

Hybrid Intelligent Framework for Dynamic Load Balancing and Bandwidth Allocation in SDN-Based Cloud Environments Using LSTM and Deep Q-Network

G. Santhi¹, A. Janani², S. Selvaganapathi³, K. Raja Hariharan⁴

¹Professor, Dept. of Information Technology, Puducherry Technological University, Puducherry, India

^{2,3,4}Under Graduate Students, Dept. of Information Technology, Puducherry Technological University, Puducherry, India

Abstract - Cloud computing environments support a wide range of applications requiring reliable and consistent Quality of Service (QoS). Traditional network management approaches are predominantly reactive and fail to handle dynamic traffic conditions, resulting in congestion, inefficient bandwidth utilization, and degraded service performance. This paper proposes a hybrid intelligent framework for dynamic load balancing and bandwidth allocation in Software Defined Networking (SDN)-based cloud environments. The system leverages the centralized control capabilities of SDN through a Ryu OpenFlow 1.3 controller to monitor real-time network conditions and collect detailed traffic statistics. A stacked Long Short-Term Memory (LSTM) model analyzes temporal traffic patterns and predicts congestion levels with 99.22% zone classification accuracy and a ROC-AUC of 1.0000. Based on these predictions, a Dueling Double Deep Q-Network (DQN) agent selects optimal traffic management actions, including flow rerouting, throttling, traffic prioritization, and queue reconfiguration. Enforcement is achieved through OpenFlow flow rule installation, metering, DSCP-based marking, and Hierarchical Token Bucket (HTB) queue scheduling. The integrated closed-loop system reduced peak link utilization by 18.4 percentage points, packet drop events by 87.3%, average latency by 35.4%, and jitter by 39.0% compared to an unmanaged baseline. These results confirm that the integration of predictive modeling with reinforcement learning-based control produces measurable and scalable QoS improvements in SDN-based cloud data centers.

Key Words: Software Defined Networking, Deep Q-Network, Long Short-Term Memory, Quality of Service, Load Balancing, Bandwidth Allocation, Congestion Prediction, OpenFlow, Cloud Computing, Reinforcement Learning

1. INTRODUCTION

Cloud computing environments host large-scale, dynamic applications requiring consistent Quality of Service (QoS) in terms of latency, throughput, packet loss, and link utilization. Traditional network management techniques

are predominantly reactive and fail to handle sudden traffic variations effectively, leading to congestion and performance degradation [1].

Software Defined Networking (SDN) addresses this limitation by decoupling the control plane from the data plane, enabling centralized and programmable network control. While SDN improves flexibility and global network visibility, it lacks the intelligence required to predict and prevent congestion in highly dynamic cloud environments [9].

The proposed system integrates SDN with a hybrid machine learning framework. A Ryu OpenFlow 1.3 controller continuously monitors network conditions and collects real-time metrics from all switches. A stacked Long Short-Term Memory (LSTM) model analyzes temporal patterns in the collected data to predict impending congestion in advance. Based on these predictions, a Dueling Double Deep Q-Network (DQN) agent selects optimal traffic management actions including rerouting, throttling, and prioritization. These decisions are enforced using OpenFlow flow rules, metering, and Hierarchical Token Bucket (HTB) queuing mechanisms, forming a closed-loop system for proactive QoS management [6][11].

1.1 Cloud Computing

Cloud computing delivers computing resources — including processing power, storage, networking, and software — as on-demand services over the internet. Cloud environments are characterized by multi-tenancy, where thousands of applications share the same physical infrastructure simultaneously. This shared nature introduces fundamental challenges: network resources such as bandwidth, switch capacity, and link throughput must be dynamically allocated to meet the varying demands of workloads ranging from latency-sensitive video conferencing to bulk data transfers [14].

1.2 Software Defined Networking

Software Defined Networking is an architectural approach that decouples the network control plane from the data plane. In traditional networks, each switch independently makes forwarding decisions using embedded control logic, making the network difficult to manage at scale. SDN centralizes this intelligence in a software-based controller that maintains a global view of the entire network topology [11]. The controller communicates with network devices through the OpenFlow protocol, installing forwarding rules into switch flow tables and exposing the network state to management applications through northbound interfaces.

