Journal of Engineering Science and Technology
ICIST2024, Vol. 20, No. 5 (2025) 89 - 105
© School of Engineering, Taylor's University

ENHANCE DYNAMIC QOS THROUGH DEEP LEARNING: ADAPTIVE BANDWIDTH MANAGEMENT FOR NETWORK OPTIMIZATION

MUHAMMAD FENDI OSMAN[1], MOHD RIZAL BIN MOHD ISA[1,*], MOHAMMAD ADIB KHAIRUDDIN[1], MOHD 'AFIZI MOHD SHUKRAN[1], NOOR AFIZA MAT RAZALI[1], NUR DIYANA BINTI KAMARUDIN[1], KAMARUZAMAN MASKAT[1], ROZIYANI RAWI[1]; HENDRA HIDAYAT[2]

[1]National Defence University of Malaysia, Faculty of Defence Science and Technology, Level 2 Bangunan Lestari, Kem Perdana Sungai Besi, 57000 Kuala Lumpur, Malaysia
[2]Department of Electronics Engineering, Universitas Negeri Padang, Padang, Indonesia
*Corresponding Author: rizal@upnm.edu.my

Abstract

Dynamic Quality of Service (QoS) in computer networks refers to techniques, mechanisms, or technologies that have the ability to automatically adjust QoS requirements or parameters to optimize network performance in real-time without the need for human intervention and manual configuration. As networks grow, maintaining optimal QoS capabilities in the face of high, fluctuating traffic loads and varying application demands becomes increasingly important and challenging. Based on the dynamic QoS that has been developed by the author through previous publications, this paper introduces a new enhancement that leverages deep learning (DL) techniques to analyse bandwidth and jitter (input parameters) to optimise the network automatically, more sophisticatedly, and responsively. Unlike traditional methods that rely on static rules or simple conditional logic, our proposed solution uses a neural network model to predict bandwidth allocation requirements based on real-time network conditions. Indirectly, the proposed solution is significantly able to improve the adaptability and efficiency of the QoS adjustment. Through comprehensive simulation and testing in a Software Defined Network (SDN) test environment that has been developed using the MiniEdit (Mininet Graphical Emulator), the results show that our DL-Enhanced Dynamic QoS Framework not only outperforms conventional QoS management approaches in terms of latency and throughput but also ensures a higher quality of experience for end users. This research highlights the potential for integrating artificial intelligence (AI) with network management practices to address the complexities of modern network infrastructure and pave the way for more resilient and user-centric network solutions.

Keywords: Deep learning, DL, Dynamic QoS, QoS, Quality of service.

1. Introduction

Nowadays, in general, all ICT network management faces increasing challenges in ensuring optimal service delivery across complex and modern digital network infrastructures [1-4]. With the rise of the Fourth Industrial Revolution (IR4.0), along with the proliferation of the Internet of Things (IoT), smart home devices and sensors, the need for advanced Quality of Service (QoS) mechanisms that can dynamically adapt to modern network demands is more prominent [4-6]. Therefore, this paper is prepared to contribute to addressing this challenge by introducing a new approach to QoS management by leveraging deep learning (DL) techniques to enable more responsive and real-time network resource allocation.

The conventional QoS mechanisms which are effective in static environments often struggle to meet the diverse and unpredictable demands of modern networks [4-6]. The high traffic variability and the diversity of real-time or online application requirements have spurred the exploration of more adaptive solutions. In response to this, the Software-Defined Networking (SDN) approach, which is a foundation for flexible network management, that comes with centralized control and a comprehensive view of network states was created in the 2010s [4-6]. However, SDN's potential for dynamic QoS remains largely untapped due to its limited predictive capabilities. Motivated by these issues, this paper was written, and research has been implemented to integrate intelligence into SDN to enhance network ability to meet real-time network traffic requirements.

Based on the previous work by Osman et al. [7], a dynamic QoS framework capable of automatically modifying the maximum bandwidth rate limit of a QoS queue was introduced. This framework operates using predefined IF/ELSE conditional logic. To enhance its capabilities, the current paper proposes integrating the existing dynamic QoS framework with a deep learning (DL) approach.

DL approach is a subset of machine learning (ML), and both are a part of AI. It uses artificial neural networks to learn and make predictions from presented data. This approach consists of layers of connected nodes, also known as neurons, where each layer transforms the input data to identify patterns. Through this layered process, DL models can perform complex tasks like image recognition or language processing. In network management, DL is particularly useful because it can automatically adapt to changing conditions, such as varying traffic loads [4-8].

Our enhancement proposal or contribution for this research is to replace the existing IF/ELSE conditional logic used by existing dynamic QoS frameworks with a DL model, in a way to analyse and predict bandwidth and jitter for automatic bandwidth allocation in real-time. This enhancement offers a more efficient and responsive mechanism for bandwidth management. The introduction of DL into dynamic QoS represents a significant step forward in the quest for smarter and autonomous network management systems [4, 6, 8, 9]. By harnessing the power of AI, networks can become more adaptive and more predictive in managing resources, indirectly it will ensure that critical/sensitive traffic application requirements are met without compromising the efficiency and reliability of the network infrastructure.

In contributing to dynamic QoS research, several previous research papers by previous researchers and scholars are relevant. Xie et al. [6] discuss the use of machine learning techniques to enhance dynamic QoS specifically in Software

Defined Networking (SDN). They highlight how this approach can automate, manage, and optimize network systems to address the challenges posed by traditional network configurations.

This research emphasizes the benefits of SDN features such as centralized control and dynamic updating of forwarding rules, which facilitate the implementation of ML algorithms for traffic classification, routing optimization and network resource management. It also suggests that the integration of ML into SDN can lead to smarter network management solutions, improving QoS and overall network performance.

Deva Sarma [4] provides a comprehensive survey on how machine learning (ML) and deep learning (DL) can enhance QoS in Software Defined Networking (SDN) by optimizing traffic management tasks such as classification, routing, and queuing. Unlike the broader ML focus presented by Xie et al. [6], this work offers an in-depth analysis of specific QoS-aware protocols that leverage ML and DL to improve network service quality.

Fadlullah et al. [10] emphasize the transformative impact of deep learning (DL) on network traffic control systems, particularly highlighting its role in enhancing network infrastructure. They describe how DL can be integrated into networks to improve management and efficiency through intelligent routing strategies that dynamically adapt to real-time network conditions. This marks a significant advancement over traditional methods, which lack flexibility and are less capable of handling the complexities of modern network traffic demands.

