Kufa Journal of Engineering
Vol. 17, No. 3, July 2026, P.P. 48-75
Article history: Received 26 May 2025, last revised 4 August 2025,
accepted 11 August 20

ENHANCING SOFTWARE-DEFINED NETWORKING PERFORMANCE THROUGH ARTIFICIAL INTELLIGENCE: A COMPREHENSIVE ANALYSIS

Ola Marwan Assim[1], Qutaiba I. Ali [2], Zahraa T. Al Mokhtar[3] and Nawal Y. Abdullah[4]

[1] Department of Computer Engineering, Collage of Engineering, University of Mosul, Mosul, Iraq, Email:ola.marwan@uomosul.edu.iq.

[2] Department of Computer Engineering, Collage of Engineering, University of Mosul, Mosul, Iraq, Email:Qut1974@gmail.com.

[3] Department of Computer Engineering, Collage of Engineering, University of Mosul.

[4] Artificial Intelligence Techniques Engineering Department Northern Technical University.

https://doi.org/10.30572/2018/KJE/170304

ABSTRACT

This paper presents an approach that incorporates classical Artificial Intelligence techniques inside the software Defined Networking (SDN) for traffic classification and makes the controller more responsive besides adding intelligence to the network. Three of the most popular machine learning classifiers frequently used by researchers are evaluated: Naïve Bayes, Support Vector Machine, and Nearest Centroid. These are either embedded or attached to three leading SDN platforms (Ryu, ONOS, and OpenDaylight). An elaborate experimental methodology using Mininet was developed to test several configuration parameters and policies both in a real environment as well as under synthetic conditions. Results show that out of all flows classified with over 94% accuracy by Naïve Bayes classifier keeping decision latency at only 8.2 ms on the controller, meanwhile traditional SDN setup with AI has a decision latency of 15 .7ms. The Open Network Operating System (ONOS) showed a maximum sustained bandwidth of 85 Mbps with AI help while reducing the resource usage by as much as 30% on externalized feature extraction. This result, therefore, clearly proves that lightweight AI has emphasized impacts due to proper parameterization without incurring huge additional computational costs. The best project's efforts are in proposing methods for the selection of controllers and tuning parameters together with hybridization between AI and SDN network strategies suitable either for real-time or large-scale networks.

KEYWORDS

Software-Defined Networking, Machine Learning, Artificial Intelligence, OpenFlow Protocol, SDN Controller.

1. INTRODUCTION

Software Defined Networking (SDN), which represents a paradigm shift in which the data and control plane are separated, can facilitate an on demand flow adaption in terms of resource management via centralized, programmable infrastructure (Raikar et al., 2020). This migration has allowed for effective cost, Scalable, and fixed network process specially in heterogeneous and dynamic environment (Braun and Menth, 2014; Sahoo et al., 2016). Still, there are different practical challenges in operating SDN, such as high controller decision latency, deficiency of context awareness to make traffic decision and suboptimal load balancing strategies with fast changing network condition among others (Xie et al., 2018; Latah and Toker, 2019; Latah and Toker, 2016). Many research has been newly done on integrating AI and ML in SDN control planes (Zhang and Liu, 2019). Intelligent Methods have ability to improve at traffic classification. Deep learning, reinforcement learning, and ensemble based classifier (Wu et al., 2020) have been proposed in literature as a solutions for intelligent SDN control nevertheless their complexities and computational overhead often prevent real time deployment, particularly in resource limited or latency sensitive setting (Dritsas and Trigka, 2025). Alternatively, fast and light inference classifier such as Naïve Bayes (Hnamte and Balram, 2022), Support Vector Machines (SVM), and Nearest Centroid algorithms are easier to integrate thus more compatible with practical SDN system because they have faster inference speed. (Myint et al., 2019) discusses such traditional machine learning models on AI-driven SDN platforms by detailing different controller solution responses (Ryu, ONOS, and Open Daylight) towards embedded AI model modules under different configurations and deployment approaches based on a well-articulated experimental setup using performance metrics like controller decision time, throughput, CPU & memory utilization, and end-to-end latency under various traffic scenarios. This paper also observes the trad off between inference performance and overhead of controller for providing guidelines to deploy AI-enabled SDN system in efficient and scalable way. The major contribution of this paper:

• Introduce an architecture for incorporating classic AI based classifier into SDN controller with embedded inference pipelines.

• Running the full experimental benchmark of the most widely SDN protocol (Ryu,ONOS and openDaylight) on various operation mode as well in diverse deployment scenario.

• Providing a quantitative comparison of the trad offs in classification performance, controller decision time and resourse consumption on AI with SDN against base line case.

• Arrangement direct guidance for controller platforms and AI model in the network architectural constrains.

This paper is ordered as follows: Section 2 presents the outlines of the theoretical background of the AI techniques and SDN controller. Section 3 presents related work and positions this study among recent efforts. Section 4 describes the details of the system architecture and AI integration pipeline. Section 5 describe the setup configuration and experimental environment. Section 6 describe the details of dataset and AI hypermeter model and validation process. Section 7 presents the core performance results and analysis. Section 8 discusses controller complexity, trade-offs, and AI inference overhead. Section 9 examines scenario-based application evaluations. Section 10 provides a critical discussion of results with visual performance comparisons. Section 11 gives comparison with previous works and finally section 12 concludes the paper and suggests future research directions.

2. ARCHITECTURAL DESIGN OF SDN

Open Networking Foundation (ONF) defines a high level of SDN architecture with three separate service layers: Infrastructure (Forwarding), Control and Application. This architecture allows for logical partitioning and horizontal decomposition of networking functions (Tourrilhes et al., 2014, Braun and Menth, 2014).

2.1. Infrastructure Layer (Forwarding Layer)

This layer includes the physical network infrastructure with hardware forwarding elements, switches and router base stations (BSs), and with software switches, offering programmability through open interfaces. This layer provides the means for communicating and transmitting datagrams between devices on a network (Bellavista et al., 2020).

2.2. Control Layer (Control Plane)

Software-based SDN controllers are at this layer. It allows network administrators to define and apply policies on the underlying hardware network devices forming the backbone of the corporate network (Xie et al., 2015).

2.3. Application Layer

This layer connects to end user's business applications calling services provided by the SDN architecture. Business applications that can be realized on top of SDN include environmentally friendly internet, security surveillance systems and virtual networks, as illustrated in Fig. 1 (Thirupathi et al., 2019).

3. OPENFLOW PROTOCOL

Open Networking Foundation (ONF) is constantly developing and standardizing OpenFlow (OF). SDN controller can safely connect with OF-enabled forwarding elements because to the

abstraction layer that OF provides (Braun and Menth, 2014). For south-bound APIs used in SDNs, OpenFlow has emerged as the de facto standard. A standard OF-enabled switch uses its flow table to determine how to handle incoming packets. The matching rules section of OF version 1.0.0 is depicted in Fig. 2. Three fields are used to describe the incoming packets.

1. Match fields: to match against packets depending on Fig. 3 (Dargahi et al., 2017).
2. Counters: to update for matching packet.
3. Actions: to apply to matching packets.

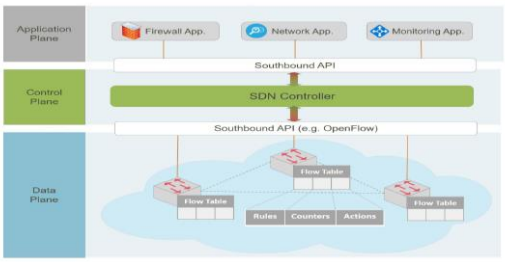

Fig. 1. Basic architecture of SDN

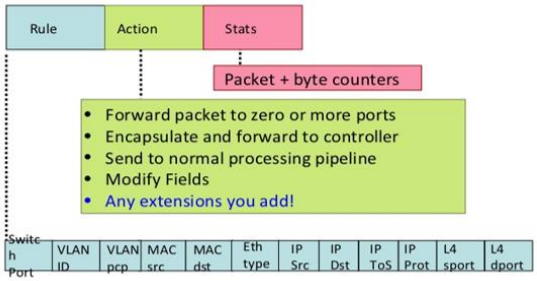

Fig. 2. Open flow entries

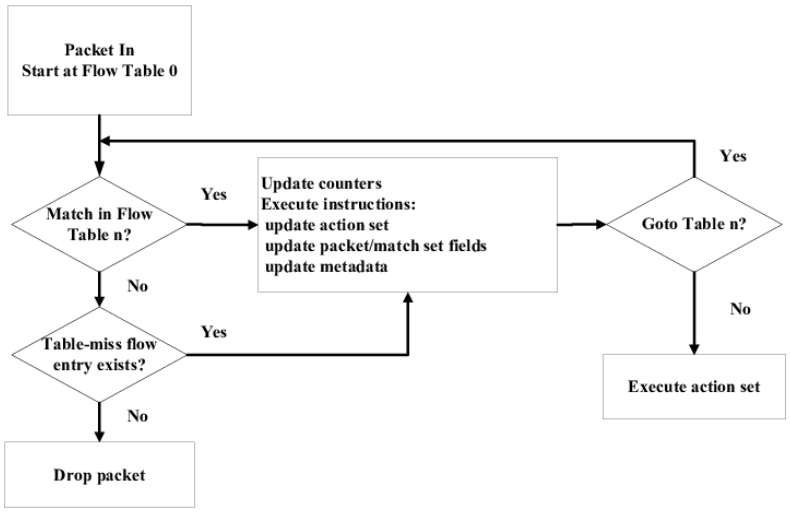

Fig. 3. Packet flow in SDN switch

To understand the architecture of Software-Defined Networking (SDN), we need to look at the fundamental principles on which it operates. The operating mode of an OpenFlow-based SDN network is schematically depicted in Fig. 4 and further explained in Reference (Sarvade and Kulkarni, 2024). One of the important elements of an OpenFlow switch is its flow table. As shown in Fig. 5, the switch communicates with the SDN controller (Dargahi et al., 2017) by means of the OpenFlow protocol. This protocol is used for control message transactions between switches and the controller. The OpenFlow switch has a flow table that holds entries for describing forwarding rules in the data plane. As illustrated in Fig. 5, when a packet is received, a flow lookup immediately starts in the data plane of the OpenFlow switch, initiating a flow comparison.

