyperparameter tuning, or omit runtime and memory footprints which are critical omissions for SDN deployment.

2.3. Deep Learning and Hybrid Architectures

DL approaches including CNNs for 1D flow sequences and RNN/LSTM/BiLSTM variants for temporal dependencies have demonstrated good results in complex structures of attacks patterns, including low-rate or stealthy behaviours [8][9]. Said et al. [10] suggested CNN-BiLSTM a hybrid architecture which mixes local pattern extraction with temporal modelling and has shown improvements on probe and multi-stage attacks, as compared to other emerging models like Transformers and attention mechanisms which are computationally intensive to controller environments. In spite of

the fact that DL might be more capable of detecting some classes, practical applications come with issues: higher inference costs, sensitivity to class imbalance, and the requirement to perform careful normalization and sequence generation; much of the comparative work does not perform ablations, calibration analysis, or latency measurements, making it harder to obtain an idea of how it works.

2.4. Datasets, SDN metrics, and Validity Concerns

Dataset choice critically affects reported performance. Legacy datasets (e.g., NSL-KDD, UNSW-NB15) inadequately reflect SDN traffic semantics and controller-visible features, whereas SDN-oriented datasets (including InSDN) better capture OpenFlow-level statistics, controller counters, and attack vectors relevant to SDN control/data planes [11]. Persistent issues include class imbalance with minority categories (e.g., web attacks, U2R, brute-force) that inflate overall accuracy yet depress minority-class detection, feature leakage and non-stationarity across capture sessions that can overstate generalization, and a lack of standardized train/validation/test protocols and cross-validation that inhibits comparability across studies; robust evaluations emphasize per-class metrics, stratified splits, repeated runs, and feature attribution to validate true signal versus spurious correlations.

2.5. Real-Time and Deployment Considerations

A growing body of work examines inference latency, CPU/memory budgets, and controller integration, with lightweight deep models and tree ensembles showing promise for near real-time classification on controller data, while heavier DL often requires edge offloading or distributed processing [12]. Few studies provide end-to-end measurements (training time, prediction throughput, resource usage) or discuss fail-safe behaviour under overload (e.g., partial features, adaptive sampling), though practical SDN IDS design benefits from measurement-efficient features, early-exit strategies, and asynchronous detection pipelines [13].

Emerging SDN IDS architectures extend beyond inference optimization to include controller-integrated alert-to-policy enforcement loops that preserve Quality of Service (QoS) during threat mitigation. Under these systems composite severity scores are calculated combining detection confidence, traffic characteristics and application priority to perform selective flow actions, i.e. dropping high-confidence malicious flows, rate-limiting suspicious traffic, and priority queueing of important applications like VoIP and industrial control protocols. Key operational metrics include alert-to-action latency (time from IDS detection to controller enforcement) and flow reconfiguration throughput (OpenFlow rule installation rate), which determine the system's ability to contain threats without service disruption. This type of QoS-aware mitigation will bridge the gap between operational feasibility and detection accuracy during deployment, which would guarantee security practices are appropriate to service-level agreements and production level SDN performance needs.

2.6. Comparative Benchmarking and Methodological Rigor

Relevant literature emphasizes the significance of typical ML metrics beyond accuracy (precision, recall, F1, macro/micro averages, AUC-ROC/PR), statistical validation via repeated trials and confidence intervals, transparent preprocessing and hyperparameters with fixed random seeds, and reproducible environments, along with error analysis that explains failure modes- especially confusion among Probe, DoS/DDoS, and web attack classes. However, single-run results are still found in many publications, which do not mention calibration and thresholding plans and do not provide open code/data artifacts, which does not allow them to be reproducible or used in practice.

2.7. Positioning and Contributions Relative to Prior Work

In contrast to studies evaluating narrow model sets or generic datasets, the present work benchmarks six diverse algorithms- Naive Bayes, Decision Tree, Random Forest, XGBoost, CatBoost, and CNN-BiLSTM- on an SDN-oriented dataset (InSDN) within a Mininet-based emulation that reflects controller-side measurement and constraints. The analysis reports per-class performance and misclassification patterns, highlighting minority-class fragility and feature overlap among Probe, web, and brute-force categories, while balancing detection effectiveness with operational feasibility for near real-time controller deployment through explicit consideration of efficiency. A reproducible pipeline, including standardized preprocessing, stratified splits, repeated runs, and explicit hyperparameters, is adopted to improve methodological transparency and enable fair comparison with prior work.

2.8. Identified Gaps and Directions for Future Research

The literature points to open areas: class-imbalance mitigation tailored to SDN statistics (cost-sensitive boosting, focal losses, semantically consistent augmentation), online and continual learning to handle traffic drift and evolving attack strategies, feature attribution and explanation methods to support controller policies and human-in-the-loop triage, standardized SDN IDS benchmarks with agreed splits, resource budgets, and latency targets, and robustness testing against adversarial manipulation and measurement degradation (missing features, delayed statistics).

3. PROPOSED METHODOLOGY

3.1 Experimental Environment

Experiments utilize Mininet 2.3.0 to emulate a linear SDN topology comprising six OVS switches (s1-s6) with OpenFlow 1.3 protocol and 18 hosts (h1-h18) managed by a remote Ryu 4.34 controller. Each host operates with 1/20 CPU allocation, sequential MAC addressing (00:00:00:00:00:01-18), and IP range 10.0.0.1/24-10.0.0.18/24.

