EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A computationally intelligent framework for traffic engineering and congestion management in software-defined network (SDN)

L. Leo Prasanth[1*] and E. Uma[1]

*Correspondence:
lleo1306@gmail.com

[1] Department of Information Science and Technology, College of Engineering Guindy, Anna University, Chennai, India

Abstract

Software-defined networking (SDN) revolutionizes network administration by centralizing control and decoupling the data plane from the control plane. Despite its advantages, the escalating volume of network traffic induces congestion at nodes, adversely affecting routing quality and overall performance. Addressing congestion has become imperative due to its emergence as a fundamental challenge in network management. Previous strategies often faced drawbacks in handling congestion, with issues arising from the inability to efficiently manage heavy packet surges in specific network regions. In response, this research introduces a novel approach integrating a multiplicative gated recurrent neural network with a congestion-aware hunter prey optimization (HPO) algorithm for effective traffic management in SDN. The framework leverages machine learning and deep learning techniques, acknowledged for their proficiency in processing traffic data. Comparative simulations showcase the congestion-aware HPO algorithm's superiority, achieving a normalized throughput 3.4–7.6% higher than genetic algorithm (GA) and particle swarm optimization (PSO) alternatives. Notably, the proposed framework significantly reduces data transmission delays by 58–65% compared to the GA and PSO algorithms. This research not only contributes a state-of-the-art solution but also addresses drawbacks observed in existing methodologies, thereby advancing the field of traffic engineering and congestion management in SDN. The proposed framework demonstrates notable enhancements in both throughput and latency, providing a more robust foundation for future SDN implementations.

Keywords: Software-defined network, Multiplicative gated recurrent neural network, Hunter prey optimization, Traffic prediction, Congestion management

1 Introduction

Due to significant advancements in technology, the current network has become increasingly complex, resulting in network traffic flow [1]. This massive network traffic can lead to congestion. Consequently, network management and traffic measurement issues have emerged [2]. Manual network administration is impractical, and software-defined networking (SDN) addresses these problems by providing a centralized controller that monitors and collects parameters for efficient management and intelligent routing. SDN separates the control plane from the forwarding plane, allowing network intelligence to be centrally located in the control plane software controllers [3]. Communication between the planes is performed through an open interface known as the OpenFlow Protocol. OpenFlow is one of the primary SDN protocol standards [4].

SDN's network programmability promotes efficient network management, traffic management, dynamic resource management, and security. The objective of the SDN paradigm is to decouple network intelligence from network devices, enabling centralized network intelligence. However, as the network size and number of flows increase, the computational complexity of the control panel also increases exponentially. Additionally, the traffic explosion and the increase in various network requirements, driven by the rapid acceleration of 5G networks, multimedia data traffic, and cloud computing, pose numerous challenges in terms of routing problem complexity, network scale, and network traffic size. Traditional routing algorithms are not suitable for SDN due to their limitations in convergence, adaptability to network topology changes, and lack of future vision on network traffic. Traffic engineering (TE) in SDN involves examining network conditions through the SDN controller to act on flow data by rapidly changing flow table information for forwarding devices [5]. Periodically rerouting flows balance the load on the network, reducing congestion and enhancing network performance. Two types of traffic flows exist in a network: mice flow and elephant flow [6]. Elephant flows indicate heavy traffic flows that require more network resources, while quick accumulation of mice flows can also degrade network performance. These traffic flows continually require resource allocation for efficient usage of scarce resources through traffic engineering (TE).

Machine learning (ML) enables the logical mining of valuable data from collected data and automatically finds correlations. The heterogeneous traffic data generated from various sources exhibit various forms and complex correlations. Traditional ML can struggle to solve this issue of interest. ML offers poor performance when dealing with a large volume of traffic data and cannot handle high-dimensional data. With a large volume of traffic data, deep learning (DL) provides hierarchical feature extraction, facilitating timely network analysis and management. Thupae et al. [7] presented an SAE-based scheme for the classification of unencrypted data flows. However, this scheme only applies to unencrypted traffic data and cannot be applied to encrypted data. Lim et al. [8] introduced a method to classify encrypted traffic based on SAE and CNN techniques. Wang et al. [9] proposed three DL models using MLP, SAE, and CNN for traffic classification based on all encrypted streaming packets from open-source data. However, these models cannot be applied to real network traffic flows because they were performed on an offline dataset. Azzouni and Pujolle [10] proposed an LSTM-RNN framework for predicting traffic matrix (TM) in a large network. Azzouni et al. [11] introduced dynamic network routing based on LSTM to predict internet traffic with high accuracy. Azzouni and Pujolle [12] performed future network traffic assessments using LSTM, leveraging past and current network data. LSTM models exhibit more accurate long-range dependencies compared to RNN.