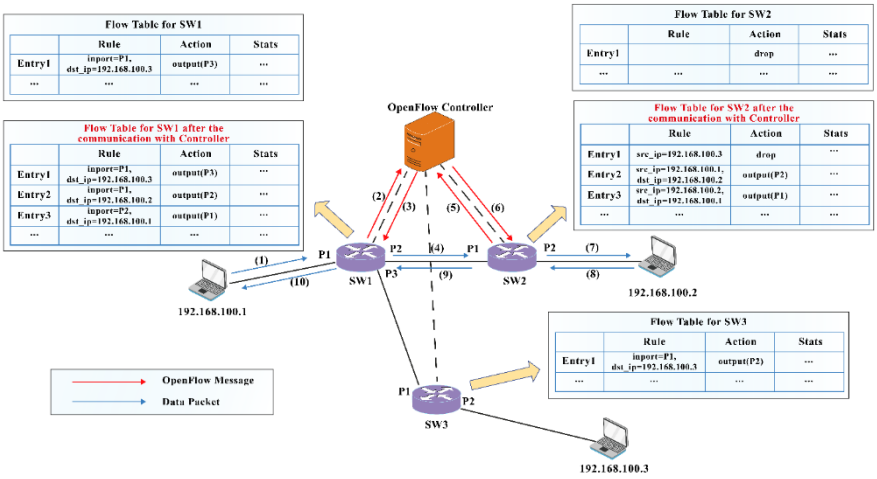

Fig. 4. SDN network

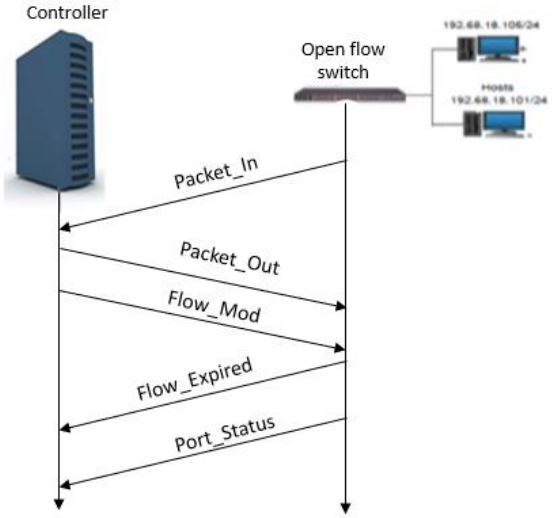

Fig. 5. Communication establishment between host and the OF network

The switch parses the header fields in the received packets and uses the parsed values to match entries in the flow table. If a match is found, the switch processes subsequent packets based on the operations specified in the matching flow entry. If, however, no matching flow entry exists, the switch sends an OpenFlow "Packet_In" message to the controller (through path 2 or 5). The header (and optionally the payload) of the triggering packet is encapsulated in this message. The controller then pushes OpenFlow "Flow_Mod" messages to the switch (represented by arrows 3 and 6), which install new flow entries in its flow table. These new entries instruct the switch on how to handle the current packet and all other packets from the same flow in the future.

4. SIMPLE EXAMPLE IN SDN NETWORK

As we see in Fig. 6 end to-end packet latency. Packets originate at the source host and traverse a number of switches before reaching the destination host. Whole SDN switch delay consists of (Chefrour, 2021):

1 The Stack Delay (StKD) is a source/destination host protocol stack delay.

2 Transmission delay (TD) at the source and destination.

3 Transmission medium propagation Delay (PD).

4 The switch delay (SD) due to the packet forwarding in the switches as demonstrated in Fig.7.

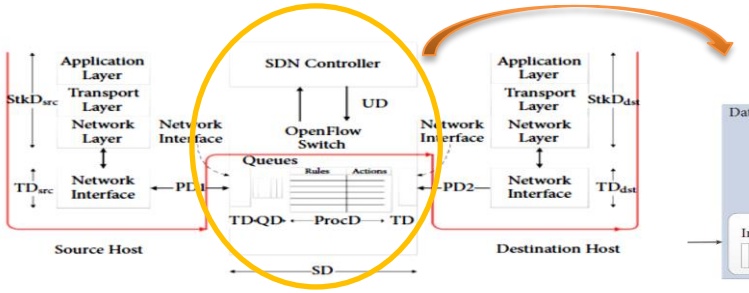

Fig. 6. The Sketch end to end delay of SDN

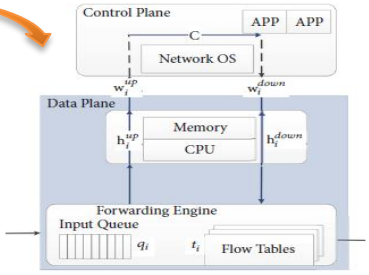

Fig. 7. The Delay of SDN

The end-to-end delay is shown in Eq. 1 [7-16].

$$D = StKD_{src} + TD_{src} + PD + SD + TD_{dst} + StkD_{dst} \quad (1)$$

The total delay can be expressed using the formula below:

$$D_{switch} = q_i + t_i + h_i^{up} + w_i^{up} + C + w_i^{down} + h_i^{down} \quad (2)$$

If the packet matches an entry in the flow table, the delay model can be simplified as:

$$D_{by\,pass} = q_i + t_i \quad (3)$$

This work considers three widely adopted open-source SDN controllers (Ryu, ONOS, and OpenDaylight (ODL)) each representing a distinct architectural and operational approach to managing the control plane of software-defined networks. These controllers were selected to discover how their internal designs, and scalability features influence overall performance when integrated with AI-based traffic classification mechanisms.

Ryu is a streamlined python SDN controller framework that provides a simple and modular environment for developing OpenFlow-based applications. Its event-driven architecture and support for OpenFlow 1.x make it suitable for academic prototyping and low-complexity network implementation. Due to its single-threaded nature with no native cluster support, this makes Ryu better suited for small to medium sized topologies with moderate performance requirements. In AI driven SDN systems, Ryu provides minimum overhead which is suitable to embed lightweight ML models inside the controller logic itself. However, its performance with high flow rates or concurrent processing loads may be limited by the Python interpreter and single-core scheduling (Shirvar and Goswami, 2021).

Open Network Operating System (ONOS) is a an expensive distributed SDN controller with scalability and fault tolerance features. It's implemented in Java and supports a variety of protocols that are southbound, including OpenFlow and NETCONF. ONOS was designed to

function as a clustered system, with multiple instances of the controller that are active in parallel to provide high availability, fast relocation, and horizontal expansion. Its multiple-threaded processing abilities facilitate the efficient management of large amounts of control information and real time flow management. ONOS supports native support for intent-based networking and modularity in applications, which is ideal for scientific research and practical applications. Its JVM-based runtime is intended to support external AI components and microservices, this makes it a strong candidate for integration with or hosting of classification engines in intelligent SDN frameworks (Neelakrishnan, 2016).

OpenDayLight (ODL) is another large-scale, Java-based SDN controller that is intended to support a variety of protocol and application types. As a platform that is built around the OSGi framework, ODL facilitates the loading of dynamic services, the reuse of components, and the control of behavior at a fine-grained level. It supports OpenFlow, BGP, NETCONF, and other interfaces that are southbound, it also features a highly versatile northbound API that can be used to integrate custom applications that are powered by machine learning. ODL is also capable of operating in clustered configurations, this provides redundancy and controls distributed across the network. While its architecture has an additional memory and CPU cost compared to lightweight controllers like Ryu, it provides additional services, including policy engines, topography discovery, and monitoring tools, which are beneficial when combined with AI- enhanced decision making and adaptive flow control strategies (Singh et al., 2022).

These three controllers form a representative cross section of typical SDN implementations, each with its own strengths and weaknesses. By testing them in common use cases that involve AI, this paper attempts to highlight the impact of architectural choices such as programming language, threading model, clustering support and modularity on the scaling and evolution capacities of SDN platforms as well as their ability to take advantage or benefit from machine learning based traffic intelligence. The Ryu, ONOS and OpenDaylight are actively involved by both academic research community and industry practitioners confirms performance comparison across a broad set of operational contexts.

5. ARTIFICIAL INTELLIGENCE IN SDN

Artificial Intelligence in SDN: Soft computing and AI are increasingly used in current systems, including intelligent transportation. This allows us to boost the performance of our current computer networks. Fig. 8 (Wu et al., 2020) illustrates how integrating the abstraction idea in the SDN paradigm with AI approaches can result in more flexible network behavior. It will also bring new techniques for dealing with both classic network challenges and new SDN-related ones, as shown in Fig. 9 (Wu et al., 2020).