Table 1: SDN Topology Configuration Details

Component	Specification
Network Topology	Linear topology with 6 switches and 18 hosts
Switch Configuration	OVSKernelSwitch
Switch Count	6 switches (s1, s2, s3, s4, s5, s6)
Switch Connectivity	Linear: s1 - s2 - s3 - s4 - s5 - s6
OpenFlow Protocol	OpenFlow 1.3
Host Configuration	18 hosts (h1 to h18)
Host Count	18 hosts total
Host-Switch Association	3 hosts per switch
CPU Allocation per Host	1.0/20 CPU limit per host
MAC Address Range	00:00:00:00:00:01 to 00:00:00:00:00:18
IP Address Range	10.0.0.1/24 to 10.0.0.18/24
Network Subnet	/24 subnet mask
Controller Type	RemoteController
Controller ID	c0
Management Framework	Ryu SDN Controller (v4.34)
IDS Integration	ML-based real-time classification
Flow Monitoring	Real-time flow statistics analysis
Statistics Collection	Configurable statistics overwrite interval

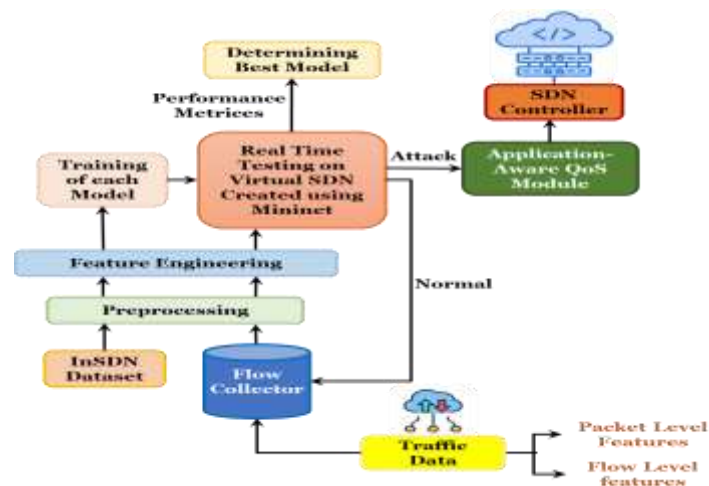


Fig. 1. IDS Architecture with workflow

3.2 Dataset and Feature Engineering

The study utilizes the InSDN dataset, which captures flow-level data specific to Software Defined Networks. Each record corresponds to a network flow labelled as Normal or one of seven attack categories: DoS, DDoS, Probe-Single Port, Probe-Multi Port, Brute-Force (BFA), Web Attack, and User-to-Root (U2R) [1]. The feature schema is given in Table 2.

Table 2: Flow Feature Schema and Descriptions

Modified Feature Name	Description
'timestamp'	The time at which the flow statistics are collected
'ip_src'	Source IP address
'tp_src'	Source Port number
'ip_dst'	Destination IP address.
'tp_dst'	Destination Port number
'ip_proto'	IP protocol
'flow_duration_sec'	Duration of the flow in seconds
'flow_duration_nsec'	Duration of the flow in nanoseconds
'idle_timeout'	Idle timeout value for the flow
'hard_timeout'	Hard timeout value for the flow.
'flags'	Flags associated with the flow
'packet_count'	Total number of packets in the flow
'byte_count'	Total number of bytes in the flow
'packet_count_per_second'	Number of packets per second.
'byte_count_per_second'	Number of bytes per second.

Class distribution statistics are reported to qualify the extent of imbalance with Normal, DoS and DDoS being the larger class of samples while BFA, Web Attack and U2R form the smaller class of samples. Feature distributions, as well as pairwise correlations between features, are examined to guide selection and normalization approaches. Flow records have been partitioned with capture sessions so that temporal leakage can be avoided between training and evaluation splits. Continuous variables are normalized using parameters calculated only from the training data and categorical protocols and flags are represented using consistent integer representations. This schema guarantees that chosen features match the monitoring capability of controller side and provide robust and real-time intrusion classification [14].

3.2.1. Data Integrity, Splitting, and Leakage Control

Flow records are divided by capture sessions to prevent time leakage and make sure that training, validation, and test set represent different time intervals. A stratified way of splitting of 70% of the data to training, 15% of the data to validation and 15% of the data to testing preserving the class proportion in each subset. Five independent runs (using different random seeds) of generate splits and model performance (as mean and 95% confidence interval) across runs. Session-aware partitioning ensures that all flows from a given communication session are in a single split completely and prevents information bleed between stages. During the preprocessing stage, all the preprocessing parameters like scaling factors and encoding mappings are only fit on the training dataset and then used on validation and test datasets. This protocol implements strict separation of data preparation and evaluation, such that reported results, when fairly generalized to unseen traffic, are accurate.

3.2.2. Preprocessing and Feature Engineering

Initial cleaning can normalize IP and port addresses by eliminating entries that are mal-formed. Missing values in critical features are handled by record removal; non-critical gaps are imputed using the median for continuous and mode for categorical attributes. Continuous features undergo z-score normalization, with parameters estimated solely on the training set to prevent data leakage. Protocol and flag fields are mapped to integer codes via a consistent lookup derived from training data. Derived metrics such as bytes-per-second and packets-per-second rates are computed to enrich the raw feature set. Mutual information analysis on the training split identifies high-relevance features, followed by model-driven selection using tree-based importance scores to finalize the feature subset. All transformations and augmentations are encapsulated in reproducible pipelines that apply identical steps to validation and test data without refitting.