Arif et al. [11] introduce the DQQS model, which integrates deep reinforcement learning with Software Defined Networking (SDN) in an SDN-IoT environment. The model optimizes both security and performance by allocating bandwidth to critical applications and proves effective in managing multiple devices and variable traffic. It leverages deep learning to enhance overall network efficiency. Rodríguez et al. [12] explore the use of machine learning techniques to optimize path planning in SDN. They highlight how the adaptability of SDN enables rapid responses to network changes by employing support vector machines (SVM), k-nearest neighbours (kNN), and artificial neural networks (ANN) for efficient path management.

Owusu and Nayak [13] developed a framework that integrates Software Defined Networking (SDN) and Virtual Network Functions (VNF) with Deep Neural Networks to achieve optimal routing, reporting an impressive classification accuracy of 99.95%. However, the study focused solely on classification accuracy, without evaluating the impact on overall network performance. In contrast, Alkubeily et al. [14] enhanced SDN multi-path routing by adapting the Uniform Cost Search (UCS) algorithm through artificial intelligence techniques, demonstrating improved performance over traditional Depth First Search (DFS). While effective, their approach is complex, resource-intensive, and presents scalability limitations.

Oh et al. [15] present a U.S. patent that addresses service quality degradation caused by traffic overload in SDN environments. The patent outlines a method for dynamically managing bandwidth at congested nodes by collecting real-time traffic data, detecting congestion, assessing available network resources, and adjusting bandwidth allocations based on Service Level Agreement (SLA) parameters. This method is reactive in nature, targeting specific nodes without improving overall network performance, and it does not incorporate predictive analytics, machine learning (ML), or deep learning (DL) techniques.

Osman et al. [7], in our previous work, focused on enhancing network performance by dynamically managing Quality of Service (QoS) in enterprise and local area networks. Leveraging Software Defined Networking (SDN), the proposed approach adaptively modifies the maximum bandwidth rate-limit of QoS queues to prioritize latency-sensitive traffic - such as video conferencing, VoIP, and media streaming - during periods of high network utilization. The framework adjusts the bandwidth rate-limit of the best-effort traffic queue based on real-time traffic, allocating more bandwidth to critical applications as needed. Operated within a simulated SDN environment using MiniEdit, the system has demonstrated significant network performance improvements in comparative studies, automatically adjusting bandwidth allocations to maintain high service quality for sensitive traffic without manual intervention.

Several studies [16–19] have focused on the application of machine learning (ML) techniques, while others [20–23] have employed deep learning (DL) approaches specifically for traffic classification—an important but singular aspect of QoS functionality. Additionally, research works [24–27] have investigated the use of ML and DL for traffic prediction in Software Defined Networking (SDN). Meanwhile, studies [28–32] have primarily concentrated on optimizing routing in SDN using both ML and DL methods.

Motivated by the context discussed above, this paper proposes to enhance [7] by integrating with DL techniques. This approach utilises a neural network to more precisely analyse and predict network conditions via jitter and bandwidth in real-time. Thereby enabling more effective bandwidth management strategies.

The new proposed framework known as DL-Enhanced Dynamic QoS Framework aims to not only continue the work of dynamically adjusting QoS based on current network needs, but also to enhance these adjustments' accuracy and efficiency leveraging DL.

The proposed Framework is developed for SDN-ready networks. It allows seamless integration with existing infrastructure, reduces costs and simplifies deployment. For non-SDN networks, new equipment would be required. The framework is scalable, easy to expand and enables the replacement of devices without major reconfiguration making it ideal for enterprise environments with evolving demands.

With SDN’s centralized control, network monitoring and management are simplified. Administration and oversee devices only from a central location, allowing quick responses to changing conditions. The framework’s Python-based design also facilitates updates and maintenance ensuring ongoing efficiency and adaptability. Overall, these features make the framework a practical choice for real-world applications, especially for organizations seeking minimal disruption and maximum scalability in enhancing their SDN networks.

This paper details the methodology of our proposed DL-Enhanced Dynamic QoS Framework, presents findings from extensive simulations within an SDN experimental infrastructure environment, and discusses the implications of our research for the future of network management.

2. Methods

The proposed DL-Enhanced Dynamic QoS Framework operates similarly to the approach presented by Osman et al. [7], as illustrated in Fig. 1. During the network router at normal incoming traffic conditions, the configuration of the network router is default, and it works as a best-effort mechanism. When the incoming traffic increases to heavier traffic conditions, the proposed DL-Enhanced Dynamic QoS Framework will expand the bandwidth size of the sensitive traffic queue to accommodate incoming traffic. When the incoming traffic increases to the heaviest traffic conditions, the proposed framework will expand the bandwidth size of the sensitive traffic queue to its maximum. The bandwidth size provided varies and is adaptive depending on the volume of the incoming traffic as depicted in Fig. 2.

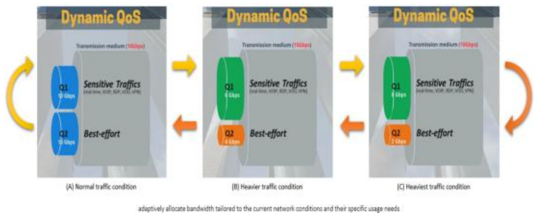

Dynamic QoS
Sensitive Traffics
Best-effort
Q1
Q2
(A) Normal traffic condition
(B) Heavier traffic condition
(C) Heaviest traffic condition
adaptively allocate bandwidth tailored to the current network conditions and their specific usage needs

Fig. 1. DL-enhanced dynamic QoS framework mechanism.

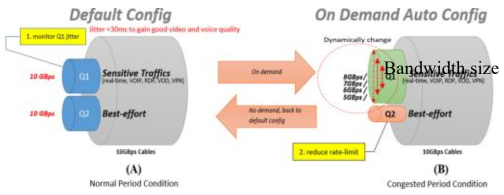

Default Config
On Demand Auto Config
1. monitor Q1 jitter
Jitter <30ms to gain good video and voice quality
10 GBps
Q1
Sensitive Traffics (real-time, VOIP, RDP, VCEO, VPN)
Q2
Best-effort
On demand
No demand, back to default config
Dynamically change
Bandwidth size
8GBps / 7GBps / 6GBps / 5GBps /
2. reduce rate-limit
10GBps Cables
(A) Normal Period Condition
(B) Congested Period Condition

Fig. 2. DL-enhanced dynamic QoS framework approach.

The configuration of the routers will return to its default configuration if there is no/less incoming traffic in the network. Incoming traffic conditions are monitored via a Jitter which represents delay variation. Jitter value may increase caused by heavy/high volume of incoming traffic. The higher the jitter value it presents, the longer the delay will be.