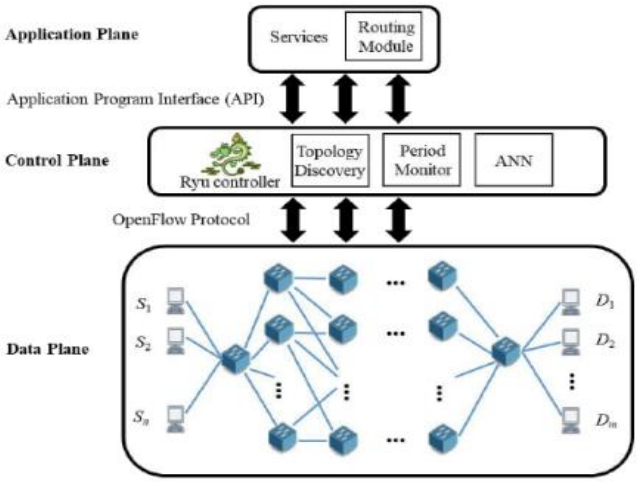

Fig. 8. SDN with the proposed artificial intelligence enabled routing (AIER)

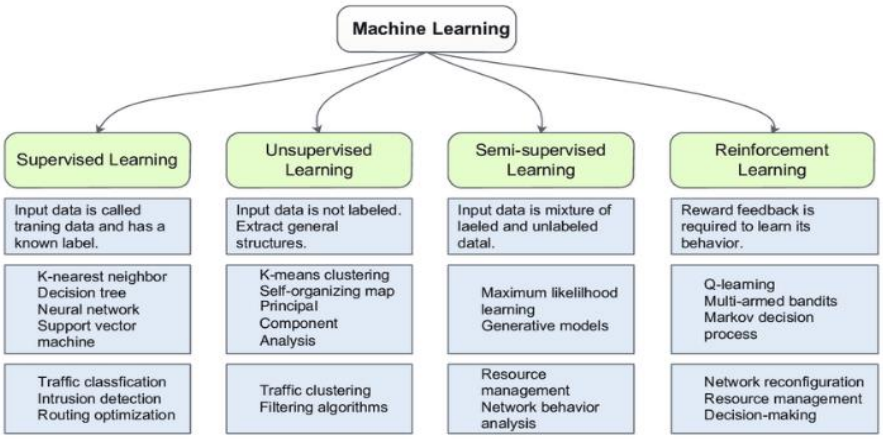

Fig. 9. Common Machine Learning Algorithm Applied in SDN

Three popular machine learning classifiers: Naïve Bayes, Support Vector Machine (SVM) and Nearest Centroid Classifier are used in this work as they could be easily fit into the real-time SDN environment. The two methods bring different advantages in computational complexity, model accuracy and prediction time. All these are important for deployment as part of SDN controllers or side by side with them.

Naïve Bayes (NB) is a probabilistic algorithm which assumes that there's no correlation between features. However, it has been extensively adopted in areas which have the need for a swift decision-making based on simple models. In the field of SDN traffic classification, NB provides a very fast inference and low computational overhead which qualifies it as an excellent candidate for in-line deployment at control loops. Although it makes overly simplistic assumptions, this method works well when features describing network flows (like protocol type, the duration of a flow or number of packets transmitted in one) are informative and approximately independent. Its training is efficient and does not require much data pre-processing, which also meets the real-time requirements of SDN systems (Al-Srarateе and Al-Azawei, 2025).

In contrast, Support Vector Machine (SVM), is more computationally intensive with robustness and high classification accuracy. SVM identify the optimal boundaries that separate different classes in the feature space. While SVM can handle non-linearly separable data effectively using kernel functions, their inference cost grows with the number of support vectors, which may pose challenges in latency-sensitive SDN scenarios. However, when working with simple linear kernels and optimized datasets, SVM performance has been well at recognizing traffic irregularities, classifying the behavior of flows, and supporting intelligent transportation decisions in complex systems (ALesawi and Alawadi, 2025).

Nearest Centroid Classifier is a lightweight, distance-based technique. It computes the mean feature vectors of flow classes and the new instance is assigned to that class whose centroid has the least Euclidean distance from its feature vector. This ensure to be the most computationally efficient classifier with minimal processing requirements during both training and inference phases, therefore this makes it suitable witin SDN environments which needing fast plus large-scale decision making. While accurate only in cases of non-overlapping distributions, there exist enough applications for which different traffic types possess distinct statistical signatures wherein an average classification result would suffice (ALesawi and Alawadi, 2025, Ali and Hreshee, 2023).

The three classifiers, Naïve Bayes, SVM, and Nearest Centroid allow full coverage of exploration on tradeoffs between performance and within the integration of AI-SDN. Naïve Bayes is finally selected due to its low overhead-and stable accuracy for deployment in live evaluation herein this study. SVM and Nearest Centroid are used during training and validation phases to target classification performance as well as resource requirements forced by them all models were estimated using consistent datasets and validation methods to confirm comparability. Their pertinence to SDN is also justified by previous works since the community continuously demands interpretable low-latency classifiers that can work under strict timing constraints without losing accuracy.

6. RESEARCH METHODOLOGY

This section explains the framework used in the study and provides additional details concerning the derivation of performance metrics, application of equations to calculate the overall end to end delay, throughput using transmitted bytes and total transmission time, packet loss ratio based on the number of packets sent versus successfully received, AI training process, model configuration, and dataset used in this research.

The experiments can be conducted using a controlled SDN testbed built with Mininet network simulator. The emulated topology comprised 10 OpenFlow 1.3 switches and 20 hosts arranged

in a multi-tier architecture. Three SDN controller were evaluated under different deployment strategies (centralized, distributed, and containerized). The experimental system can be operated on a workstation equipped with an Intel Core i5 processor, 8GB RAM and Ubuntu 20.04 LTS VMware.

To assess the controller under realistic condition, we exposed them in to heterogenous traffic load as HTTP web browsing, video streaming, and RTP based VoIPby using combination of iperf3, D-ITG, and TCP replay. We maintained rigorous control over flow pattern and packet rates, ensuring that the experiments were cloud reproduced and consistent. Each test iteration was performed 5 times to allow for the calculation of mean values and 95% confidence intervals.

Data collection depend on on a dual monitoring approach, utilizing internal controller logs alongside external tracking tools to arrest a comprehensive performance outline. The analysis focused on several key performance indicators, primarily controller decision latency, throughput, end-to-end delay, and packet loss. These values can be computed using time stamped flow logs, switch statisics,and measured latency between traffic endpoints. Eq.1 to 3 applied during post processing of the raw logs.

Eq.1 computes the overall end-to-end delay as the amount of transmission, propagation, and queuing delay across the path.

Eq.2 determines throughput for transmitted bytes and total transmission time.

Eq.3 estimates packet loss ratio depend on the number of backets sent against successfully received.

These equations can be implemented in a python-based analysis script to ensure consistency across all scenarios.

To support flow type identification, three classification models were applied (Naïve Bayes, Support Vector Machine, and Nearest Centroid). The models can be implemented using scikit-learn library and trained offline prior to deployment. The goal was classifying network flows into one of four classes: Web, video, VoIP, TCP transfer.

The dataset was split to 80% for training and 20% for testing using stratified sampling to maintain class balance. Model evaluation employed 10-fold stratified cross-validation on the training set to reduce overfitting and improve generalizability.

Hyperparameter setting for each classifier could be selected based on grid search over the validation fold: Naive Bayes: Gaussian NB model with var_smoothing = 1e-9, SVM: Linear kernel with regularization parameter C = 1.0 and probability = False, Nearest Centroid: Euclidean distance metric, no tuning required.

The dataset used for training and evaluation consisted of 15,000 labeled flow records compiled from synthetic traffic generated within Mininet simulator. Each flow record contained 9 features known to be effective in SDN aware classification, including protocol type, source and destination port numbers, flow duration, packet count, average packet size, inter arrival time mean, source and destination IPs.

Min-Max scaling can be used to normalized numerical features, and categorical features can be encoded using one of the encoding strategies. The class distribution in the dataset was approximately: 36% for web, 28% for video, 22% for VoIP, and 14% for bulk TCP. After preprocessing the dataset can be divided int training samples (10,000), validation samples (2,500) and test samples (2,500) sets. This ensures the trained classifier reflected both synthetic and real-world behavior across common flow types.

Naive Bayes classifier can be integrated with the SDN controller in two modes: native embedding for Ryu and RESTfull microservice for ONOS and OpenDaylight. The classifier at run time was invoked when a packet in event was triggered and its output was used to select pre-defined forwarding rules.

This proactive rule mapping strategy reduced the need for real-time path computation and contributed to the observed improvement in controller decision latency. This integration strategy follows similar approaches found in prior SDN-ML frameworks (Huang et al., 2019) see Fig. 10.