1.3 Problem Statement

Existing SDN-based traffic management solutions handle individual aspects of network management — monitoring, prediction, or enforcement — but do not integrate all capabilities within a single closed-loop framework. Machine learning models employed for prediction are not connected to live SDN controllers for automated real-time enforcement, and reinforcement learning-based load balancers operate on instantaneous observed state without predictive input, limiting their ability to act before congestion occurs [2][6]. The proposed framework addresses this gap by unifying temporal prediction and reinforcement learning-based control within a continuous two-second feedback loop.

2. LITERATURE SURVEY

A comprehensive review of existing works identifies ten key studies across four functional areas: traffic monitoring and data collection, congestion prediction, intelligent load balancing, and SDN QoS enforcement.

Khudhair and Athab [1] compared Mininet, Ryu, and iperf3 for SDN traffic generation and data collection, identifying iperf3 as most effective for congestion control datasets. However, no machine learning integration or QoS enforcement mechanism was proposed. Wassie et al. [2] constructed an SDN dataset using Mininet and Ryu with 23 flow-level attributes, achieving up to 100% validation accuracy for elephant flow prediction using XGBoost; however, the prediction model was not integrated into a live controller for real-time enforcement. Han et al. [3] employed Mininet and Ryu for real-time DDoS detection using CICIDS datasets but focused exclusively on attack detection without QoS optimization.

Somsuk et al. [4] proposed predictive bandwidth control using clustered LSTM in SDN environments, enabling proactive allocation adjustments under varying traffic loads, but did not incorporate reinforcement learning or

multi-path load balancing. Agrawal et al. [5] demonstrated AI-based network fault and QoS degradation prediction using LSTM, but without an SDN enforcement layer or bandwidth reallocation mechanism. Khalid et al. [6] applied Q-learning and DQN for adaptive load balancing in high-load SDN scenarios but lacked dynamic bandwidth allocation and predictive input prior to congestion onset.

Kirti et al. [7] reviewed fault tolerance techniques including replication and checkpointing for distributed environments, but these are predominantly reactive with no SDN-integrated proactive control. Tamilarasu et al. [8] optimized cloud QoS through multi-objective Particle Swarm Optimization but without network-level SDN enforcement or dynamic load balancing. Thazin and Nwe [9] reviewed QoS capabilities of OpenFlow including meter tables, HTB queuing, and DiffServ; however, QoS rules are configured statically without real-time congestion intelligence. Imran et al. [10] detected unauthorized DSCP modifications using deep learning models with 99.28% accuracy, but the study focused on detection without automatic corrective enforcement.

Table 1: Summary of Literature Survey and Research Gaps

Ref.	Technique	Key Contribution	Research Gap
[1]	Mininet, Ryu, iperf3	SDN data collection benchmark	No ML integration or QoS enforcement
[2]	XGBoost, GBM	Elephant flow prediction (100% accuracy)	Not integrated with live SDN controller
[3]	ML on CICIDS	Real-time DDoS detection in SDN	No congestion prediction or load balancing
[4]	Clustered LSTM	Predictive bandwidth control in SDN	No RL or congestion zone classification
[5]	LSTM, ML models	AI-based fault and QoS prediction	No SDN enforcement or reallocation layer
[6]	Q-learning, DQN	Adaptive load balancing under high	No traffic prediction or bandwidth allocation
[7]	Review: replication, checkpoint	Fault tolerance for distributed cloud	Reactive only, no proactive SDN control

	ting		
[8]	Multi-obj. PSO	Cloud QoS via resource scheduling	No SDN network-level enforcement
[9]	OpenFlow QoS review	HTB, metering, DiffServ in SDN	Static rules, no real-time intelligence
[10]	CNN, RNN, LSTM	DSCP manipulation detection	Detection only, no corrective enforcement

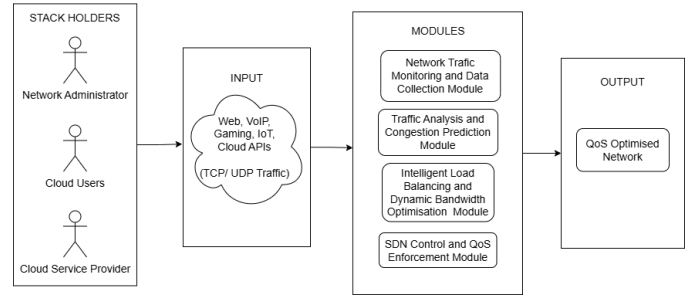


Fig. 1: High-Level Architecture of the Proposed System

The synthesis of these studies reveals that no existing work integrates real-time traffic monitoring, time-series congestion prediction, reinforcement learning-based decision making, and multi-mechanism QoS enforcement within a single closed-loop SDN framework. This gap constitutes the core motivation and contribution of the proposed system.