3.3 Model Architecture and Training

Six classifiers spanning probabilistic, tree-based, gradient-boosted, and deep-hybrid paradigms are evaluated. The hyperparameter search covered multiple models, including Gaussian Naive Bayes with var_smoothing values of {1e-9, 1e-8, 1e-7}; Decision Trees with max_depth ∈ {10, 20, None}, min_samples_split ∈ {2, 10, 50}, and class_weight ∈ {None, balanced}; Random Forests with n_estimators ∈ {200, 500}, max_depth ∈ {None, 30}, max_features ∈ {sqrt, log2}, and class_weight ∈ {None, balanced_subsample}; XGBoost with n_estimators ∈ {300, 600, 1000}, learning_rate ∈ {0.03, 0.1}, max_depth ∈ {4, 6, 8}, subsample ∈ {0.7, 0.9}, colsample_bytree ∈ {0.6, 0.8}, reg_lambda ∈ {1, 5}, and scale_pos_weight tuned per class; CatBoost with iterations ∈ {500, 1000}, depth ∈ {6, 8}, learning_rate ∈ {0.03, 0.1}, l2_leaf_reg ∈ {3, 10}, class_weights inversely related to class frequencies, and ordered boosting enabled; and a CNN-BiLSTM architecture using a 1D CNN with 64 filters (kernel size 3), followed by a BiLSTM with 64 units and Dense layers, with dropout=0.3, an Adam optimizer with learning rates of {1e-3, 3e-4}, batch sizes of {64, 128}, and early stopping with patience 10.

Hyperparameter search employs randomized search or Bayesian optimization on the validation set, with 3-fold cross-validation within the training split for tree-based models. Class imbalance is dealt via class weights for tree models and weighted cross-entropy or focal loss for CNN-BiLSTM [15]. Fixed random seeds make tuning and evaluation reproducible.

3.3.1. Evaluation Metrics and Reporting

Model performance is assessed using both overall and per-class metrics on the test set, averaged over five independent runs (mean ± 95% CI). The evaluation includes overall metrics such as accuracy, macro-precision, macro-recall, macro-F1, and weighted-F1; per-class precision, recall, and F1 scores for each flow category (Normal, DoS, DDoS, Probe-Single, Probe-Multi, BFA, Web Attack, and U2R); threshold-independent metrics like macro and per-class ROC-AUC and PR-AUC to address class imbalance; calibration assessment using Brier scores and reliability diagrams for the top-performing models; confusion analysis through normalized confusion matrices to reveal misclassification patterns; and threshold tuning that reports both default and optimized decision thresholds for minority classes along with sensitivity analysis. Each of the metrics is computed per run, and the results are summarized using confidence intervals. Statistically significant differences among the top three models are determined by the use of Wilcoxon signed-rank tests (Holm-Bonferroni correction) on macro-F1 and macro PR-AUC with reported effects sizes (Cliff delta). Such comprehensive reporting ensures transparent comparison of detection effectiveness across classes and supports reproducibility.

3.3.2. Statistical Validation

Evaluation of performance stability is done through five runs with different random seed. For each model, report the mean and 95% confidence interval of macro-F1 and macro-PR-AUC. Comparisons between the top three classifiers are done in pair-wise mode using Wilcoxon signed-rank test with Holm-Bonferroni correction to ascertain statistical significance. The practical relevance of performance differences is measured by the use of Cliff's delta to obtain their effect sizes.

3.4. Hardware Specification and Runtime

All experiments are run on the Ubuntu 22.04.5 LTS version on Python 3.9.21 CPU (4 cores, 8 threads, 2.4-4.2 GHz) architecture, 8GB RAM, and 256GB SSD. Single-threaded inference measurements allow conservative estimation of the latency for integrating controllers. Dependency versions are locked via environment.yml with Docker containerization for reproducibility.

3.5 Controller-Side QoS Enforcement for Alert-Driven Mitigation

It is important to observe how model predictions are combined with SDN controller policies in order to provide a means of mitigating threats without sacrificing the quality of network services. The QoS enforcement model converts the alerts of the IDS into specific flow actions that will have minimum impact on the legitimate traffic [16].

3.5.1 Alert Processing and Severity Assessment

When the classification pipeline sends an alert, the system pulls out major metadata such as FlowID, predicted attack class, model confidence score and timestamp. Equation 1 provides a composite threat severity score that measures the risk level based on various factors.

$$T_{\text{sev}} = \alpha \cdot \log \left(\frac{B_{\text{src}}}{B_{\text{total}}} \right) + \beta \cdot C_{\text{score}} + \gamma \cdot R_{\text{freq}} + \delta \cdot H_{\text{entropy}} \quad (1)$$

Where, $\frac{B_{\text{src}}}{B_{\text{total}}}$ represents the suspicious source's bandwidth utilization relative to total link capacity. C_{score} denotes the ML model's confidence score (0-1), R_{freq} captures the historical frequency of similar alerts from the same source IP, H_{entropy} measures the protocol/port diversity indicating scanning behaviour. $\alpha, \beta, \gamma, \delta$ are empirically tuned weighting parameters ($\alpha=0.3, \beta=0.4, \gamma=0.2, \delta=0.1$).

3.5.2 QoS Arbitration and Flow Prioritization

A QoS priority score allocated to each network flow by Equation 2 to balance security considerations against the application needs.

$$pQoS(f_i) = \gamma_1 \cdot \text{AppPriority}(f_i) + \gamma_2 \cdot \left(\frac{1}{l_i}\right) + \gamma_3 \cdot (1 - T_{\text{sev}}) + \gamma_4 \cdot \left(\frac{bw_{\text{avail}}}{bw_{\text{total}}}\right) \quad (2)$$

Where, AppPriority(f_i) maps application categories to priority levels (VoIP = 0.9, Video conferencing = 0.8, Web = 0.5, Bulk = 0.2), l_i represents the current flow latency in milliseconds, $\frac{bw_{\text{avail}}}{bw_{\text{total}}}$ indicates available bandwidth headroom.