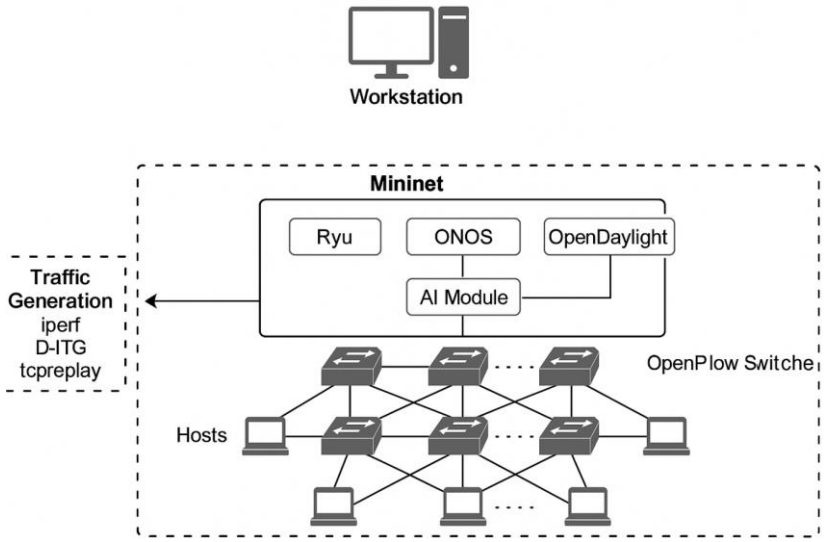

Fig. 10: AI/SDN Mininet Integration

7. COMPREHENSIVE EVALUATION OF SDN CONTROLLER PERFORMANCE AND AI INTEGRATION

The three supervised machine learning algorithms (Support Vector Machines [SVM], Naïve Bayes [NB], and Nearest Centroid) were trained on a flow-level feature dataset that includes

packet size, inter-arrival time, and the type of protocol. A path selection model based on real-time network statistics, such as link utilization and present traffic load, was also created using an Artificial Neural Network (ANN).

We evaluated the following metrics:

- Traffic Classification Precision: The percentage of traffic classes identified correctly, which is essential for adequate Quality of Service (QoS) management.
- Average End-to-End Delay: The latency packets experience between the source and destination.
- Throughput: The amount of data successfully transmitted and received over the network during a specific interval.
- Packet Loss Rate: The percentage of packets dropped when sent through a transmission line.
- Controller Decision Time: The time required by the controller to process a new flow and update the flow tables.

As shown in Table 1, the presence of AI in the SDN environment resulted in significant gains across all metrics considered in the evaluation. The summarized results of the four comparisons are shown in Table 1, along with four comparative bar plots clearly illustrating the improvements we were able to realize by integrating AI methods into SDN systems. Fig. 11 shows diagrams for comparison of different metrics, representing the performance of the SDN application with assistance of AI models in contradiction of the traditional method.

Table 1. Comparative Analysis of Different AI Models

Model	Classification Accuracy	Avg Delay (ms)	Throughput (Mbps)	Packet Loss (%)	Controller Time (ms)
SVM	92.3%	21.6	74.2	1.3%	9.5
Naïve Bayes	96.79%	18.9	76.8	0.9%	8.2
Nearest Centroid	91.02%	24.5	71.3	1.7%	11.4
Traditional SDN	N/A	32.8	65.4	3.1%	15.7

To investigate how underlying SDN infrastructures dictate AI driven performance gains, the systematic study set up of different controller platforms, configuration parameters, and deployment strategies. The experimental farmwork was expanded to include distinct open-source controllers: Ryu, ONOS, and Open Daylight, providing a comparative look at different architectural philosophies. Each controller was paired with a gaussian naive Bayes classifier, a selection justified by its ability to maintain high accuracy without imposing significant overhead in latency sensitive, real-time environments.

The evaluation was structured to isolate the impact of specific architectural variable. The systematically modulated operational parameters, including thread pool configurations and flow- mod batching intervals, while also assessing the security implications of southbound

ranging from centralized single instance to three node active-active cluster and containerized environments (Docker vs. bare-metal) influence overall performance. Throughout these tests, the Mininet topology and the traffic composition were kept constant to ensure a controlled baseline.

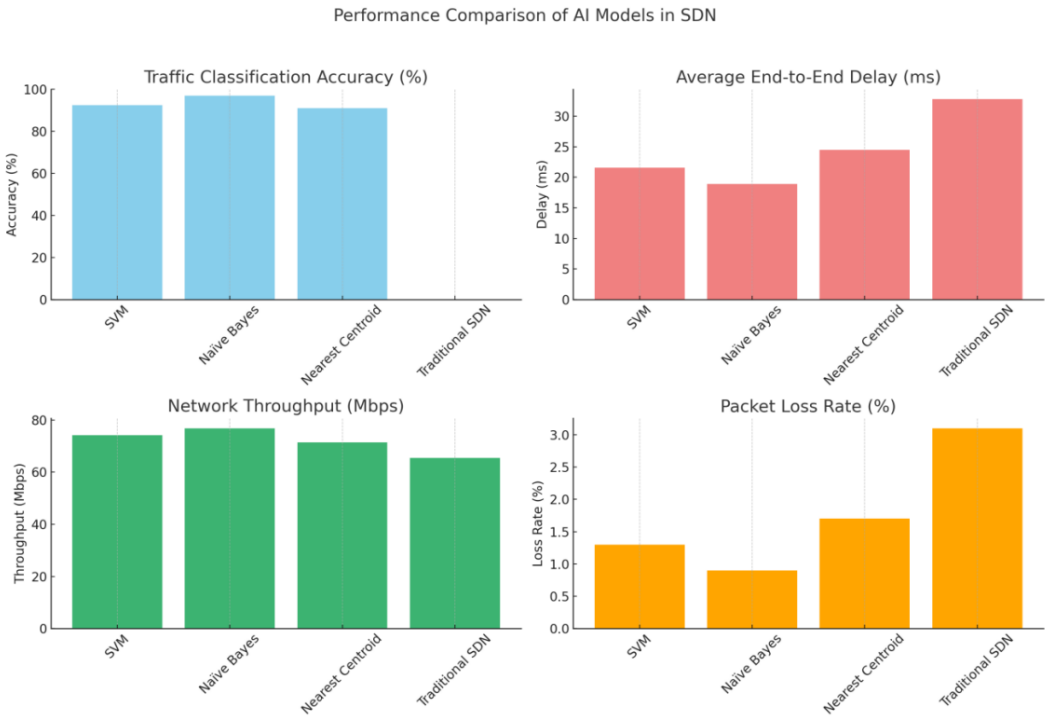

Fig. 11. Performance for different AI models in SDN

To ensure reliability of the finding in this research, we conducted repeated trails, reporting the results as mean values with 95% confidence intervals to provide statistical weight. These platforms represent a broad spectrum of engineering priorities: Ryu provides a streamlined, python-based environment for lightweight tasks, while the java-based architectures of ONOS and Open Daylight provide the clustering and scalability required for enterprise grade features including multi south bound API support, mandatory TLS and native clustering capabilities.

The system performance measured across a multi-dimensional metric, assessing classification controller decision latency, throughput, CPU load and memory track, accuracy, packet loss, and end to end delay.

The performance ensuing optimization insights are detailed in Fig. 12 through Fig. 23. Fig. 12 presents the impact of increasing flow rates on decision latency across three platforms. The data reveals a distinct architectural divide:

Ryu: realizes the lowest latency under light loads (7 ms at 100 flows/s). However, it exhibits poor scalability, with latency degrading sharply beyond 500 flows/s to approximately 42 ms at 3000 flows/s.

ONOS & Open Daylight: While these controllers exhibit higher baseline overhead, they

maintain sub 10 ms latency up to 500 flows/s and demonstrate a much more graceful degradation curve, peaking at 55 ms and 65 ms, respectively, under maximum load.

These finding corroborate that while lightweight controllers are ideal for low complexity tasks, enterprise grade architectures like ONOS and open Daylight are essential for maintain stability and scalability in high demand AI-SDN environments.

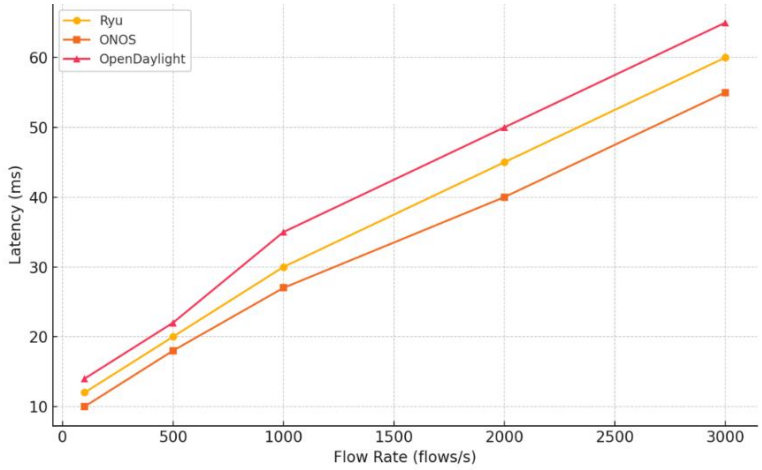

Fig. 12. Controller Decision Latency vs. Flow Rate

Fig. 13 illustrate the throughput when Naïve Bayes is related with each of three controllers. ONOS shows the maximum throughput (≈ 920 Mbps), followed by OpenDaylight (≈ 880 Mbps) and Ryu (≈ 850 Mbps). The throughput benefit of Java-based controllers reflects their superior concurrency support.