The purpose of this mechanism is to guarantee high priority for sensitive traffic (sensitive traffic application example: video conferencing, Live video, VoIP, and media streaming) run successfully under limited network capacity. Jitter must be less than 30 milliseconds for video and voice to gain good quality [33]. By reducing the QoS Queue's Maximum Bandwidth Rate-Limit of the best-effort queue, the rest of the available bandwidth of the transmission medium will be provided as guarantee/adequate bandwidth for the Sensitive Traffic queue [7].

The development and testing of the proposed DL-Enhanced Dynamic QoS Framework will be implemented in the same simulation environment as [7]. The methodology for this research is shown in Fig. 3. and the details explanation of each phase is described in Sections 2.2 to 2.7.

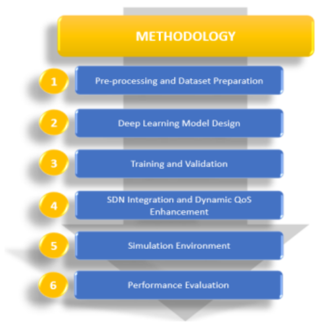

METHODOLOGY
1 Pre-processing and Dataset Preparation
2 Deep Learning Model Design
3 Training and Validation
4 SDN Integration and Dynamic QoS Enhancement
5 Simulation Environment
6 Performance Evaluation

Fig. 3. A research method.

2.1. Pre-processing and dataset preparation

Pre-processing and Dataset Preparation are the phases of preparing the DL training dataset and to ensuring the model's input would be clean and structured. Data was collected from the routers directly. The information gathered from each router is (i) the *Q1 jitter value* and (ii) the *current Q2 maximum bandwidth rate-limit* information, as illustrated in Fig. 4. The collected data were stored as CSV file format in SDN Controller which is in the same place DL-Enhanced Dynamic QoS Framework was located.

Figure 5 presents a sample pseudo-code illustrating the method used to capture (i) and (ii). Compared to the previous version discussed in the earlier publication, the DL-Enhanced Dynamic QoS Framework requires data to effectively learn and make accurate predictions. Therefore, we provide a training dataset as shown in Fig. 6.

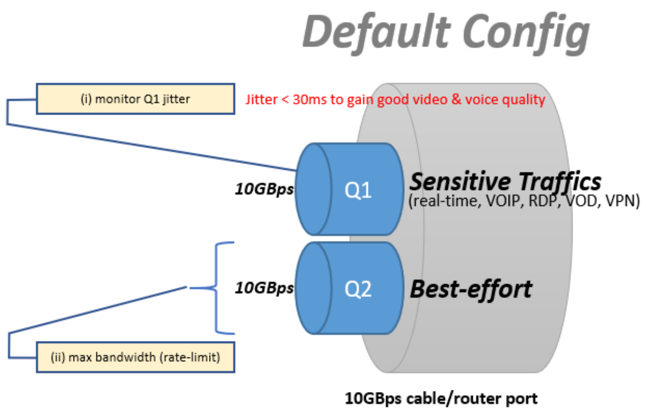
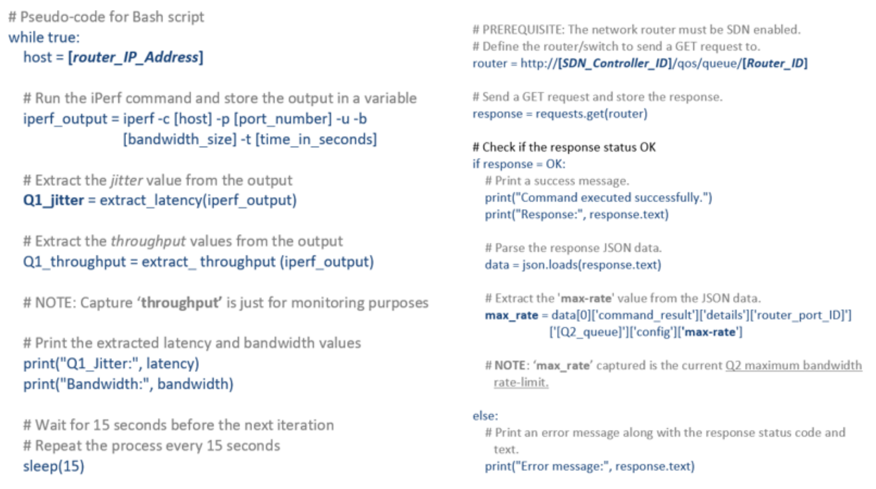

Default Config
(i) monitor Q1 jitter
Jitter < 30ms to gain good video & voice quality
10GBps
Q1
Sensitive Traffics
(real-time, VOIP, RDP, VOD, VPN)
10GBps
Q2
Best-effort
(ii) max bandwidth (rate-limit)
10GBps cable/router port

Fig. 4. Maximum bandwidth rate-limit & Jitter.



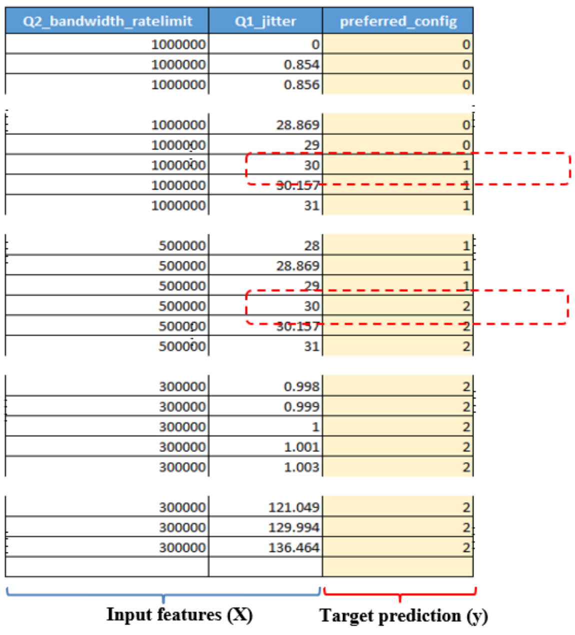
```
# Pseudo-code for Bash script
while true:
   host = [router_IP_Address]

   # Run the iPerf command and store the output in a variable
   iperf_output = iperf -c [host] -p [port_number] -u -b
                    [bandwidth_size] -t [time_in_seconds]

   # Extract the jitter value from the output
   Q1_jitter = extract_latency(iperf_output)

   # Extract the throughput values from the output
   Q1_throughput = extract_ throughput (iperf_output)

   # NOTE: Capture 'throughput' is just for monitoring purposes

   # Print the extracted latency and bandwidth values
   print("Q1_Jitter:", latency)
   print("Bandwidth:", bandwidth)

   # Wait for 15 seconds before the next iteration
   # Repeat the process every 15 seconds
   sleep(15)
```