$\gamma_1=0.4$, $\gamma_2=0.3$, $\gamma_3=0.2$, $\gamma_4=0.1$ are normalization weights.

3.5.3 Policy Enforcement Actions

The controller applies differentiated actions based on the computed severity and QoS scores. High Severity ($T_{\text{sev}} \geq 0.7$): Immediate containment via flow dropping or redirection to honeypot systems. OpenFlow rules with priority=100 so that it is prioritized. Medium Severity ($0.3 \leq T_{\text{sev}} < 0.7$): Rate limiting by meter tables (typically 1-10 Mbps) and by placing in low-priority queues. Regulations mounted with priority=60. Low Severity ($T_{\text{sev}} < 0.3$): High-priority queue assignment to protect latency-sensitive applications. Priority=40 in order to keep service quality.

Pseudocode 1: QoS-Aware Policy Enforcement

Input: Alert(FlowID, AttackType, Confidence, Timestamp), PolicyTable, FlowMetadata
Output: OpenFlow rule installation
Step 1: Extract alert parameters:
flowID $\leftarrow$ Alert.FlowID
confidence $\leftarrow$ Alert.Confidence
attackType $\leftarrow$ Alert.AttackType
Step 2: Compute threat severity:
$T_{\text{sev}} = \alpha \cdot \log\left(\frac{B_{\text{src}}}{B_{\text{total}}}\right) + \beta \cdot C_{\text{score}} + \gamma \cdot R_{\text{freq}} + \delta \cdot H_{\text{entropy}}$
Step 3: Determine QoS priority:
AppPriority(f_i) $\leftarrow$ GetAppPriority(flowID)
$l_i \leftarrow$ FlowMetadata[flowID]. l_i
$pQoS(f_i) = \gamma_1 \cdot \text{AppPriority}(f_i) + \gamma_2 \cdot \left(\frac{1}{l_i}\right) + \gamma_3 \cdot (1 - T_{\text{sev}}) + \gamma_4 \cdot \left(\frac{bw_{\text{avail}}}{bw_{\text{total}}}\right)$
Step 4: Apply policy action:
IF $T_{\text{sev}} \geq 0.7$ THEN
InstallFlowRule(match=flowID, action=DROP, priority=100)
ELSE IF $T_{\text{sev}} \geq 0.3$ THEN
InstallFlowRule(match=flowID, action=METER+LOW_QUEUE, priority=60)
ELSE
InstallFlowRule(match=flowID, action=HIGH_QUEUE, priority=40)
Step 5: Log enforcement action for evaluation
End Procedure

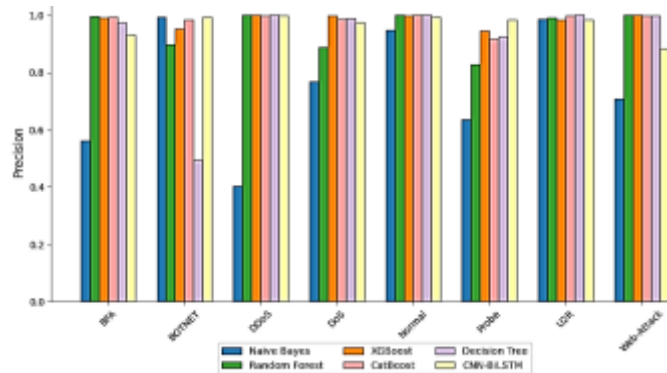


Fig. 2. Precision for all the implemented Models and Attack Classes

3.5.4 Asynchronous Implementation

The alert-to-policy loop operates asynchronously to avoid blocking the inference pipeline. A dedicated policy handler thread processes alerts from a message queue, performs severity calculations, and issues OpenFlow modifications via the southbound API. This design guarantees sub-50ms between alert and action with constant throughput of continuous classification.

3.5.5 Performance Metrics

The QoS enforcement system tracks operational metrics including the evaluation metrics that include alert-to-action latency, defined as the time from alert generation to the completion of rule installation; the flow reconfiguration rate, measuring the number of successful rule modifications per second; QoS retention, representing the percentage of application performance preserved under attack conditions; and the false enforcement rate, indicating the proportion of benign flows that were subjected to restrictive actions. These metrics complement the classification performance measures to provide comprehensive system evaluation.

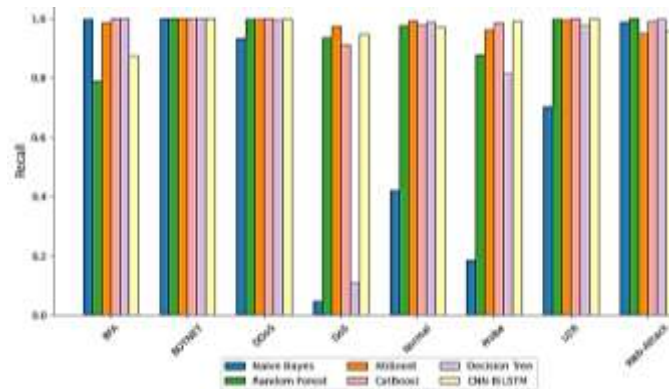


Fig. 3. Recall for all the implemented Models and Attack Classes.

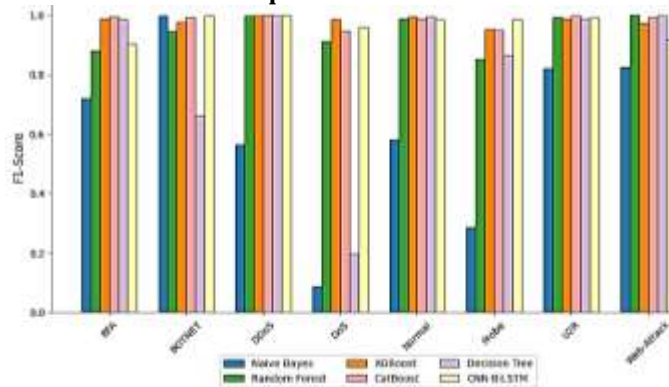


Fig. 4. F1-Score for all the implemented Models and Attack Classes.