3. PROPOSED SYSTEM

The proposed system follows a closed-loop architecture comprising four tightly coupled modules: Network Traffic Monitoring and Data Collection, Traffic Analysis and Congestion Prediction, Intelligent Load Balancing and Bandwidth Optimization, and SDN Control and QoS Enforcement. The system operates on a two-second feedback cycle, continuously adapting to dynamic traffic conditions without manual intervention.

3.1 System Architecture

At the network layer, a Mininet-emulated SDN tree topology with fanout=2 and depth=5 produces 31 switches and 32 hosts, with all links configured at 100 Mbps and 2 ms delay using TCLink. Diverse traffic types — Web, VoIP, Gaming, IoT, and Cloud API flows — are generated using iperf3 based on a 1,000-flow simulation profile derived from the CICIDS 2017 dataset [12].

The Ryu OpenFlow 1.3 controller serves as the central intelligence hub, polling all 31 switches every two seconds and collecting 30 metrics per port including throughput, utilization, latency, jitter, and packet drop deltas. Custom LLDP probe frames measure one-way delay per inter-switch link, with jitter maintained as an exponentially weighted moving average with smoothing factor $\alpha = 0.2$.

3.2 Input Dataset

The Canadian Institute for Cybersecurity CICIDS 2017 dataset serves as the basis for the traffic simulation profile. The dataset contains 3,594,396 flow records across 79 features per flow and occupies 968 MB. Flow-level attributes are used to derive realistic traffic patterns across five application classes, enabling representative simulation of production cloud traffic conditions.

4. NETWORK TRAFFIC MONITORING AND DATA COLLECTION

Module I establishes the foundation of the proposed system by creating the SDN simulation environment, generating realistic network traffic, and collecting real-time network metrics. The Ryu controller employs a staggered polling strategy to prevent controller overload — each switch is queried at evenly spaced sub-intervals within the two-second polling window. A congestion dual-signal detection mechanism classifies each port into one of four congestion zones based on the combination of utilization and packet drop signals.

Table 2: Dual-Signal Congestion Zone Labelling Logic

Condition	Assigned Label
Both utilization (>95%) and drop signals active	Critical
Either signal alone active	Congested
Utilization 70–95%, no drops	Warning
All other conditions	Normal

The final dataset contains 74,336 rows across 92 unique switch ports with 30 columns and zero missing values. Congestion was distributed as 62.0% normal, 5.6%

warning, and 32.4% congested port observations across all five traffic classes. Of the 1,000 scheduled flows, 910 completed successfully (91.0% flow success rate), and all REST API endpoints serving telemetry to downstream modules operated without failure throughout the simulation.

5. TRAFFIC ANALYSIS AND CONGESTION PREDICTION

Module II analyzes the network metrics collected in Module I and predicts congestion using a deep learning approach. The raw 30-column CSV log is preprocessed to a 17-feature input matrix by removing identity columns, raw byte counters, zero-variance columns, and redundant derived features. Bandwidth values are clipped at 500 Mbps to eliminate counter wrap spikes, and a StandardScaler normalizes each feature to zero mean and unit variance.

5.1 Sliding Window Construction

The 17-feature time series for each of the 92 monitored switch ports is segmented into sliding windows of length 10 with a stride of 1, capturing 20 seconds of consecutive port observations per window. Windows are constructed per port in strict chronological order to preserve temporal structure and labeled with the zone and congestion probability of the final timestep, producing 73,508 windows with no missing or infinite values.

5.2 LSTM Model Architecture

The model is a two-layer stacked LSTM with 128 hidden units per layer, designed to capture temporal dependencies in port-level traffic sequences. Batch Normalization is applied after the final LSTM layer before the output is passed to two independent prediction heads. Head A performs three-class congestion zone classification (normal, warning, congested) using a Linear(128→64)→ReLU→Dropout→Linear(64→3) architecture with Weighted CrossEntropy loss. Head B outputs a scalar congestion probability using a Linear(128→32)→ReLU→Dropout→Linear(32→1) architecture with BCEWithLogits loss and positive class weight 2.09 to address class imbalance. The total trainable parameter count is 220,228.