(ii) Capture Q1 jitter

```
# PREREQUISITE: The network router must be SDN enabled.
# Define the router/switch to send a GET request to.
router = http://[SDN_Controller_ID]/qos/queue/[Router_ID]

# Send a GET request and store the response.
response = requests.get(router)

# Check if the response status OK
if response = OK:
   # Print a success message.
   print("Command executed successfully.")
   print("Response:", response.text)

   # Parse the response JSON data.
   data = json.loads(response.text)

   # Extract the 'max-rate' value from the JSON data.
   max_rate = data[0]['command_result']['details']['router_port_ID']
              ['[Q2_queue]']['config']['max-rate']

   # NOTE: 'max_rate' captured is the current Q2 maximum bandwidth
     rate-limit.

else:
   # Print an error message along with the response status code and
     text.
   print("Error message:", response.text)
```

(i) Capture Q2 max bandwidth rate-limit

Fig. 5. Pseudo-code to capture (i) and (ii) [7].



Q2_bandwidth_ratelimit	Q1_jitter	preferred_config
1000000	0	0
1000000	0.854	0
1000000	0.856	0
1000000	28.869	0
1000000	29	0
1000000	30	1
1000000	30.157	1
1000000	31	1
500000	28	1
500000	28.869	1
500000	29	1
500000	30	2
500000	30.157	2
500000	31	2
300000	0.998	2
300000	0.999	2
300000	1	2
300000	1.001	2
300000	1.003	2
300000	121.049	2
300000	129.994	2
300000	136.464	2

Input features (X)
Target prediction (y)

Fig. 6. Training dataset sample.

The DL-Enhanced Dynamic QoS Framework will use a Supervised DL Learning Model. Supervised learning involves training a model on a labelled dataset. It means that each input in the training set is paired with an output label. The DL model learns to map inputs to outputs based on the dataset provided.

Therefore, as shown in Fig. 6, we provide *preferred_config* as a **DL target prediction (y)** or output label, *Q2_bandwidth_ratelimit* and *Q1_jitter* as a **DL input features (X)**.

For the training dataset, we train the model input to pair with *preferred_config* either 0, 1 and 2. The definition of each *preferred_config* is described in Table 1.

Later *preferred_config* will be used by the DL-Enhanced Dynamic QoS Framework to modify the QoS queue bandwidth of the applicable router/switch to accommodate incoming traffic and to provide guaranteed bandwidth for sensitive traffic to travel across the network on demand.

Table 1. *preferred_config* definition.

preferred_config	Class / Condition
0	Default configuration or return to default;
1	Reduce *Q2_bandwidth_ratelimit* to 500MBps and automatically Sensitive Traffic queue will gain 500MBps guarantee bandwidth;
2	Reduce *Q2_bandwidth_ratelimit* to 200MBps and automatically Sensitive Traffic queue will gain 800MBps guarantee bandwidth.

2.2. DL model design

DL tasks are categorized by prediction type: Binary, Multi-Label, or Multi-Class Classification. Binary Classification predicts between two possible outcomes, while Multi-Label Classification assigns multiple labels to each input. In Multi-Class Classification, each input is assigned to one of three or more classes.

Multi-Class Classification suits the DL-Enhanced Dynamic QoS Framework by pairing two inputs (bandwidth, jitter) with a *preferred_config* {0, 1, 2, or 3}. One-hot encoding is used here, converting each label into a binary vector where the index of the class label is set to 1, and other positions are set to 0, as shown in Table 2.

Table 2. One-hot encoding.

Label		Binary
0	:	[1, 0, 0]
1	:	[0, 1, 0]
2	:	[0, 0, 1]

To identify the most suitable DL model for the DL-Enhanced Dynamic QoS Framework, we tested two models: Artificial Neural Network (ANN) and Convolutional Neural Network (CNN). These models were chosen due to their different input processing methods. ANN processes input as flat vectors, while CNN maintains spatial structures, making it ideal for two- or three-dimensional data like images. The Python source code for both models can be accessed at: https://drive.google.com/drive/folders/16sgJkSmv-Zkb-Rcz3bqW8m2w372tMhbR?usp=sharing

2.3. Training and validation

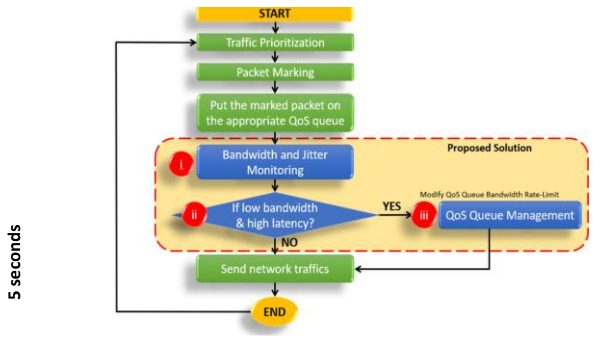
The training process involves feeding data sets to the ANN and CNN models in batches and evaluating them using classification accuracy (a metric that calculates the proportion of correct predictions). This process aims to identify the best model (with the highest accuracy) to fit the DL-Enhanced Dynamic QoS Framework. By using a dataset of 2,500 rows containing bandwidth and jitter as input features, and preferred_config as target label, the ANN shows better accuracy than CNN. Detailed results are discussed in the Results and Discussion section.

2.4. SDN integration and dynamic QoS enhancement

The integration of the DL-Enhanced Dynamic QoS Framework with SDN marks a significant advance in network management capabilities. This DL-Enhanced Dynamic QoS Framework leverages the centralized control and flexibility of SDN to implement real-time, intelligent adaptations to network conditions based on predictive analytics provided by the DL model.