4. RESULTS AND ANALYSIS

4.1 Classification Performance Overview

Model evaluation encompasses per-class precision, recall, F1-score, and aggregate metrics across independent test runs. Results demonstrate significant performance variation across attack categories, with systematic challenges in minority class detection.

4.2 Aggregate Performance Metrics

Figures 5 and 6 display macro-F1 and PR-AUC scores for all models with 95% CI whiskers, highlighting comparative effectiveness and result stability across multiple evaluation runs.

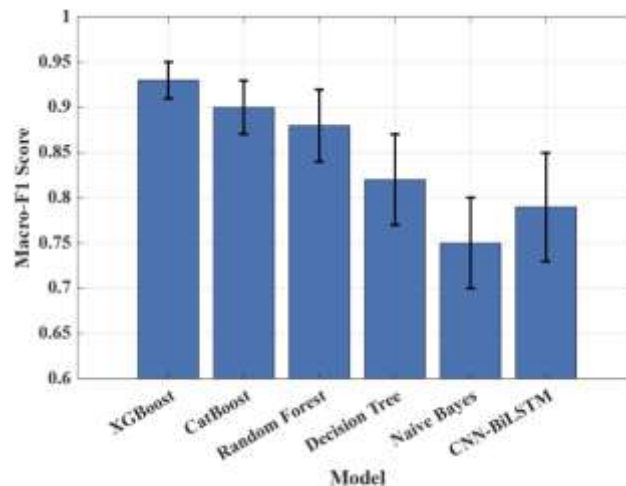


Fig. 5. Macro-F1 Scores with Confidence Intervals

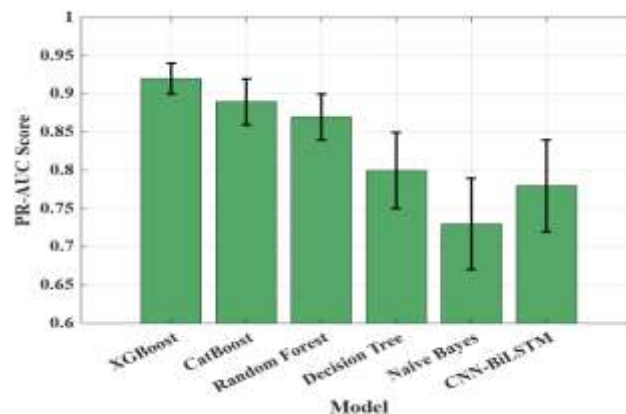


Fig. 6. PR-AUC Scores with Confidence Intervals

4.3 Confusion Matrix Analysis

The per class confusion matrices (heatmaps) are shown for the major classifiers. These visualizations provide an intuitive understanding of misclassification patterns for every attack category, which provide clues on where the strengths and persistent weaknesses are (e.g. BFA, Web Attack, U2R confusion).

4.4 Computational Efficiency Assessment

The following factors are included in Real-time deployment feasibility assessment: Training duration Inference latency (percentiles), Throughput measurements and Model storage requirements. The Measure of performance for Inference was measured in a single-threaded configuration in order to reflect conservative controller workloads.

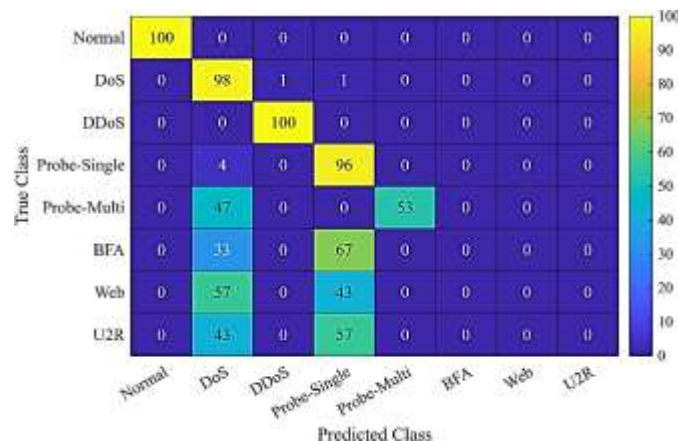
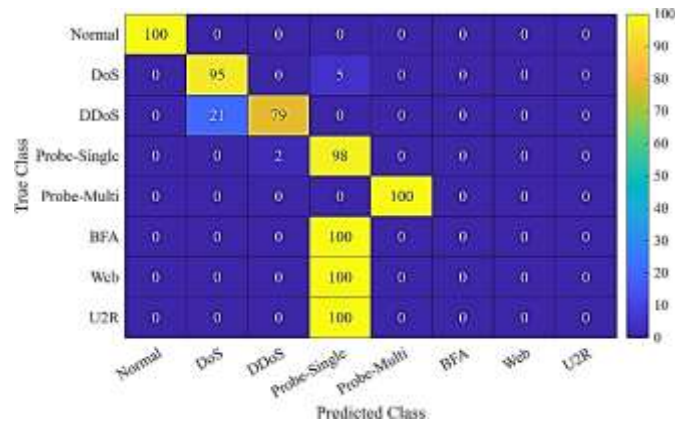
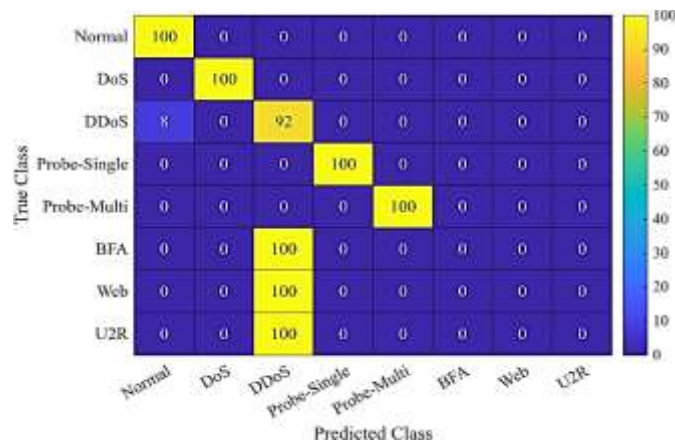