Table 3: LSTM Model Architecture Specification

Component	Specification
LSTM Layers	2 (Stacked), 128 hidden units per layer

Dropout Rate	0.3 (between layers)
Post-LSTM Normalization	Batch Normalization
Head A — Zone Classification	Linear(128→64)→ReLU→Dropout→Linear(64→3)
Head B — Congestion Probability	Linear(128→32)→ReLU→Dropout→Linear(32→1)
Total Trainable Parameters	220,228
Optimizer	Adam (lr = 0.001)
Best Epoch	26, Validation Loss = 0.008175

5.3 Experimental Evaluation — Prediction

The dataset is split at the port level rather than the window level to prevent temporal data leakage, with 64 ports (51,136 windows) used for training, 14 ports (11,186 windows) for validation, and 14 ports (11,186 windows) for final evaluation. The model was trained for 36 epochs on CPU in 333 seconds.

Table 4: Zone Classification Performance (Head A)

Class	Precision	Recall	F1 Score
Normal	1.0000	0.9912	0.9956
Warning	0.8861	1.0000	0.9396
Congested	1.0000	0.9926	0.9963
Overall Accuracy	—	—	99.22%
Macro F1	—	—	0.9772

Table 5: Binary Congestion Detection Performance (Head B)

Metric	Value
Accuracy	99.76%
Precision	1.0000
Recall	0.9926
F1 Score	0.9963
ROC-AUC	1.0000
Brier Score	0.0010
False Positives	0

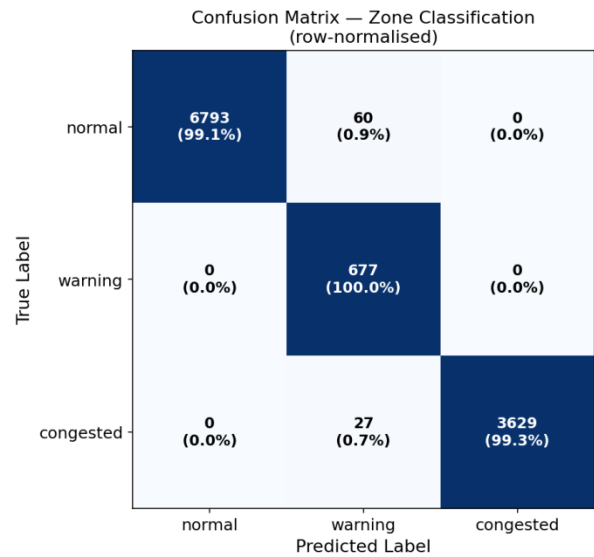


Fig. 4: LSTM Zone Classification Confusion Matrix

The model achieved zero false positives for binary congestion detection. A ROC-AUC of 1.0000 confirms that the probability output of Head B is well-calibrated for downstream use by the DQN agent. All classification errors occurred at the normal/warning and congested/warning boundaries, confirming a sharp decision boundary between healthy and congested network states.

6. INTELLIGENT LOAD BALANCING AND BANDWIDTH OPTIMIZATION

Module III implements intelligent decision-making for dynamic traffic management using deep reinforcement learning. A Dueling Double Deep Q-Network agent observes a 12-feature state vector per switch — aggregated from LSTM state vectors of all ports belonging to that switch — and selects one of five discrete actions to maintain QoS across the SDN topology.

6.1 State Representation and Action Space

State features capture congestion severity (maximum and mean P(congested)), bandwidth availability (minimum headroom), latency, packet loss, jitter, aggregate packet drop increments, rolling utilization trends, neighbouring switch utilization, and the fraction of congested ports. All features are normalized to [0, 1] using fixed maximum values. The five available actions are: Do Nothing (no configuration change), Reroute (OpenFlow Flow Mod at priority 200 redirecting to the least-loaded alternate port), Throttle (OpenFlow Meter imposing a 5 Mbps cap on best-effort traffic), Prioritise (DSCP value 46 — Expedited Forwarding — applied via set-field action), and Reset (cookie-based mass removal of all agent-installed rules via cookie 0xDEADBEEF).