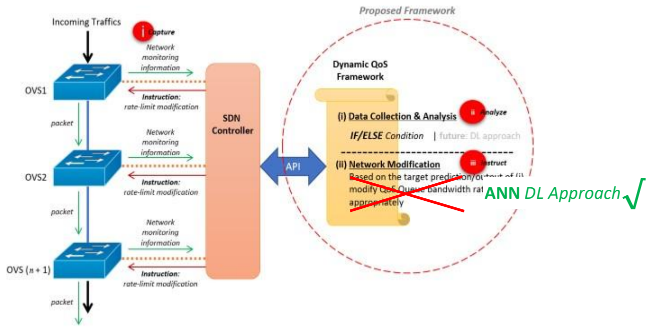
This integration allows for dynamic adjustments of bandwidth in real-time, which are crucial for prioritizing sensitive traffic applications under varying network loads. With the neural network, the system can predict potential network congestion and adjust QoS parameters proactively rather than reactive. This ensures high-quality service continuity without manual intervention making the network management process more efficient and effective.

For seamless integration with SDN, our DL-Enhanced Dynamic QoS Framework and DL script (modelANN.h5), have been deployed within the simulation environment. While the conceptual framework aligns with that established in [7] as depicted in Fig. 7, we have transitioned from using IF/ELSE conditional logic to a deep learning model for the *Analysing Bandwidth and Latency* process as shown in Fig. 8. Dynamic QoS is also improved by speeding up the network monitoring process from 15 seconds to 5 seconds.



5 seconds
START
Traffic Prioritization
Packet Marking
Put the marked packet on the appropriate QoS queue
Proposed Solution
i
Bandwidth and Jitter Monitoring
ii
If low bandwidth & high latency?
YES
Modify QoS Queue Bandwidth Rate-Limit
iii
QoS Queue Management
NO
Send network traffics
END

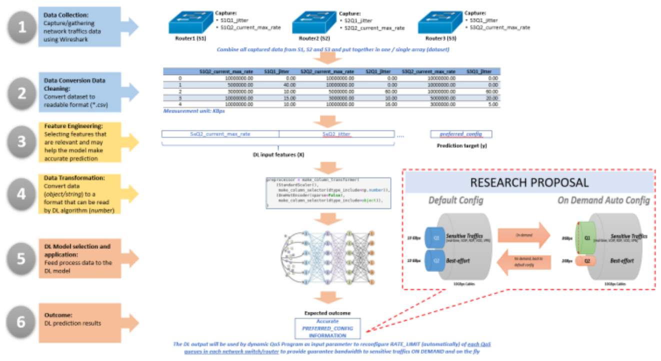
Fig. 7. Conceptual framework [7].



Proposed Framework
Incoming Traffics
i Capture
Network monitoring information
Instruction: rate-limit modification
OVS1
packet
OVS2
OVS (n + 1)
SDN Controller
API
Dynamic QoS Framework
(i) Data Collection & Analysis
ii Analyze
IF/ELSE Condition | future: DL approach
(ii) Network Modification
iii Instruct
~~Based on the target prediction/output of (i) modify QoS Queue bandwidth rate appropriately~~
ANN DL Approach √

Fig. 8. Process flow diagram.

The overall overview DL process of the DL-Enhanced Dynamic QoS Framework is illustrated in Fig. 9.


1
Data Collection: Capture/gathering network traffics data using Wireshark
2
Data Conversion Data Cleaning: Convert dataset to readable format (*.csv)
3
Feature Engineering: Selecting features that are relevant and may help the model make accurate prediction
4
Data Transformation: Convert data (object/string) to a format that can be read by DL algorithm (number)
5
DL Model selection and application: Feed process data to the DL model
6
Outcome: DL prediction results
Capture: • S1Q1_jitter • S1Q2_current_max_rate
Router1 (S1)
Capture: • S2Q1_jitter • S2Q2_current_max_rate
Router2 (S2)
Capture: • S3Q1_jitter • S3Q2_current_max_rate
Router3 (S3)
Combine all captured data from S1, S2 and S3 and put together in one / single array (dataset)
Measurement unit: KBps
SxQ2_current_max_rate
SxQ2_jitter
....
preferred_config
DL input features (X)
Prediction target (y)
RESEARCH PROPOSAL
Default Config
On Demand Auto Config
Q1
Q2
Sensitive Traffics
Best-effort
On demand
No demand, back to default config
10Gbps Cables
Expected outcome
Accurate PREFERRED_CONFIG INFORMATION
The DL output will be used by dynamic QoS Program as input parameter to reconfigure RATE_LIMIT (automatically) of each QoS queues in each network switch/router to provide guarantee bandwidth to sensitive traffics ON DEMAND and on the fly

	S1Q2_current_max_rate	S1Q1_jitter	S2Q2_current_max_rate	S2Q1_jitter	S3Q2_current_max_rate	S3Q1_jitter
0	10000000.00	0.00	10000000.00	0.00	10000000.00	0.00
1	5000000.00	40.00	10000000.00	0.00	10000000.00	0.00
2	3000000.00	10.00	5000000.00	60.00	10000000.00	60.00
3	10000000.00	15.00	3000000.00	10.00	5000000.00	20.00
4	10000000.00	10.00	10000000.00	16.00	3000000.00	5.00

```
preprocessor = make_column_transformer(
    (StandardScaler(),
     make_column_selector(dtype_include=np.number)),
    (OneHotEncoder(sparse=False),
     make_column_selector(dtype_include=object)),
)
```

Fig. 9. Deep learning process.

The results of the DL process will be utilised by the DL-Enhanced Dynamic QoS Framework through the *QoS Queue Management process* to adjust/modify the Bandwidth Rate-Limit of the QoS Queue.

The general overview of how the DL-Enhanced Dynamic QoS Framework integrates/works within the simulation environment is shown in Fig. 10. The Python source code for the DL-Enhanced Dynamic QoS Framework is also accessible via a Google Drive link provided.

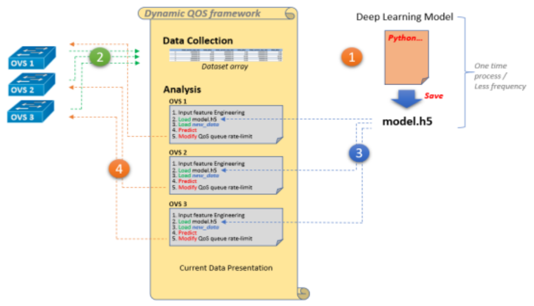


Fig. 10. General overview of how the proposed framework works.

2.5. Simulation environment

The simulation environment for evaluating the DL-Enhanced Dynamic QoS Framework mirrors the setup in [7], using MiniEdit (Graphical Mininet emulator) as shown in Fig. 11, to simulate various network conditions, from normal to high congestion. As in [7], the demonstration involves three routers representing the access, distribution/core, and gateway routers of an Enterprise Network, as shown in Fig. 12.