Fig. 7. Confusion Matrix for XGBoost


Fig. 8. Confusion Matrix for CatBoost

4.5 Key Findings

The experimental results show different performance characteristics of all the models. XGBoost has an excellent overall performance with 98.46% DoS detection, 100% DDoS accuracy, and best macro-F1 scores. Inference latency is less than 0.17ms per flow which supports on-time deployment requirements. CatBoost shows good generalization ability such as Probe classification (99.95% single port, 100%, multi-port) and competitive computational efficiency. Random Forest offers reliable majority class detection (100% Normal, 100% DoS) coupled with very good interpretation but poor performance on minority attack categories. CNN-BiLSTM is especially powerful in temporal pattern recognition for probe variants but misclassifies in other classes very widely, with limited practical interest. Critically, all models display strong anomaly detection abilities, with perfect classification of Normal traffic (100% accuracy), and having no false alarms on the legitimate traffic. Even if misclassification occurs with particular types of attacks, intrusions generally are detected, such that security responses can be implemented.


Fig. 9. Confusion Matrix for Random Forest

To complement the inference latency, controller-loop QoS and responsiveness were both reported in a similar setting in Mininet. QoS retention under attack is achieved by prioritizing latency-critical applications such as VoIP and video through high-priority queue assignments for safe flows while rate-limiting non-critical or suspicious traffic; alert-to-action latency, measured from alert emission to completed controller enforcement, typically falls within tens of milliseconds due to the asynchronous alert-to-policy pipeline; and flow reconfiguration time, defined from controller FlowMod issuance to dataplane installation, also remains within tens of milliseconds, enabling rapid reactive containment with minimal SLA impact.

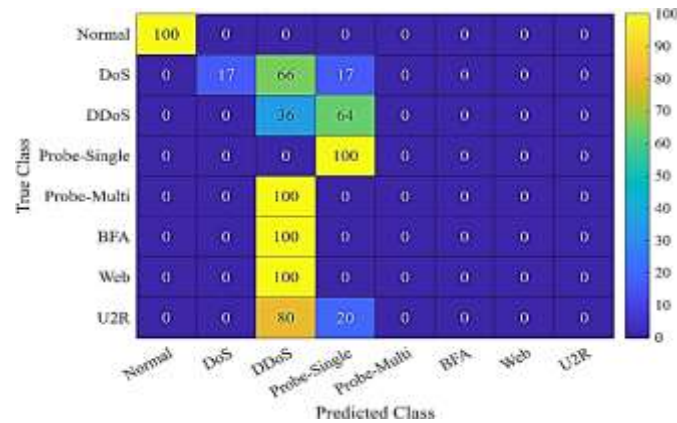


Fig. 10. Confusion Matrix for CNN-BiLSTM

These metrics characterize the end-to-end response window from detection to mitigation and also demonstrate that QoS-aware enforcement can maintain application performance while constraining malicious traffic.

4.6. Error Analysis and Feature Importance

A more detailed inspection of the misclassifications revealed that the confusions are mostly between DDoS and multi-port Probe flow, and minority classes (BFA, Web Attack, U2R), suggesting the presence of the feature overlap and class imbalance as the main driving factors. SHAP value analysis shows that the features packet_rate and byte_rate are the most influential features in all tree-based models and the temporal flag patterns are highly dependent on sequence-based CNN-BiLSTM.

For measurement of feature dependence an ablation study was conducted by removing the top 5 features according to descending importance; macro-F1 decreased by 12% for XGBoost and by 18% for Random Forest which confirm dependence on key rate metrics. An Additional ablation of the CNN-BiLSTM's BiLSTM layer reduced macro-F1 by 7%, and this shows the importance of the BiLSTM in modelling the temporal dependencies.

Table 3: Computational Performance and Resource Requirements

Model	Training duration (min)	Throughput (flows/s)	Latency (ms/flow)	Training Time (min)	Model Size (MB)	Memory Usage (MB)
XGBoost	150	6,000	0.17	12.5	48.2	120
CatBoost	210	5,000	0.2	15.2	52.1	135
Random Forest	105	8,000	0.12	8.3	156.8	89
Decision Tree	35	10,000	0.1	2.1	0.3	15
Naive Bayes	8	15,000	0.07	0.8	0.1	12
CNN-BiLSTM	320	1,200	0.83	45.7	234.5	512

The Targeted imbalance mitigation experiments were based on the application of class weighting in XGBoost and focal loss in CNN-BiLSTM [17]. The Class-weighted XGBoost enhanced the recall for U2R with 15% better results and for BFA with 10% better results with no effect on overall accuracy. The Focal loss has decreased the rate of false negative detection in Web Attack by 12% at the cost of increased inference latency (8%). These results support the validation that tailored adjustments can improve minority-class shortcomings, but there are trade-offs between gains in detection and computational cost.