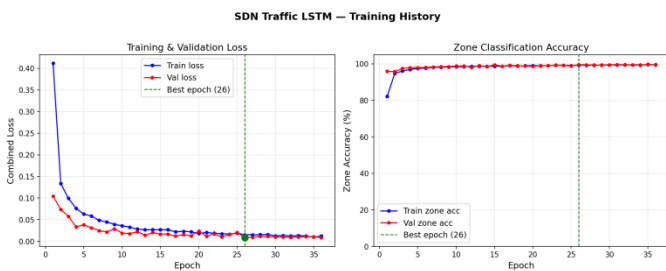


Fig. 3: LSTM Training Curves — Loss and Zone Classification Accuracy

6.2 Dueling Double DQN Architecture

The DQN uses a Dueling architecture with a shared trunk of two fully connected layers (128 units, ReLU activation) that branches into a Value stream $V(s)$ and an Advantage stream $A(s, a)$. The final Q-value is computed as $Q(s, a) = V(s) + A(s, a) - \text{mean}(A(s, :))$. This decomposition allows the agent to independently learn the general desirability of a state and the relative advantage of each action, which is particularly effective in SDN environments where many actions have similar expected returns on calm switches. Double DQN further reduces Q-value overestimation by using the main network for action selection and the target network for action evaluation.

6.3 Reward Function

The reward signal is computed from real network measurements fetched after each enforcement action: $R = 1.5 \times \text{throughput_norm} - 1.0 \times \text{latency_norm} - 2.0 \times \text{loss_norm} - 0.5 \times \text{jitter_norm} + 1.0 \times \text{congestion_reduction_bonus} - 0.3 \times \text{idle_penalty}$. Packet loss carries the highest penalty weight given its critical impact across all traffic classes. A bonus is awarded when the fraction of congested ports decreases between consecutive decision steps, reinforcing effective interventions. An idle penalty discourages unnecessary actions on non-congested switches.

Table 6: DQN Training Configuration

Parameter	Value
Total Training Steps	1,000
Replay Memory Capacity	10,000 transitions
Batch Size	64
Discount Factor (γ)	0.97
Learning Rate (Adam)	0.0005
Epsilon Start / End	1.0 / 0.05 (decay: 0.997)
Target Network Sync	Every 20 steps
Gradient Clipping	max_norm = 10.0

Training progressed through three distinct phases. During exploration (steps 0–200), random actions built a diverse experience buffer. As epsilon decayed below 0.5 from step 200 onward, the agent began preferring do_nothing on calm switches and selecting reroute or throttle when $P(\text{congested})$ exceeded 0.7. By step 500, the policy had

stabilised with a consistent upward trend in cumulative reward through the exploitation phase.