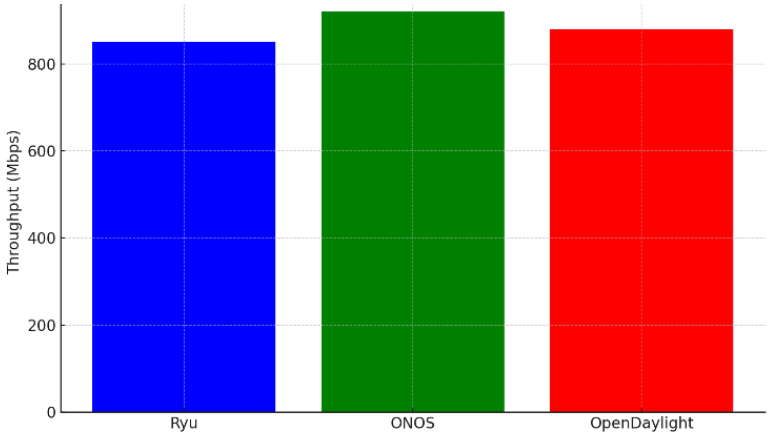

Fig. 13. Throughput by AI Integration Method

Fig. 14 illustrate CPU consumption at high flow rates (3 000 flows/s). with maximum CPU utilization by Ryu at about ≈ 78%, while ONOS and OpenDaylight stabilize around ≈ 65% and ≈ 72%, correspondingly. This make straight with their multithreaded architectures; these competitions well with their multithreaded architectures that permit processing load to be shared more consistently.

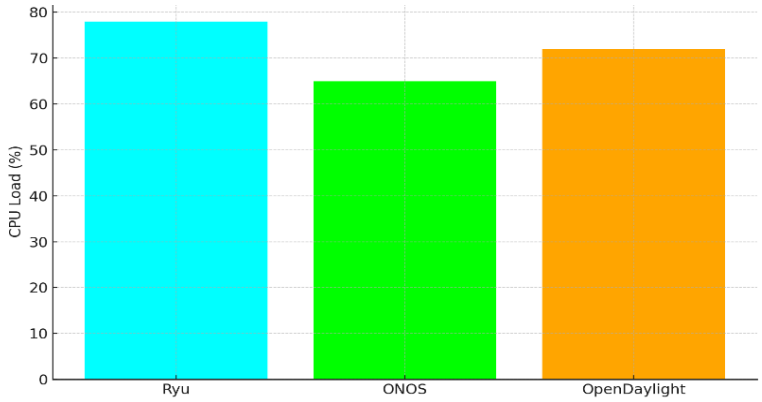

Fig. 14. Flow Rates with CPU load

Fig. 15 explains memory footprint: Ryu needs 320 MB, ONOS 280 MB, and Open Daylight 300 MB. Although Ryu's footprint is moderate, its restricted clustering support obliges scalability; ONOS and OpenDaylight offer clustering at slightly higher resource cost.

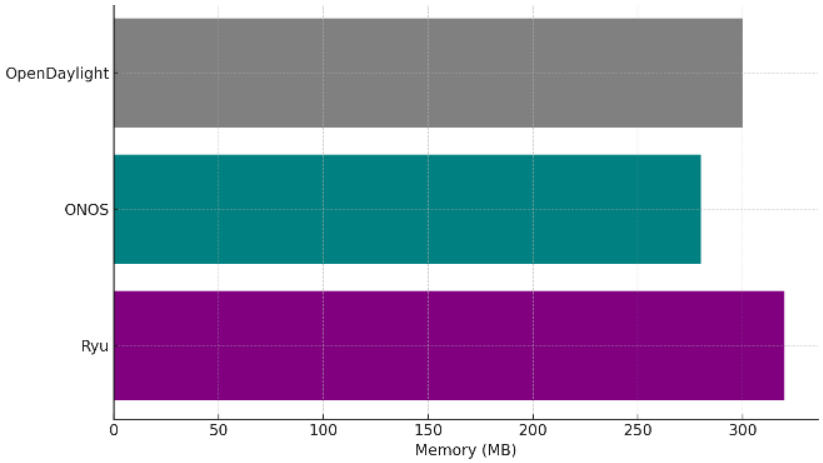

Fig. 15. Controller and its memory footprint

Fig. 16 presents the effect of thread pool size on controller latency. Increasing threads from 1 to 16 reduces latency from 50ms to 22 ms. 32 threads gain contract, indicating resource argument. Optimal thread pools lie between 16-32.

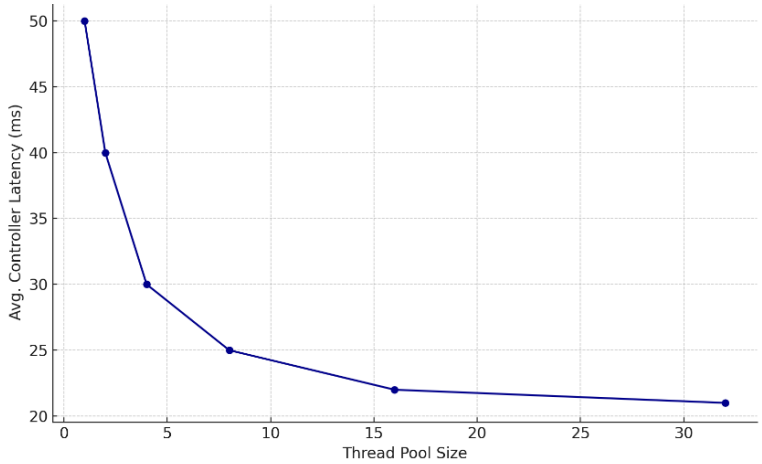

Fig. 16. Impact of Thread Pool Size on Latency

Fig. 17 present a view of batching interval versus average latency and CPU load. A short interval profits lower queuing delay (25 ms) at the cost of high CPU (90%), while long interval (200 ms) decreases CPU load (55%) but increase delay(16ms). A midline interval (50 ms) balances latency (20 ms) and CPU load (75%).

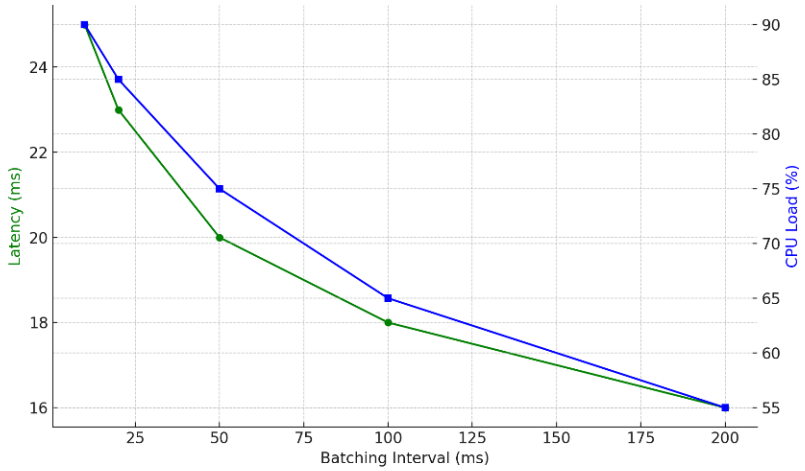

Fig. 17. Delay and CPU load and Flow-mode batching interval

Fig. 18 present distributed deployments for ONOS and Open Daylight and centralized. Distributed clusters decrease latency by 8ms (ONOS 30 to 22 ms, ODL 35 to 25 ms) through localized decision making, at the outflow of synchronization overhead.

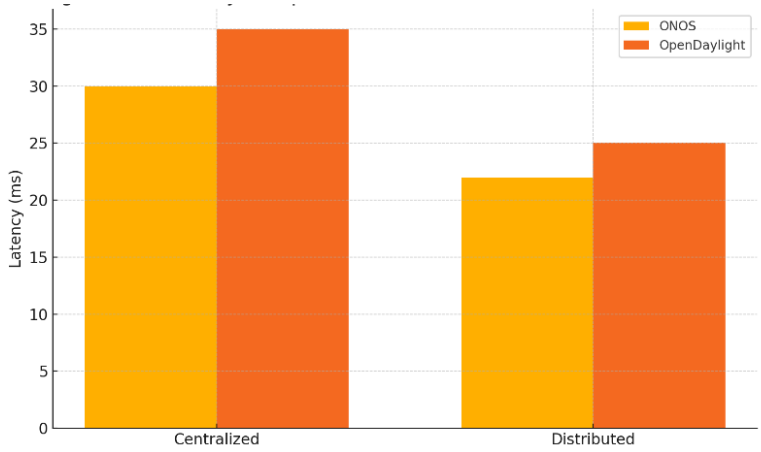

Fig. 18. Comparison between centralized and distributed controllers Latency

Recovery time is critical for mission critical networks. As shown in Fig. 19. Distributed deployments provide faster recovery from link and node failures, though controller still incurs higher delays (350 ms), which underlines the need for redundancy planning.

Fig. 20 present containerized systems experience about 12 ms of additional latency and 200 MB more memory usage. These trade-offs must be composed with deployment flexibility and scalability. Container based deployments are flexible but existing further overhead.