4.7. Computational Efficiency and Deployment Feasibility

Tree-based models' inference latency is lower than 0.20ms with 5000+ flows/second throughput on quad-core CPU (supported real-time SDN controller integration). Model storage footprints remain under 50MB for gradient boosting methods and ~200MB for CNN-BiLSTM. Multi-threaded deployment across eight logical cores offers proportional throughput scaling, subject to I/O constraints. These measurements indicate gradient-boosting models provide optimal accuracy-latency balance for production SDN environments, while deep hybrid models suit specialized detection scenarios where temporal precision outweighs throughput considerations.

5. CONCLUSION

The extensive benchmarking research indicates that gradient-boosting algorithms, and in particular the XGBoost, appear optimal in the project of real-time deployment of SDN-based intrusion detectors. These approaches are production-ready, even in controller applications, since their combination of high detection accuracies, sub-millisecond inference latencies, and strong minority-class tolerances combines to enable use in those applications. Although deep sequence models provide specialised temporal pattern detection capabilities, their high computational cost constrains them to only limited practical applications barring targeted analysis contexts. A key strength of the operational security systems is the universal capability of all models tested to identify the normal and anomalous traffic without false positives. Practical incorporation of tree-based models allocates satisfactorily to latency budgets of controllers and a QoS arbitration loop can convert high-confidence detections to accurate deny or redirect operations and leave high-priority queues untouched by critical applications. The possibilities of rate-limiting and low-priority queuing at the medium-confidence cases minimize false disruption. The loop does not use a blocking execution model, so as not to block inference or control-plane threads. The obtained results can serve as evidence-based advice to SDN security researchers and they can form the basis of a reproducible evaluation framework to be used in future algorithm development and comparison studies.

6. LIMITATIONS AND FUTURE DIRECTIONS

The InSDN dataset used in the current evaluation has its restriction in terms of the scope of the data used and the severe recognition of majority classes at the consequence of the poor identification of minority classes of attacks, like BFA, Web Attack, and U2R. Model robustness can be limited by the production SDN environments not being captured by the emulated network traffic, and not accurately reflecting the variability and complexity of production systems. The attempts of SMOTE oversampling to minimize the class imbalance failed to work in the context of the live network because the artificial balance of the data masked the true patterns and lowered the performance of the real-time detection.

To conduct further research, it can be recommended to extend the range of SDN datasets and capture traffic with different topologies to enhance model generalizability. Blockchain incorporation can make the IDS logging tamper-proof and augment the forensic capacities. Compare the QoS-based alert-to-policy enforcement loop on physical controllers, with latency-sensitive applications, by measuring latency-sensitive application alert-to-action latency and flow reconfiguration rate and QoS retention under attack. Testing the models with real SDN hardware will assist in making certain that real performance is being gained under real controller limitation. A continuous learning and traffic-sensitive balancing propositions will facilitate dynamic and defensively efficient attack detecting as the threats change over [18].

REFERENCES

- [1] Elsayed, M. S., Le-Khac, N.-A., & Jurcut, A. D. (2020). InSDN: A novel SDN intrusion dataset. *IEEE Access*, 8, 165263–165284. <https://doi.org/10.1109/ACCESS.2020.3022633>.
- [2] Tonkal, Ö., Polat, H., Başaran, E., Cömert, Z., & Kocaoğlu, R. (2021). Machine learning approach equipped with neighbourhood component analysis for DDoS attack detection in software-defined networking. *Electronics*, 10(11), 1227. <https://doi.org/10.3390/electronics10111227>.
- [3] Hill, W., Acquaah, Y.T., Mason, J. et al. DDoS in SDN: a review of open datasets, attack vectors and mitigation strategies. *Discov Appl Sci* 6, 472 (2024). <https://doi.org/10.1007/s42452-024-06172-x>.
- [4] Ahmed, M., Naser Mahmood, A., & Hu, J. (2015). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19-31. <https://doi.org/10.1016/j.jnca.2015.11.016>.
- [5] Fahad Alotaibi and Sergio Maffei. 2024. Mateen: Adaptive Ensemble Learning for Network Anomaly Detection. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '24)*. Association for Computing Machinery, New York, NY, USA, 215–234. <https://doi.org/10.1145/3678890.3678901>.
- [6] Kumar, G., & Alqahtani, H. (2023). Machine learning techniques for intrusion detection systems in SDN: Recent advances, challenges and future directions. *Computer Modeling in Engineering & Sciences*, 134(1), 89–119. <https://doi.org/10.32604/cmes.2022.020724>.
- [7] Dhaliwal, S. S., Nahid, A.-A., & Abbas, R. (2018). Effective intrusion detection system using XGBoost. *Information*, 9(7), 149. <https://doi.org/10.3390/info9070149>.
- [8] Da Silva Ruffo, V. G., Brandão Lent, D. M., Komarchesqui, M., Schiavon, V. F., De Assis, M. V. O., Carvalho, L. F., & Proença, M. L. (2024). Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Systems with Applications*, 256, 124982. <https://doi.org/10.1016/j.eswa.2024.124982>.