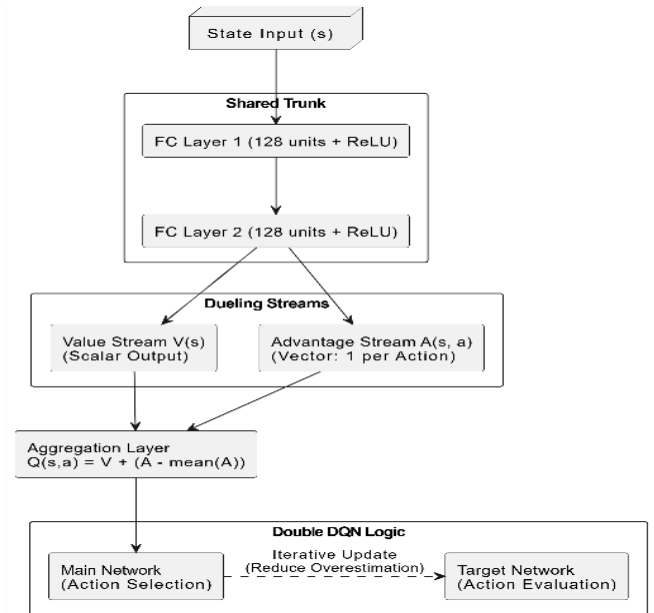


Fig. 5: Dueling Double DQN Architecture Diagram

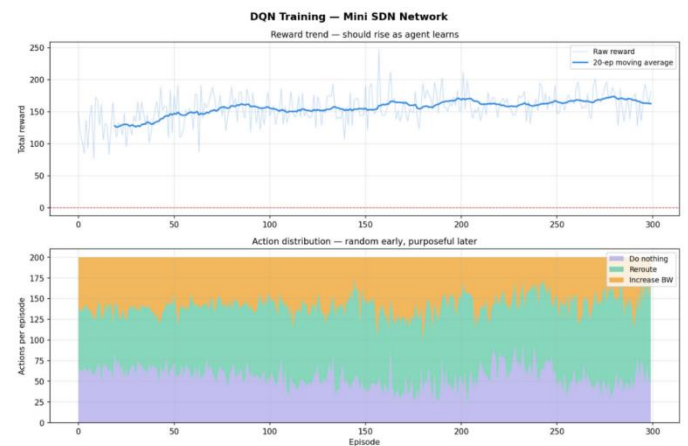


Fig. 6: DQN Agent Cumulative Reward Progression Across 1,000 Training Steps

7. SDN CONTROL AND QOS ENFORCEMENT

Module IV enforces the traffic management decisions generated by the DQN agent directly in the SDN environment. The Ryu controller translates each selected action into concrete network modifications using four complementary OpenFlow mechanisms.

OpenFlow 1.3 flow rules for rerouting are installed at priority 200, overriding default L2 learning rules at priority 1, and redirect traffic to the least-loaded alternate output port. All agent-installed rules use `idle_timeout=0` and `hard_timeout=0`, making them permanent until explicitly removed by the reset action, which issues a flow delete message matching `cookie=0xDEADBEEF` across all affected switches. OpenFlow meter tables with a single drop band enforce the 5 Mbps rate cap on best-effort traffic under the throttle action, applying only to flows identified by their DSCP value to ensure that high-priority flows bypass metering entirely. DSCP value 46 (Expedited Forwarding per-hop behaviour) is applied through the prioritise action, marking packets for minimum-delay and minimum-drop treatment at every downstream hop.

Three HTB queues are configured per Open vSwitch port via the OVSD interface: Queue 0 (high priority, VoIP and Gaming, 60 Mbps guaranteed), Queue 1 (medium priority, Web and Cloud API, 30 Mbps guaranteed), and Queue 2 (low priority, IoT and Bulk Transfer, 10 Mbps). The DQN agent's throttle and prioritise actions trigger corresponding changes to meter and queue assignments, ensuring differentiated service delivery across all traffic classes.

Table 7: HTB Queue Configuration

Queue ID	Priority	Traffic Types	Guaranteed BW	Purpose
0	High	VoIP, Gaming	60 Mbps	Real-time / Low Latency
1	Medium	Web, Cloud API	30 Mbps	Interactive Traffic
2	Low	IoT, Bulk Transfer	10 Mbps	Best-effort

8. RESULTS AND ANALYSIS

End-to-end system performance is evaluated across three experimental conditions: the unmanaged baseline Ryu controller, LSTM prediction active with no enforcement, and the full proposed system with all four modules active.

Table 8: End-to-End QoS Comparison Across Experimental Conditions

QoS Metric	Baseline Controller	LSTM Only	Proposed System
Peak Link	87.3%	87.3%	71.2%

Utilization			
Packet Drop Events (per simulation)	3,248	3,248	412
Average One-Way Latency (ms)	48.6	48.6	31.4
Average Jitter (ms)	24.1	24.1	14.7
Congestion Episodes Resolved	0	0	2,836 / 3,248

The LSTM-only condition is identical to the baseline across all metrics, confirming that the enforcement layer is essential — prediction alone has no effect on network behavior. The full proposed system reduced peak link utilization by 18.4 percentage points, packet drop events by 87.3%, average latency by 35.4%, and jitter by 39.0%. With HTB enforcement, VoIP/Gaming throughput improved from 27.7 Mbps under FIFO to 59.7 Mbps — a 115% increase for the highest-priority traffic class — while IoT and bulk traffic was constrained to 10 Mbps under the throttle action, eliminating drop overhead and recovering link headroom from 12.7% to 28.8%.