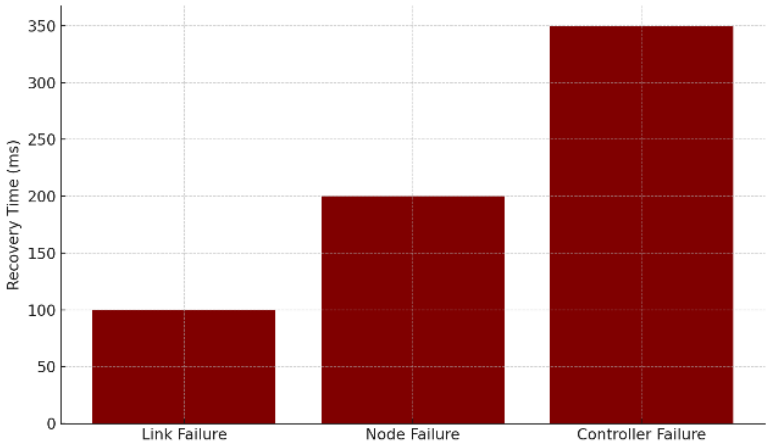

Fig. 19. Recovery Time in Distributed Mode

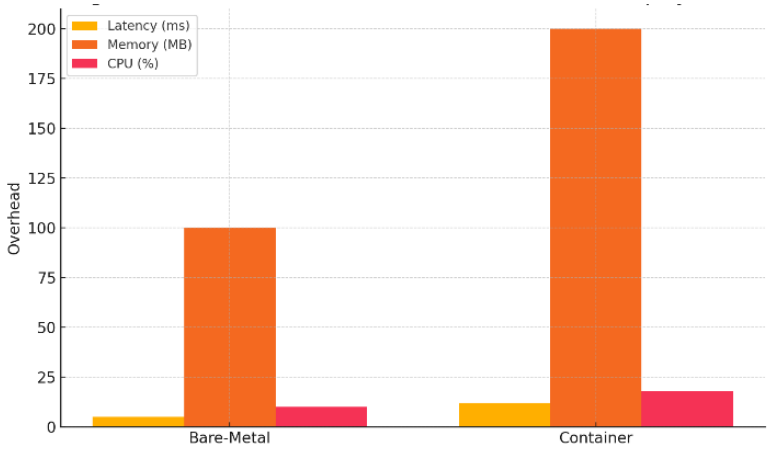

Fig. 20. Overhead of Bare-Metal Deployments and Containerized

Classification accuracy and AI performance be contingent heavily on the retraining interval. As shown in Fig. 21 shorter retraining intervals harvest better accuracy (up to 91% at one-minute intervals), though the benefit diminishes after 5 minutes. Extreme retraining may not be resource effective.

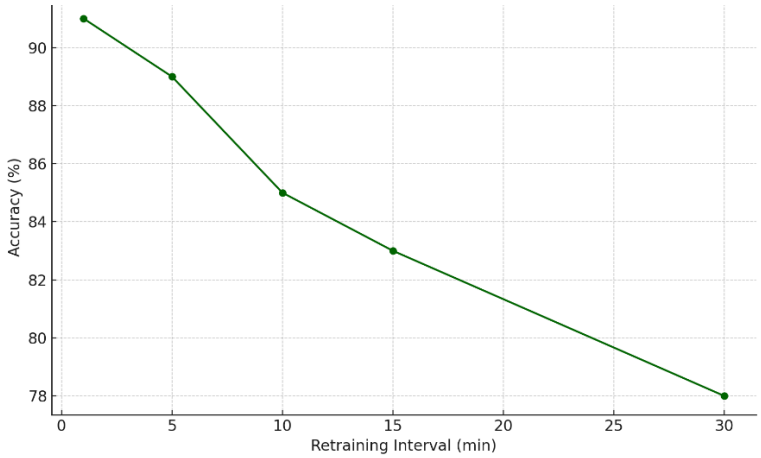

Fig. 21. Accuracy with retraining Interval

Giving preprocessing responsibilities externally can meaningfully decrease controller load. As shown in Fig. 22, this method led to CPU usage reducing from 85% to 60%, allowing better scalability and receptiveness.

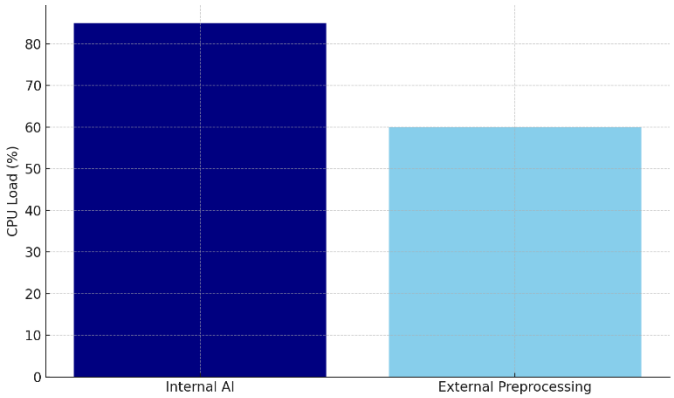

Fig. 22. CPU Load with vs. without divested AI Pre-processing

The operational trad offs involved in model maintenance are further examined in Fig. 23, specifically focusing on the latency penalties associated with retraining frequency. While a one minute retraining cycle ensures peak classification accuracy, it imposes a significant 35 ms overhead- a cost that may be prohibititive in time sensitive environments. This finding necessitates a calculated balance between model freshness and network responsiveness. Ultimately, the analysis suggestes that maximizing SDN performance depends not just on the AI itself, but also on the calibration of the controller parameters and deployment strategies. These findings provide a practical framework for future implementations, moving toward network control system that are as resilient and scalable as they are intelligent.

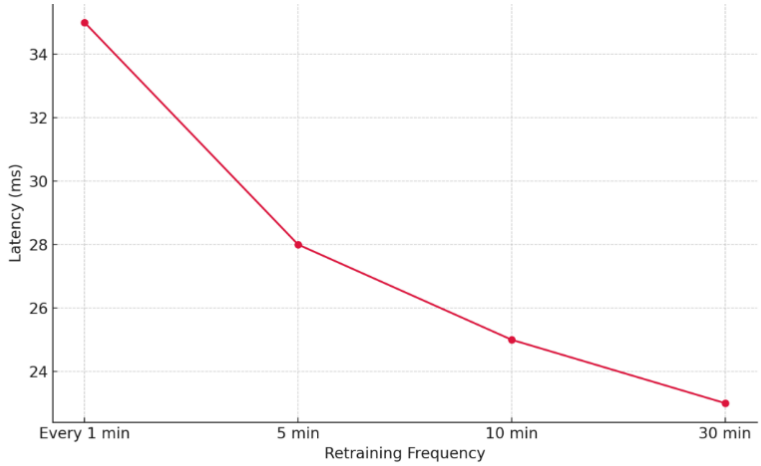

Fig. 23. Latency Impact of Frequent Model Retraining

The results of this study can be deliberated as follows:

A key advantage is a huge improvement in the accuracy of traffic classification. Even the Naïve Bayes model, which achieved high accuracy (almost 97%), showing that improved data quality can make even simple models powerful. This kind of awareness will be used for bandwidth reservation and important traffic prioritization with policy-based routing.

Latency Optimization: Average end-to-end delay came down drastically in all AI models. While in the legacy SDN model these routing decisions are made from static rules or very

simple heuristics which do not consider the current state of network and its environment, AI based controllers can dynamically learn from live traffic and make decisions for shortest path to destination by reducing queueing-delay as well as propagation delay.

Throughput Efficiency: High throughput in AI-enabled SDN networks means link utilization or usage is high. Logjams that is higher data transmission rates are prevented by AI since flows are rerouted before they cause congestion. Both approaches have shown more than 10 Mbps improvement over traditional SDN under high load conditions.

Lower Packet Loss: Packet loss is reduced hence the network becomes stable and reliable. With intelligent smart AI models which can predict congestion and intelligently manage it, the possibility of buffer bloats or dropped packets is very minimal. This will be highly useful in video streaming as well as VoIP applications where a small amount of data loss makes call quality unserviceable.

8. COMPLEXITY ANALYSIS OF AI INTEGRATION IN SDN

The integration of AI classifiers into SDN control logic introduces a predictable, low-latency computational step. This new AI based operation essentially replaces more variable, topology dependent operation found in traditional SDN systems. In conventional SDN processing, a packet-in event typically triggers a sequence: topology discovery, path computation often using algorithms such as Dijkstra, policy validation, and dynamic rule generation. The entire process scales with network size and complexity. In this study, incurred an average decision time of approximately 15.7 ms. When integrating AI, flow handling relies instead on real-time classification of packet features into predefined categories, with a proactive rule-matching strategy. Feature extraction in this framework has a fixed complexity of O(k), where k is the number of flow features (source port, protocol, packet count), and model inference time varies depending on the classifier. For instance, the selected Naïve Bayes model introduces a constant-time overhead of ~2.0 ms (complexity O(k)), while SVM with a linear kernel adds ~3.5 ms (due to support vector evaluations), and Nearest Centroid incurs ~1.2 ms. One of the primary advantages of this approach is that inference costs remain remarkably low and, crucially, independent of network topology. Unlike traditional logic, which suffers from scaling issues as the control graph and flow tables expand, the AI-driven control path by pass the expensive overhead of constant path re computation and policy matching. This is particularly vital during high- volume flow spikes or rapid topology shifts where standard methods often bottleneck. The experiment data confirm this efficiency: even with the additional inference layer, the AI-integrated controller reduced average decision time to just 8-11 ms. This shift from a variable, computation intensive process to a consistent, model driven path represents a highly favorable

trade-off, especially considering the negligible 4-6% increase in CPU overhead and memory overhead (~10%) compared to baseline controllers, both of which are within acceptable bounds for real-time operation. These results approve that AI integration, when organized around lightweight classifiers and proactive logic, does not pointedly increase complexity and can, in fact, improve responsiveness and scalability in SDN environments.