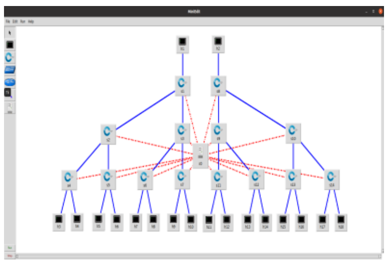


Fig. 11. SDN network topology on miniedit (graphical mininet).

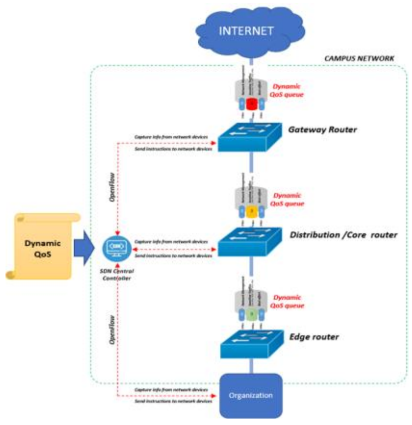

INTERNET
CAMPUS NETWORK
Dynamic QoS queue
Gateway Router
Capture info from network devices
Send instructions to network devices
OpenFlow
Dynamic QoS queue
Distribution /Core router
Dynamic QoS
Capture info from network devices
Send instructions to network devices
SDN Central Controller
Dynamic QoS queue
OpenFlow
Edge router
Capture info from network devices
Send instructions to network devices
Organization

Fig. 12. Simulation/testing environment.

2.6. Performance evaluation

As mentioned in section 2.4, the performance of the DL model was evaluated via Classification Accuracy. We examined that ANN has a better accuracy compared to CNN and the results will be discussed further in the Results and Discussion section.

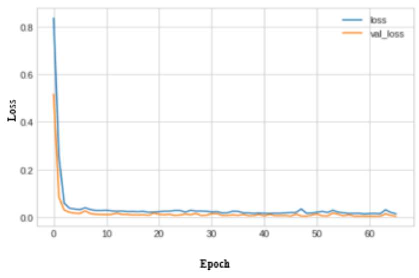
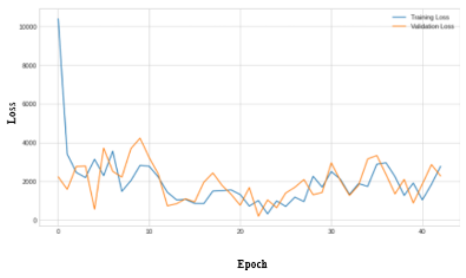
The performance of the DL-Enhanced Dynamic QoS Framework evaluated via comparison study as similar as testing method applied in [7]. The testing script as follows:

TESTING SCRIPT: Throughput and Jitter evaluation.

Step 1:	Enable the Proposed DL-Enhanced Dynamic QoS framework program.
Step 2:	Run two iPerf packet generators simultaneously. One of them will represent sensitive traffic (run it for 100 ms) and the other will represent best-effort traffic (run it for 120 ms). The proposed framework uses differentiated service as a classification technique. It classifies based on a 6-bit Differentiated Services Code Point (DSCP) that is located in the IP header.
Step 3:	Capture the value of *Q2 maximum bandwidth rate-limit* and *Q1 jitter* for every millisecond and tabulate it into a graph.
Step 4:	Repeat Step 2 to Step 3 without enabling the Proposed Dynamic QoS Framework program.
Step 5:	Compare the observed differences.

3. Results and Discussion

Figure 13 shows the *Classification Accuracy* outcome for ANN and CNN models.

ANN Model	CNN Model
	

```
Final Training Loss: 0.1301
Final Training Accuracy: 94.29%
Final Validation Loss: 0.1320
Final Validation Accuracy: 93.87%
Model: "sequential_38"
```

```
Final Training Loss: 152.9779
Final Training Accuracy: 34.11%
Final Validation Loss: 195.9454
Final Validation Accuracy: 31.52%
Model: "sequential_15"
```

Fig. 13. Classification accuracy results.

Based on the results presented, can be concluded that the ANN model is significantly more effective and stable compared to the CNN model for the dataset provided. It archives high accuracy with minimal loss. The CNN model, while typically strong in imaging and spatial data application performs poorly in this scenario. The CNN underperforms because of insufficient complexity and the size dimension of input data (*bandwidth* and *jitter*) is too small. Therefore, for small/low-dimension datasets of this scenario, using simpler models like ANN is more suitable.

The outcome of the *Throughput Evaluation* testing is presented in Figs. 14 and 15.

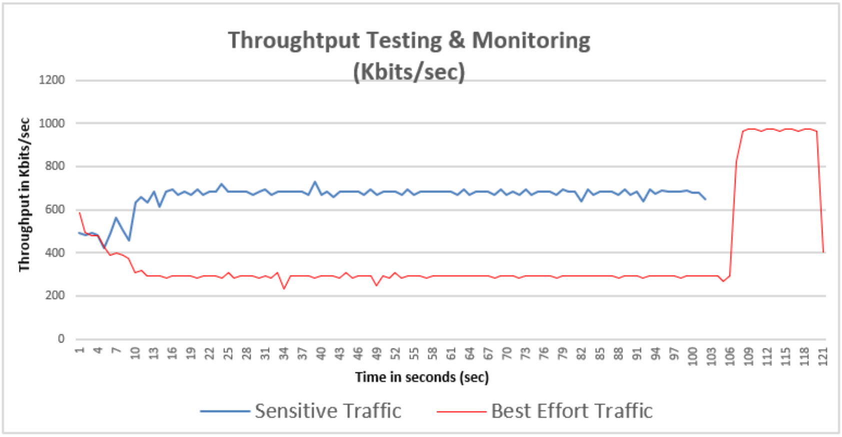


Fig. 14. Test with DL-enhanced dynamic QoS framework enable.

IN summary, the graph in Fig. 14 shows throughput testing for (i) Sensitive Traffic and (ii) Best Effort traffic under the DL-Enhanced Dynamic QoS

Framework. When priority is given, the sensitive traffic maintains stable throughput, demonstrating the framework's ability to prioritize and allocate bandwidth effectively-crucial for applications like video conferencing and VoIP.

Best-effort traffic, which includes general data tasks, experiences a drop when sensitive traffic is prioritized but shows stability until a spike at the end. This indicates that the framework dynamically reallocates bandwidth, allowing best-effort traffic to use more resources when sensitive traffic demands decrease. The final increase in best-effort throughput reflects this adaptive bandwidth management, optimizing resource utilization.