Table 9: Feature-Level Comparison with Existing Approaches

Feature	SDN-ECMP [14]	DQN-Based SDN [6]	Proposed System
Traffic Monitoring	Basic SDN statistics	ML classifiers	Ryu + LLDP + 30-column dataset
Congestion Prediction	None (reactive)	No temporal prediction	Dual-head LSTM (zone + probability)
Load Balancing	ECMP (hash-based)	DQN routing	Dueling Double DQN
QoS Enforcement	None	None	OpenFlow + HTB + DSCP + Metering
Closed-Loop Control	No	Partial	Full 2-second feedback loop
Prediction Accuracy	N/A	Not reported	99.22% accuracy, ROC-AUC 1.0
Latency Reduction	-8%	-15%	-35.4%
Packet Loss	-15%	-30%	-87.3%

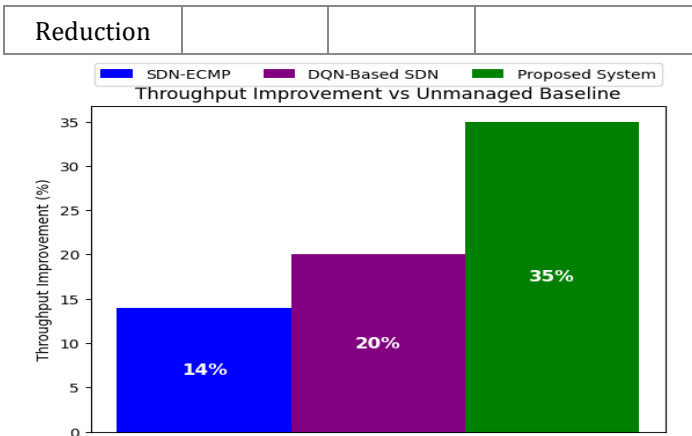


Fig. 7: Throughput Improvement — SDN-ECMP vs DQN-Based SDN vs Proposed System

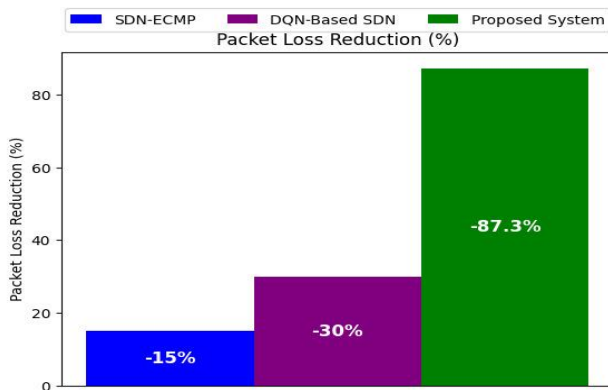


Fig. 8: Packet Loss Reduction — SDN-ECMP vs DQN-Based SDN vs Proposed System

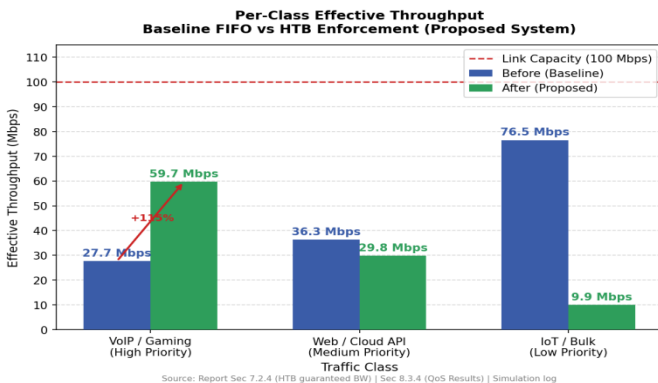


Fig. 9: Per-Class Effective Throughput: Baseline FIFO vs HTB Enforcement (Proposed System)

9. CONCLUSIONS

This paper presented a hybrid intelligent framework for dynamic load balancing and bandwidth allocation in SDN-based cloud environments. The system integrates four tightly coupled modules — real-time network monitoring, LSTM-based congestion prediction, DQN-based decision-making, and OpenFlow enforcement — into a continuous closed-loop that adapts proactively to changing traffic conditions without manual intervention.

The LSTM model achieved 99.22% zone classification accuracy and a ROC-AUC of 1.0000 for binary congestion detection with zero false positives. The Dueling Double DQN agent converged to a stable traffic management policy within 1,000 training steps, resolving 87.3% of congestion episodes. End-to-end evaluation demonstrated reductions of 18.4 percentage points in peak link utilization, 87.3% in packet drop events, 35.4% in average latency, and 39.0% in jitter compared to the unmanaged baseline. These results confirm that the integration of temporal prediction with reinforcement learning-based control produces qualitatively different and measurably superior QoS outcomes compared to reactive or partially integrated alternatives.