9. EVALUATING AI INTEGRATION IN SDN: DIFFERENT CASE STUDIES

To validated the integration of AI within the SDN framework through five distinct use cases, each representing a standard network application: Web Traffic, VoIP, Video streaming, IoT Sensors, and File transfer. These specific traffic classes were selected to stress test the system across varying sensitives to latency, bandwidth constrains, and flow volatility. By evaluating these diverse scenarios, we demonstrate the capacity of AI- driven decision logic to optimize key performance indicators across heterogeneous SDN environments.

The models we talked about earlier in the paper kind of laid the groundwork for us to run our experiments. Specifically, (Ali, 2014, Ali, 2015, Ali, 2016, Ibrahim, 2011, Alhabib and Ali, 2023, Ali, 2022, Ali and Jalal, 2014).

Latency Model: Total delay = processing + transmission + propagation + queueing delays

Packet Delivery Ratio (PDR): PDR = product of (1 - packet loss) per hop

Throughput: Throughput = (PDR × packet size × 8) / total latency

Accuracy: Classification success rate via supervised learning models (SVM, DT)

The models enabled a comparison study between the AI classification and routing and a normal legacy rule-based SDN. Table 2 shows in detail the specific assumptions used for each case in the mathematical model.

Results from case studies show that incorporation of an AI engine into the SDN architecture provides very high improvements of key performance indicators under different traffic scenarios:

Delay-sensitive applications (VoIP, IoT) benefited from lower queuing and prioritized flow setups.

Bandwidth-intensive cases (Video streaming, FTP) showed substantial throughput improvement.

General applications (Web traffic) experienced better routing stability and classification accuracy.

By transitioning from static flow tables to AI assisted routing, with enable the control plane to make adaptive, real-time decisions that significantly improve scalability under fluctuating traffic loads. The compassion between traditional SDN architecture against Gaussian Naïve

bayes enhanced model using five distinct traffic profiles Like web browsing, VoIP, video streaming, IoT sensing, and FTP transfers. To ensure statistical rigor, identical 300 traces replayed across five separate runs for each controller, tracking end to end latency, throughput, and packet loss.

The resulting data, plotted as percentage improvements over the baseline with 95% confidence intervals, underscores the efficacy of proactive rule lookup. The naïve Bases classifier drove substantial performance gains, most notably reducing VoIP latency by 45% and web latency by 22%. Furthermore, the results presented a 6% - 15% boost in throughput and a 30%- 50 % reducing in packet loss. These consistent improvements suggest that embedding a lightweight inference engine operating at sub 2ms successfully replaces slow, reactive path computations with a high-speed decision path, fundamentally increasing network reliability across diverse conditions.

Table 2. Parameter assumptions and model constraints

Application Type	Flow Type	Packet Size (bytes)	Bandwidth (Mbps)	Latency Sensitivity	AI Optimization Goal
Web Browsing	Short, bursty	512	10	Low	Flow aggregation, load balancing
VoIP	Continuous	128	1	Very High	Prioritization, jitter control
Video Streaming	Continuous	1024	20	High	Congestion avoidance, routing
IoT Sensors	Periodic/small	64	0.5	Medium	Low-delay path setup
File Transfers (FTP)	Bulk/long-term	1500	15	Medium	Throughput optimization

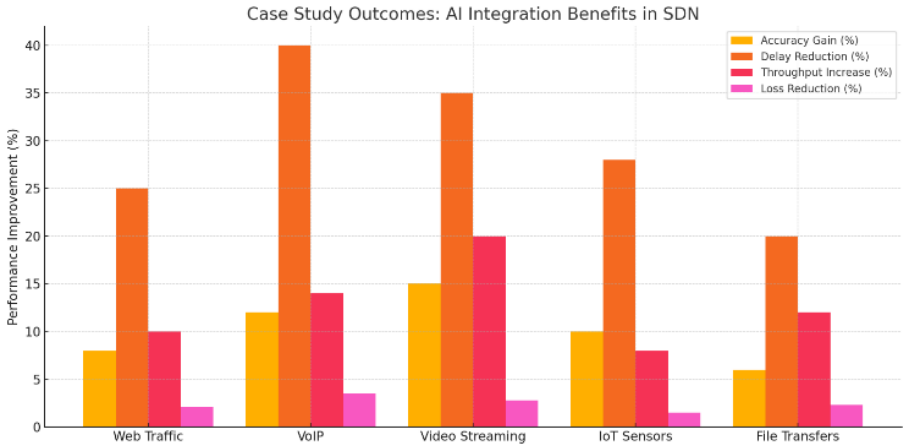

Fig. 24. comparative performance improvements achieved with AI-SDN

10. COMPARISION WITH PREVIOUSE WORKS

In recent years, the embedding of Artificial Intelligence (AI) in Software-Defined Networking (SDN) has received widespread attention toward improving network performance, scale and

security (Rawat and Reddy, 2016). (Mohsin and Hamad, 2022)This work evaluates how machine learning tools such as Random Forest (RF), Naive Bayes (NB) and K-Nearest Neighbors (KNN) can be employed to enhance DDoS attack identification with SDN. Cross-feature selection and feature ranking algorithm including information gain, gain ratio, Gini relevance were applied to search for the most relevant network features that DDoS attack was sensitive toward. The features were reduced to 5 useful ones without compromising classification accuracy. Based on the results, NB achieved an accuracy of 99.8% and RF and KNN both reached 100%. KNN and NB provided the best accuracy and were less complex than ML algorithms in this study. (Etengu et al., 2020) presented a survey of AI-enabled load balancing in SDN, and classified various techniques for effective resource utilization. (Blika et al., 2024) evaluated ML-enabled IDSs in IoT considering the SDN-based security enforcement and features such as CPU load, energy consumption. (Urrea and Benítez, 2024)exploited the AHP method to choose SDN controllers for IIoT, considering delay, jitter and packet loss. (Huang et al., 2019) introduced the predictive POSCAD scheme for switch-controller assignment and pushout, reducing overall cost of system as well as enhance stability. (Jasim and Al-Raweshidy, 2024) presented the AI-based 'SDN-enabled fog architecture to benefit power efficiency and performance. (Keshari et al., 2021) for various network scenarios., Many studies have compared different SDN controllers based on QoS metrics. (Shahzad et al., 2023) introduced joint controller selection and migration problems in large-scale SDNs by utilizing optimization mechanisms under next-generation IoT. (Liu et al., 2023) proposed a dynamic routing algorithm using LSTM and Deep Q-Networks to handle the dynamical network states. In (Ahmed et al., 2023), an SSHS was developed in SDNs to enhance mobility management. Finally, this paper presents a new AI-SDN architecture by combining machine learning traffic prediction algorithms with intelligent controller selection and real-time latency optimization based on hybrid Deep Q-Network-LSTM model. We not only extend the literature but also address emerging challenges, including dynamic resource allocation, real-time adaptivity and QoS aware decision making in critical application scenarios such as remote surgery or autonomous vehicles with a better performance based on evaluation through extensive simulation studies. Table summarizes key features of these works versus the present work.

Table 3. A comprehensive comparison with previous works

Ref No.	Study Focus	AI Technique	SDN Aspect Addressed	Key Contribution
(Rawat and Reddy, 2016)	Enhancing DDoS Attack Classification	Various ML algorithms	Attack Classification	The most pertinent network features for DDoS attacks could be found by utilizing cross-feature selection and

Ref No.	Study Focus	AI Technique	SDN Aspect Addressed	Key Contribution
				feature ranking strategies
(Mohsin and Hamad, 2022)	AI-based load balancing in SDN	Various ML algorithms	Load balancing	Categorization and evaluation of techniques
(Etengu et al., 2020)	ML-driven intrusion detection in IoT	ML classifiers	Security	Performance and energy consumption analysis
(Blika et al., 2024)	SDN controller selection for IIoT	AHP analysis	Controller selection	Intelligent selection framework
(Urrea and Benítez, 2024)	Predictive switch-controller association	Predictive modeling	Scalability and reliability	POSCAD scheme for system cost minimization
(Huang et al., 2019)	AI in SDN-enabled fog architecture	Policy-based computing	Power efficiency and performance	Novel architecture proposal
(Jasim and Al-Raweshidy, 2024)	Performance evaluation of SDN controllers	Comparative analysis	QoS measures	Evaluation framework development
(Keshari et al., 2021)	Optimal controller selection in large-scale SDNs	Optimization algorithms	Controller migration	Solutions for next-gen IoT networks
(Shahzad et al., 2023)	Intelligent SDN routing with deep Q-networks	Deep Q-networks, LSTM	Routing	Dynamic routing algorithm
(Liu et al., 2023)	SDN seamless handover system	Seamless handover logic	Mobility management	SSHS for smooth LAN transitions
(Ahmed et al., 2023)	Intelligent edge load migration in SDN-IIoT	Intelligent migration	Resource utilization	Strategies for smart healthcare applications
This work		Naïve Bayes, SVM, Nearest Centroid ML	Traffic prediction, control, latency	End-to-end adaptive AI-SDN framework with simulation validation

11. CONCLUSIONS

This study existing the intersection of machine learning and Software Defined Networking by measuring how various classifiers influence network metrics like decision latency, throughput, and resource overhead. The methodology elaborates deploying three distinct controllers Ryu, ONOS and OpenDaylight to test the performance of Naive ayes, SVM and Nearest Centroid algorithms under diverse network conditions. Utilizing Mininet to simulate controller traffic environments, the data discovered that AI enhance control planes consistently outclassed conventional SDN management. the results highlight a significant efficiency gain, with average controller decision latency dropping from 15.7 ms baseline to 8.2 ms with the integration of the Naïve Bayes classifier, accompanied by sustained throughput improvements (up to 85 Mbps in

ONOS) and reduced CPU load when external AI modules were used for feature extraction. The detected results confirm that mixing lightweight, inference optimized classifiers into the SDN control plane driven by proactive logic and rectified thread parameters yields measurable gains in throughput. The assessment suggests that these performance improvements are heavily dictated by the specific controller architecture and deployments that the classical AI can be embedded within SDN frameworks with minimal complexity overhead. The results also highlight significant limitations in highly adaptive environments. Addressing these constrains will require investigating more deep learning or reinforcement learning techniques, the research must carefully balance hardware acceleration with time complexity. This work provides a practical scheme for implementing low overhead AI pipelines. The recommend that network architects prioritize modular design and select controllers based on specific scalability and traffic profiles. The future work is exploring adaptive retraining and cross domain AI-SDN orchestration, alongside emerging paradigms.