Overall, this behaviour demonstrates the framework's capacity to adjust to sudden network demand shifts, ensuring high bandwidth for priority traffic while allowing less critical tasks to benefit when network conditions permit, without compromising service quality for sensitive applications.

Figure 15 compares sensitive traffic throughput across three configurations: (i) DL-Enhanced Dynamic QoS Framework (blue), (ii) previous Dynamic QoS Framework (red), and (iii) without QoS (grey). The DL-Enhanced framework consistently maintains a throughput of around 700 Kbits/sec, demonstrating effective bandwidth management. The previous Dynamic QoS framework shows an increase from 500 Kbits/sec at 6 to 21 seconds, 650 Kbits/sec from 21 to 36 seconds, and stabilizes near 700 Kbits/sec. In contrast, the no-QoS setup is highly variable, often below 500 Kbits/sec, indicating poor performance.

Overall, the DL-Enhanced framework outperforms both alternatives, providing improved stability and throughput. It also reduces network monitoring intervals from 15 to 5 seconds, allowing faster responses to network changes and more efficient bandwidth allocation, which enhances network performance and user experience

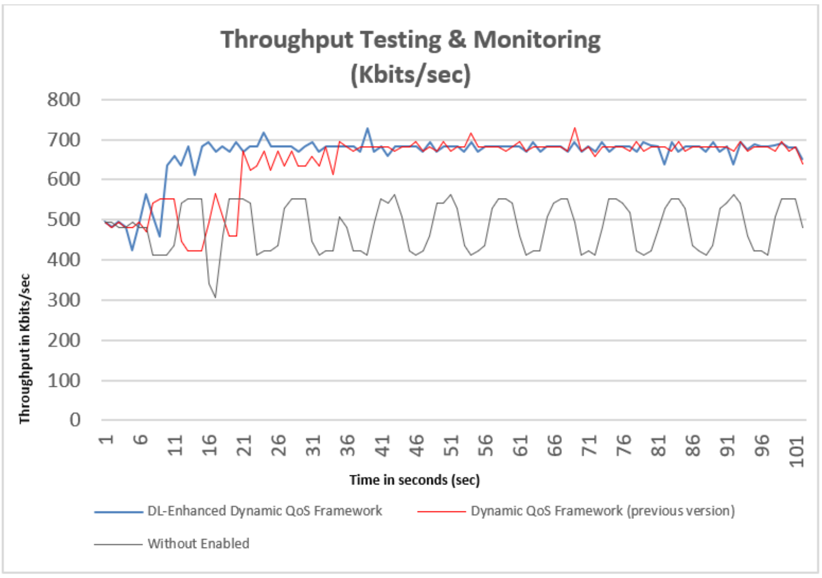


Fig. 15. Comparison of sensitive traffic throughput gained via (i) DL-Enhanced Dynamic QoS Framework, (ii) Dynamic QoS Framework (previous version) and without QoS framework enabled.

4. Conclusion

The DL-Enhanced Dynamic QoS Framework significantly improves network performance by providing stable, high throughput for sensitive traffic and effectively managing real-time bandwidth allocation. These results show that the DL-Enhanced Dynamic QoS Framework is more responsive compared to traditional methods.

Comparative analysis shows that the DL-Enhanced Framework outperforms previous versions of Dynamic QoS and conventional methods. It provides consistent throughput for critical/sensitive traffic applications. Besides, by reducing the monitoring interval from 15 to 5 seconds, results show further improved responsiveness, allowing for immediate adjustments to changing network conditions.

Integrating DL into dynamic QoS is an important step towards smarter and autonomous network management. This research demonstrates the potential of AI in optimizing network performance, paving the way for more adaptive and user-centric solutions. Future work could expand AI-driven QoS management across network infrastructure, increasing predictive capabilities and resilience to meet the complex demands of modern networks.

Acknowledgement

The authors would like to express their deepest gratitude for the funding provided by the Ministry of Higher Education under the Fundamental Research Grant Scheme (FRGS) (FRGS/1/2023/ICT11/UPNM/02/1) for this research. We also express our gratitude to the National Defence University of Malaysia (NDUM), lecturers, fellow researchers, collaborators partners, and network SMEs for their valuable time and support towards this paper and research.

References

1. Safar, N.Z.M.; Abdullah, N.; Kamaludin, H.; Abd Ishak, S.; and Isa, M.R.M. (2020). Characterising and detection of botnet in P2P network for UDP protocol. *Indonesian Journal of Electrical Engineering and Computer Science*, 18(3), 1584-1595.
2. Mustafa, M.F.; Isa, M.R.M.; Rauf, U.F.A.; Ismail, M.N.; Shukran, M.A.M.; Khairuddin, M.A.; Wahab, N.; and Safar, N.Z.M. (2021). student perception study on smart campus: A case study on higher education institution. *Malaysian Journal of Computer Science*, 1-20.
3. Isa, M.R.M.; Khairuddin, M.A.; Sulaiman, M.A.B.M.; Ismail, M.N.; Shukran, M.A.M.; and Sajak, A.A.B. (2021). SIEM network behaviour monitoring framework using deep learning approach for campus network infrastructure. *International Journal of Electrical and Computer Engineering Systems*, 9-21.
4. Deva Sarma, H.K. (2021). A survey on machine learning and deep learning-based quality of service aware protocols for software-defined networks. *TechRxiv*.
5. Khater, A.; and Hashemi, M.R. (2018). Dynamic flow management based on diffServ in SDN networks. *Proceedings of the Electrical Engineering* (*ICEE*), *Iranian Conference On*, Mashhad, 1505-1510.