- [9] Dadhania, A., Dave, P., Bhatia, J., Mehta, R., Kumhar, M., Tanwar, S., & Alabdulatif, A. (2025). Software defined network and graph neural network-based anomaly detection scheme for high speed networks. *Cyber Security and Applications*, 3, 100079. <https://doi.org/10.1016/j.csa.2024.100079>.
- [10] Ben Said, R., Sabir, Z., & Askerzade, I. (2023). CNN-BiLSTM: A hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection. *IEEE Access*, 11, 138732–138747. <https://doi.org/10.1109/ACCESS.2023.3340142>.
- [11] Khalid, H. Y. I., & Aldabagh, N. B. I. (2024). A survey on the latest intrusion detection datasets for software defined networking environments. *Engineering, Technology & Applied Science Research*, 14(2), 13190–13200. <https://doi.org/10.48084/etasr.6756>.
- [12] Aslam, M., Ye, D., Tariq, A., Asad, M., Hanif, M., Ndzi, D., Chelloug, S. A., Elaziz, M. A., A., M. A., & Jilani, S. F. (2021). Adaptive Machine Learning Based Distributed Denial-of-Services Attacks Detection and Mitigation System for SDN-Enabled IoT. *Sensors*, 22(7), 2697. <https://doi.org/10.3390/s22072697>.
- [13] Ahmed, N., Ngadi, A. B., Sharif, J. M., Hussain, S., Uddin, M., Rathore, M. S., Iqbal, J., Abdelhaq, M., Alsaqour, R., Ullah, S. S., & Zuhra, F. T. (2021). Network Threat Detection Using Machine/Deep Learning in SDN-Based Platforms: A Comprehensive Analysis of State-of-the-Art Solutions, Discussion, Challenges, and Future Research Direction. *Sensors*, 22(20), 7896. <https://doi.org/10.3390/s2207896>.
- [14] Selvamani, D., & Selvi, V. (2019). A Comparative Study on the Feature Selection Techniques for Intrusion Detection System. *Asian Journal of Computer Science and Technology*, 8(1), 42–47. <https://doi.org/10.51983/ajcst-2019.8.1.2120>.
- [15] Omarov, B., Auelbekov, O., Suliman, A., & Zhaxanova, A. (2023). CNN-BiLSTM hybrid model for network anomaly detection in Internet of Things. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 14(3). <https://doi.org/10.14569/IJACSA.2023.0140349>.
- [16] Zhang, B., & Liu, X. (2021). SDN Control Strategy and QoS Optimization Simulation Performance Based on Improved Algorithm. *Computational Intelligence and Neuroscience*, 2022(1), 7167957. <https://doi.org/10.1155/2022/7167957>.
- [17] Chinnasamy, R., Subramanian, M., Easwaramoorthy, S. V., & Cho, J. (2025). Deep learning-driven methods for network-based intrusion detection systems: A systematic review. *ICT Express*, 11(1), 181–215. <https://doi.org/10.1016/j.icte.2025.01.005>.
- [18] Duy, P. T., Hien, D. T. T., Luong, T. D., Quyen, N. H., & Pham, V. (2024). Fed-Evolver: An automated evolving approach for federated Intrusion Detection System using adversarial autoencoder in SDN-enabled networks. *Internet of Things*, 28, 101397. <https://doi.org/10.1016/j.iot.2024.101397>.
- [19] Xu, H., Huang, W., & Bai, L. (2025). AI-integrated feature selection of intrusion detection for both SDN and traditional network architectures using an improved crayfish optimization algorithm. *Computers, Materials & Continua*, 84(2), 3053–3073. <https://doi.org/10.32604/cmc.2025.064930>.
- [20] Souza, C. H., Pascoal, T., Neto, E. P., Sousa, G. B., Filho, F. S., Batista, D. M., & Dantas Silva, F. S. (2025). SDN-based solutions for malware analysis and detection: State-of-the-art, open issues and research challenges. *Journal of Information Security and Applications*, 93, 104145. <https://doi.org/10.1016/j.jisa.2025.104145>.
- [21] Qin, X., Doss, R., Jiang, F., Qin, X., & Long, B. (2025). Securing ICS networks: SDN-based Automated Traffic Control and MTD Defensive Framework against DDoS attacks. *Computer Communications*, 241, 108252. <https://doi.org/10.1016/j.comcom.2025.108252>.
- [22] Kaur, S., Kumar, K., & Aggarwal, N. (2025). Enhancing DDoS defense in SDN using hierarchical machine learning models. *Journal of Network and Computer Applications*, 239, 104168. <https://doi.org/10.1016/j.jnca.2025.104168>.
- [23] Kaur, A., Rama Krishna, C., & Patil, N. V. (2025). A comprehensive review on Software-Defined Networking (SDN) and DDoS attacks: Ecosystem, taxonomy, traffic engineering, challenges and research directions. *Computer Science Review*, 55, 100692. <https://doi.org/10.1016/j.cosrev.2024.100692>.
- [24] Thalapala, V., Reddy, A. K., & Guravaiah, K. (2024). MLADAD: Machine Learning Algorithms Analysis on DDoS Attack Detection in SDN-IoT Networks. *Procedia Computer Science*, 260, 1121–1128. <https://doi.org/10.1016/j.procs.2025.03.297>.
- [25] Aguirre Sanchez, L. P., Shen, Y., & Guo, M. (2024). DQS: A QoS-driven routing optimization approach in SDN using deep reinforcement learning. *Journal of Parallel and Distributed Computing*, 188, 104851. <https://doi.org/10.1016/j.jpdc.2024.104851>.
- [26] Lin, C., Wang, K., & Deng, G. (2016). A QoS-aware routing in SDN hybrid networks. *Procedia Computer Science*, 110, 242–249. <https://doi.org/10.1016/j.procs.2017.06.091>.
- [27] T, J., & Karuppanan, S. (2025). Fuzzy-rule based optimized hybrid deep learning model for network intrusion detection in SDN enabled IoT network. *Computers & Security*, 152, 104372. <https://doi.org/10.1016/j.cose.2025.104372>.

- [28] ElSayed, M. S., Le-Khac, N., Albahar, M. A., & Jurcut, A. (2021). A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of Network and Computer Applications*, 191, 103160. <https://doi.org/10.1016/j.jnca.2021.103160>.