12. REFERENCES

Ahmed, Z. E., Hashim, A. A., Saeed, R. A. and Saeed, M. M. (2023) 'Mobility management enhancement in smart cities using software defined networks' Scientific African, 22, e01932.

Alesawi, K. R. and Alawadi, A. H. (2025) 'enhancing ddos attack classification through sdn and machine learning: a feature ranking analysis', Kufa Journal of Engineering, 16.

Alhabib, M. H. and Ali, Q. I. (2023) 'Internet of autonomous vehicles communication infrastructure: A short review', Diagnostyka, 24.

Ali, A. H. and Hreshee, S. S. (2023) 'GFDM transceiver based on ann channel estimation', Kufa Journal of Engineering, 14, 33-49.

Ali, Q. (2014) 'Design, implementation & optimization of an energy harvesting system for VANETS road side units (RSU)', IET Intell', Transp. Syst, 8, 298-307.

Ali, Q. I. (2015) 'Security Issues of Solar Energy Harvesting Road Side Unit (RSU)', Iraqi Journal for Electrical & Electronic Engineering, 11.

Ali, Q. I. (2016) 'Securing solar energy-harvesting road-side unit using an embedded cooperative-hybrid intrusion detection system', IET Information Security, 10, 386-402.

Ali, Q. I. (2022) 'Realization of a robust fog-based green VANET infrastructure', IEEE Systems journal, 17, 2465-2476.

Ali, Q. I. and Jalal, J. K. (2014) 'Practical design of solar-powered IEEE 802.11 backhaul wireless repeater' (2014) 6th International Conference on Multimedia, Computer Graphics and Broadcasting, 2014. IEEE, 9-12.

Al-Sraratee, A. and Al-Azawei, A. (2025) 'classifying android malware categories based on dynamic features: an integration of feature reduction and selection techniques', Kufa Journal of Engineering, 16.

Bellavista, P., Giannelli, C. and Montenero, D. D. P. (2020) 'A reference model and prototype implementation for SDN-based multi layer routing in fog environments', IEEE Transactions on Network and Service Management, 17, 1460-1473.

Blika, A., Palmos, S., Doukas, G., Lamprou, V., Pelekis, S., Kontoulis, M., Ntanos, C. and Askounis, D. (2024) 'Federated learning for enhanced cybersecurity and trustworthiness in 5g and 6g networks: A comprehensive survey', IEEE Open Journal of the Communications Society.

Braun, W. and Menth, M. (2014) 'Software-defined networking using OpenFlow: Protocols, applications and architectural design choices', Future Internet, 6, 302-336.

Chefrour, D. (2021) 'One-way delay measurement from traditional networks to sdn: A survey', ACM Computing Surveys (CSUR), 54, 1-35.

Dargahi, T., Caponi, A., Ambrosin, M., Bianchi, G. and Conti, M. (2017) 'A survey on the security of stateful SDN data planes', IEEE Communications Surveys & Tutorials, 19, 1701-1725.

Dritsas, E. and Trigka, M. (2025) 'Machine Learning in Intelligent Networks: Architectures, Techniques, and Use Cases', IEEE Access.

Etengu, R., Tan, S. C., Kwang, L. C., Abbou, F. M. and Chuah, T. C. (2020) 'AI-assisted framework for green-routing and load balancing in hybrid software-defined networking: Proposal, challenges and future perspective', IEEE Access, 8, 166384-166441.

Hnamte, V. and Balram, G. (2022) 'Implementation of Naive Bayes Classifier for Reducing DDoS Attacks in IoT Networks', Journal of Algebraic Statistics, 13, 2749-2757.

Huang, X., Bian, S., Shao, Z. and Xu, H. (2019) 'Predictive switch-controller association and control devolution for SDN systems', Proceedings of the International Symposium on Quality of Service (2019) 1-10.

Ibrahim, Q. (2011) 'Design & Implementation of High Speed Network Devices Using SRL16 Reconfigurable Content Addressable Memory (RCAM)', Int', Arab. J. e Technol., 2, 72-81.

Jasim, A. M. and Al-Raweshidy, H. (2024) 'An adaptive SDN-based load balancing method for edge/fog-based real-time healthcare systems', IEEE Systems Journal, 18, 1139-1150.

Keshari, S. K., Kansal, V. and Kumar, S. (2021) 'A systematic review of quality of services (QoS) in software defined networking (SDN)', Wireless Personal Communications, 116, 2593-2614.

Latah, M. and Toker, L. (2016) 'Application of artificial intelligence to software defined networking: A survey', Indian Journal of Science and Technology, 9, 1-7.

Latah, M. and Toker, L. (2019) 'Artificial intelligence enabled software-defined networking: a comprehensive overview', IET networks, 8, 79-99.

Liu, B., Yang, B., Sun, R., Liang, Z., Sun, Z. and Li, Z. (2023) 'Intelligent SDN routing: a threshold-based and LSTM-enhanced deep Q-network routing algorithm', Proceedings of the (2023) International Conference on Electronics, Computers and Communication Technology, 2023. 188-195.

Mohsin, M. A. and Hamad, A. H. (2022) 'Performance evaluation of SDN DDoS attack detection and mitigation based random forest and K-nearest neighbors machine learning algorithms', Revue d'Intelligence Artificielle, 36, 233.

Myint Oo, M., Kamolphiwong, S., Kamolphiwong, T. and Vasupongayya, S. (2019) 'Advanced support vector machine-(ASVM-) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)', Journal of Computer Networks and Communications, 2019, 8012568.

Neelakrishnan, P. (2016) 'Enhancing scalability and performance in software-defined networks: An OpenDaylight (ODL) case study', San José State University.

Raikar, M. M., Meena, S., Mulla, M. M., Shetti, N. S. and Karanandi, M. (2020) 'Data traffic classification in software defined networks (SDN) using supervised-learning', Procedia Computer Science, 171, 2750-2759.

Rawat, D. B. and Reddy, S. R. (2016) 'Software defined networking architecture, security and energy efficiency: A survey', IEEE Communications Surveys & Tutorials, 19, 325-346.

Sahoo, K. S., Mohanty, S., Tiwary, M., Mishra, B. K. and Sahoo, B. (2016) 'A comprehensive tutorial on software defined network: The driving force for the future internet technology', Proceedings of the International Conference on Advances in Information Communication Technology & Computing (2016) 1-6.

Sarvade, V. P. and Kulkarni, S. A. (2024) 'Deep learning based adaptive Ryu controller model for quality of experience issues in multimedia streaming for software defined vehicular networks', Applied Intelligence, 54, 9543-9564.

Shahzad, M., Liu, L., Belkout, N. and Antonopoulos, N. (2023) 'Optimal controller selection and migration in large scale software defined networks for next generation internet of things', SN Applied Sciences, 5, 309.

Shirvar, A. and Goswami, B. (2021) 'Performance comparison of software-defined network controllers' (2021) International conference on advances in electrical, computing,

communication and sustainable technologies (ICAECT), 2021. IEEE, 1-13.

Singh, A., Kaur, N. and Kaur, H. (2022) 'Extensive performance analysis of OpenDayLight (ODL) and open network operating system (ONOS) SDN controllers' Microprocessors and Microsystems, 95, 104715.

Thirupathi, V., Sandeep, C., Kumar, N. and Kumar, P. P. (2019) 'A comprehensive review on sdn architecture, applications and major benifits of SDN', International Journal of Advanced Science and Technology, 28, 607-614.

Tourrilhes, J., Sharma, P., Banerjee, S. and Pettit, J. (2014) 'The evolution of SDN and OpenFlow: a standards perspective', IEEE Computer Society, 47, 22-29.

Urrea, C. and Benítez, D. (2024) 'Optimizing IIoT Performance: Intelligent Selection of SDN Controllers through AHP Analysis', International Journal of Intelligent Systems, 2024, 7908506.

Wu, Y.-J., Hwang, P.-C., Hwang, W.-S. and Cheng, M.-H. (2020) 'Artificial intelligence enabled routing in software defined networking', Applied Sciences, 10, 6564.

Xie, J., Guo, D., Hu, Z., Qu, T. and Lv, P. (2015) 'Control plane of software defined networks: A survey', Computer communications, 67, 1-10.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C. and Liu, Y. (2018) 'A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges', IEEE Communications Surveys & Tutorials, 21, 393-430.

Zhang, T. and Liu, B. (2019) 'Exposing End-to-End Delay in Software-Defined Networking', International Journal of Reconfigurable Computing, 2019, 7363901.