6. 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(1), 393-430.
7. Osman, M.F.; Isa, M.R.M.; Khairuddin, M.A.; Shukran, M.A.M.; Razali, N.A.M.; Kamarudin, N.D.B.; Maskat, K., Rawi, R.; and Hidayat, H. (2023). Dynamic QoS: Automatically modifying QoS queue's maximum bandwidth rate-limit of network devices for network improvement. *International Journal on Advanced Science, Engineering and Information Technology*, 13(6), 2112-2119.
8. Abbasi, M.; Shahraki, A.; and Taherkordi, A. (2021). Deep learning for network traffic monitoring and analysis (NTMA): A survey. *Computer Communications*, 170, 19-41.
9. Bağiröz, B.; Güzel, M.; Yavanoğlu, U.; and Özdemir, S. (2019). QoS prediction methods in IoT a survey. *Proceedings of the* 2019 *IEEE International Conference on Big Data* (*Big Data*), Los Angeles, CA, USA, 2128-2133.
10. Fadlullah, Z.M.; Tang, F.; Mao, B.; Kato, N.; Akashi, O.; Inoue, T.; and Mizutani, K. (2017). State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials*, 19(4), 2432–2455.
11. Arif, F.; Khan, N.A.; Iqbal, J.; Karim, F.K.; Innab, N.; and Mostafa, S.M. (2024). DQQS: Deep reinforcement learning based technique for enhancing security and performance in SDN-IoT environments. *IEEE Access*, 12, 60568-60587.
12. Rodríguez, M.; Moyano, R.F.; Pérez, N.; Riofrío, D.; and Benítez, D. (2021). Path planning optimization in SDN using machine learning techniques. *Proceedings of the* 2021 *IEEE Fifth Ecuador Technical Chapters Meeting* (*ETCM*), Cuenca, Ecuador.
13. Isaac Owusu, A.; and Nayak, A. (2020). A framework for QoS-based routing in SDNs using deep learning. *Proceedings of the* 2020 *International Symposium on Networks*, *Computers and Communications* (*ISNCC*), Montreal, QC, Canada, 1-6.
14. Alkubeily, M.; Hasan, B.; and Zudina, O.V. (2024). Enhancing multipath routing and QoS in SDNs with AI algorithms. *Proceedings of the* 2024 6th *International Youth Conference on Radio Electronics*, *Electrical and Power Engineering* (*REEPE*), Moscow, Russian Federation, 1-6.
15. Ok. K.K; Hwan, D.K.; and Yeol, O.J. (2020). Bandwidth control method and apparatus for solving service quality degradation caused by traffic overhead in SDN-based communication node (U.S. Patent No. 10,673,523 B2). *U.S. Patent and Trademark Office*. . Retrieved November 4, 2024, from https://portal. unifiedpatents.com/patents/patent/US-10673523-B2.
16. Owusu, A.I.; and Nayak, A. (2020). An intelligent traffic classification in SDN-IOT: A machine learning approach. *Proceedings of the* 2020 *IEEE International Black Sea Conference on Communications and Networking* (*BlackSeaCom*), Odessa, Ukraine, 1-6.

17. Deart, V.; Mankov, V.; and Krasnova, I. (2020). Development of a feature matrix for classifying network traffic in SDN in real-time based on machine learning algorithms. *Proceedings of the* 2020 *International Scientific and Technical Conference Modern Computer Network Technologies* (*MoNeTeC*), Moscow, Russia, 1-9.

18. Goud, N.P.K.; Reddy, G.S.C.; and Maryposonia, A. (2022). Traffic classification of SDN network using machine learning algorithms. *Proceedings of the* 2022 6th *International Conference on Intelligent Computing and Control Systems* (*ICICCS*), Madurai, India, 1181-1185.

19. Shafiq, M.; Yu, X.; Laghari, A.A.; Yao, L.; Karn, N.K.; and Abdessamia, F. (2016). Network traffic classification techniques and comparative analysis using machine learning algorithms. *Proceedings of the* 2016 2nd *IEEE International Conference on Computer and Communications* (*ICCC*), Chengdu, 2451-2455.

20. Wei, W.; Gu, H.; Deng, W.; Xiao, Z.; and Ren, X. (2022). ABL-TC: A lightweight design for network traffic classification empowered by deep learning. *Neurocomputing*, 489, 333-344.

21. Wu, Z.; Dong, Y.; Qiu, X.; and Jin, J. (2022). Online multimedia traffic classification from the QoS perspective using deep learning. *Computer Networks*, 204, 108716.

22. Long, Z.; and Jinsong, W. (2023). Network traffic classification based on a deep learning approach using netflow data. *The Computer Journal*, 66(8), 1882-1892.

23. Malik, A.; de Fréin, R.; Al-Zeyadi, M.; and Andreu-Perez, J. (2020). Intelligent SDN traffic classification using deep learning: Deep-SDN. *Proceedings of the* 2020 2nd *International Conference on Computer Communication and the Internet (ICCCI)*, Nagoya, Japan, 184-189.

24. Volkov, A.; Proshutinskiy, K.; Adam, A.B.M.; Ateya, A.A.; Muthanna, A.; and Koucheryavy, A. (2019). *SDN load prediction algorithm based on artificial intelligence*. In Vishnevskiy, V.M.; Samouylov, K.E.; and Kozyrev, D.V. (Eds.), *Distributed Computer and Communication Networks*. Springer International Publishing.

25. Wassie, G.; Ding, J.; and Wondie, Y. (2023). Traffic prediction in SDN for explainable QoS using deep learning approach. *Scientific Reports*, 13(1), 20607.

26. Abood, M.S.; Wang, H.; He, D.; Fathy, M.; Rashid, S.A.; Alibakhshikenari, M.; Virdee, B.S.; Khan, S.; Pau, G.; Dayoub, I.; Livreri, P.; and Elwi, T.A. (2023). An LSTM-based network slicing classification future predictive framework for optimized resource allocation in C-V2X. *IEEE Access*, 11, 129300-129310.

27. Jiang, W.; Han, H.; He, M.; and Gu, W. (2024). ML-based pre-deployment SDN performance prediction with neural network boosting regression. *Expert Systems with Applications*, 241, 122774.

28. Al-Obaidi, Z.H.S. (2023). *IOT routing optimization by using the K-Means clustering algorithm and whale optimization algorithm*. MSc dissertation, Altınbaş Üniversitesi/Lisansüstü Eğitim Enstitüsü.

29. Gunavathie, M.A.; and Umamaheswari, S. (2024). Traffic-aware optimal routing in software defined networks by predicting traffic using neural network. *Expert Systems with Applications*, 239, 122415.
30. Casas-Velasco, D.M.; Rendon, O.M.C.; and da Fonseca, N.L.S. (2021). DRSIR: A deep reinforcement learning approach for routing in software-defined networking. *IEEE Transactions on Network and Service Management*, 19(4), 4807-4820.
31. Hussein, D.H.; and Askar, S. (2023). Federated learning enabled SDN for routing emergency safety messages (ESMs) in IoV under 5G environment. *IEEE Access*, 11, 141723-141739.
32. He, Y.; Xiao, G.; Zhu, J.; Zou, T.; and Liang, Y. (2024). Reinforcement learning based SDN routing scheme empowered by causality detection and GNN. *Frontiers in Computational Neuroscience*, 18, 1393025.
33. CiscoPress. (2022). Network fundamentals: Introduction to network performance measurement. Retrieved October 5, 2024, from https://www.ciscopress.com/.