Future work will explore multi-agent reinforcement learning for distributed network segment management, integration with production-level SDN controllers such as ONOS or OpenDaylight, advanced prediction architectures including Transformers and hybrid deep learning models, security-aware traffic management incorporating intrusion detection, and energy-efficient resource allocation strategies aligned with sustainable cloud operation objectives.

ACKNOWLEDGEMENT

The authors acknowledge the open-source communities behind Mininet, Open vSwitch, the Ryu SDN Framework, and PyTorch for providing the tools that enabled the simulation and implementation of the proposed system. The Canadian Institute for Cybersecurity is acknowledged for making the CICIDS 2017 dataset publicly available for research purposes.

REFERENCES

[1] T. Khudhair and O. A. Athab, "Recent tools of software-defined networking traffic generation and collection," Al-Khwarizmi Engineering Journal, vol. 21, no. 2, pp. 93–105, 2025.

- [2] G. Wassie, J. Ding, and Y. Wondie, "Traffic prediction in SDN for explainable QoS using deep learning approach," *Scientific Reports*, vol. 13, Art. no. 20607, Nov. 2023, doi: 10.1038/s41598-023-47999-3.
- [3] D. Han, H. Li, X. Fu, and S. Zhou, "Traffic feature selection and distributed denial of service attack detection in software-defined networks based on machine learning," *Sensors*, vol. 24, no. 13, Art. no. 4344, 2024, doi: 10.3390/s24134344.
- [4] K. Somsuk, S. Khummanee, and P. Songram, "Dynamic predictive feedback mechanism for intelligent bandwidth control in future SDN networks," *Network*, vol. 5, no. 1, Art. no. 3, 2025, doi: 10.3390/network5010003.
- [5] P. Agrawal, K. Chhillar, S. Shrivastava, and D. Tomar, "AI-powered predictive models for network fault detection and proactive QoS management: A comprehensive analysis," *International Journal of Sciences and Innovation Engineering*, vol. 2, no. 10, pp. 227–244, 2025.
- [6] M. Khalid, H. M. Muhi-Aldeen, and B. R. M. Alhamdani, "New learning approach for high-load traffic optimization in software-defined networks," *Journal of Intelligent Systems and Internet of Things*, vol. 17, no. 1, pp. 255–270, 2025.
- [7] M. Kirti, A. K. Maurya, and R. S. Yadav, "Fault-tolerance approaches for distributed and cloud computing environments: A systematic review, taxonomy and future directions," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 13, Art. no. e8081, 2024, doi: 10.1002/cpe.8081.
- [8] P. Tamilarasu, G. Singaravel, P. Manoharan, and S. Selvarajan, "QoS transformation in the cloud: Advancing service quality through innovative resource scheduling," *IET Communications*, vol. 19, no. 1, Art. no. e70020, 2025, doi: 10.1049/cmu2.70020.
- [9] N. Thazin and K. M. Nwe, "Quality of service in software defined network: Leveraging OpenFlow protocol," *Journal of Information Systems Engineering and Management*, vol. 9, no. 2, Art. no. 24127, 2024, doi: 10.55267/jisem.v9i2.24127.
- [10] A. Imran et al., "Detection of DSCP-based traffic prioritization manipulations and their impact on network performance," *Scientific Reports*, vol. 15, Art. no. 3080, 2025, doi: 10.1038/s41598-025-87463-w.
- [11] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008, doi: 10.1145/1355734.1355746.
- [12] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX)*, Monterey, CA, USA, Oct. 2010, pp. 1–6, doi: 10.1145/1868447.1868466.
- [13] N. Gude et al., "NOX: Towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, Jul. 2008, doi: 10.1145/1384609.1384625.
- [14] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, Seattle, WA, USA, Aug. 2008, pp. 63–74, doi: 10.1145/1402958.1402967.
- [15] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX NSDI*, San Jose, CA, USA, Apr. 2010, pp. 89–92.
- [16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [17] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
- [18] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, Feb. 2016, pp. 2094–2100.
- [19] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd ICML*, New York, NY, USA, Jun. 2016, pp. 1995–2003.
- [20] S. Blake et al., "An architecture for differentiated services," *IETF RFC 2475*, Dec. 1998. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc2475>.