Zeng et al. [13] introduced a lightweight framework using DL for encrypted traffic classification and demonstrated its superiority. The proposed framework classifies network traffic based on the time features of the network traffic. Zhang et al. [14] performed automatic feature extraction from network traffic and classification of malicious traffic using CNN. In vehicular ad hoc networks, the SDN controller utilizes CNN to learn the highest routing path trust value. The CNN-enabled SDN controller provides trust-based optimized routing with a classification accuracy of 98.2%. CNN is also used for optimized feature selection using the CNN algorithm [15]. Tang et al. [16, 17] proposed a new DL algorithm to predict traffic and congestion in SDN. Deep belief and deep CNN were used, and the prediction algorithm was coupled with a DL-based channel assignment algorithm to route traffic. The gated recurrent unit (GRU) network is a DL model widely used in speech and image processing [18] and natural language processing [19]. It is well-suited for solving complex and nonlinear forecasting problems [20], such as traffic flow prediction [21], energy consumption prediction [22], and rainfall prediction. Tang et al. [16, 17] introduced a gated recurrent unit-recurrent neural network (GRU-RNN)-based intrusion detection system for SDN. The proposed system was tested with the NSL-KDD dataset and achieved an accuracy of 89% with only six raw features. It was concluded that the proposed GRU-RNN does not degrade network performance. Sun and Guan [23] proposed a traffic situation prediction model based on the GRU network in SDN. The Salp Swarm algorithm is used to optimize the hyperparameters of the GRU automatically. However, the GRU possesses problems such as a low convergence rate and low learning efficiency, resulting in excessively long training times, and even under-fitting [9].

The main problem addressed in this research is the inefficiency and lack of adaptability in current SDN routing algorithms, especially concerning dynamic traffic patterns. While algorithms like shortest path first (SPF) and Dijkstra's algorithm are efficient, they may lack adaptability and scalability in handling the dynamic nature of network traffic. Zhang et al. [24] introduced box-covering-based routing (BCR) for large-scale SDN to reduce the time and space complexity of the Dijkstra algorithm by decreasing the number of nodes and edges in the network. Although the BCR algorithm decreases the network's size, it still utilizes the Dijkstra algorithm in the routing process. This prompts the use of meta-heuristic techniques for the SDN's routing process.

The incorporation of meta-heuristic algorithms becomes crucial due to several shortcomings in traditional routing approaches. The proposed meta-heuristic approach aims to overcome these limitations by considering real-time congestion levels during routing, optimizing network performance, and ensuring a high Quality of Service (QoS). In summary, the role of meta-heuristic algorithms, exemplified by the novel hunter prey optimization (HPO) technique, becomes pivotal in optimizing routing decisions based on real-time congestion awareness.

The subsequent sections will delve into the methodology and simulation outcomes, showcasing the contributions of the proposed multiplicative gated recurrent neural network (mGRNN) for traffic prediction and the congestion-aware hunter prey optimization (CA-HPO) algorithm for dynamic traffic routing in SDN.

In this regard, the proposed research work develops two major units, namely, traffic prediction and traffic-aware routing unit. The following are the novel contributions of this research work.

- A novel multiplicative gated recurrent neural network (mGRNN) is developed for enhanced and accurate traffic prediction with excellent long-term dependencies. Consequently, mGRNN exhibits effective in handling temporal dynamics of the SDN traffic data including sudden changes, periodic fluctuations, and trends, as it can get adapted to the changes by learning from historical data.
- A novel congestion-aware hunter prey optimization (HPO) algorithm is developed for dynamic traffic routing. The HPO technique is influenced by the behavior of predatory animals such as lions, wolves, and leopards, as well as prey species such as stags and gazelles. The animal hunting behavior is used to find the shortest path between nodes. The advantages of meta-heuristic approach of HPO consider the current state of network congestion and seek to minimize congestion levels during traffic routing. This helps in optimizing the performance of the network and ensuring smooth transmission of data, thereby improving Quality of Service (QoS) and dynamic network adaptability.

The research paper is organized in a systematic structure with distinct sections at the end of Sect. 1 as follows. Section 2, titled "Literature review," incorporates case studies to underscore the adaptability of software-defined networking (SDN) in addressing diverse challenges. This section serves as a foundational exploration, providing context and insights that contribute to the subsequent sections. Section 3, titled "Methods/experimentation," provides a comprehensive account of the experimental setup and methodologies applied in the study. It details the data collection process, model configurations, and experimental procedures conducted to ensure the reliability of the results. Following this, Sect. 4, titled "Proposed methodology," introduces the novel framework developed for traffic engineering and congestion management in SDN. This section delves into the specifics of the multiplicative gated recurrent neural network (mGRNN) for traffic prediction and the congestion-aware hunter prey optimization (CA-HPO) algorithm for dynamic traffic routing. Section 5, titled "Results and discussion," presents the outcomes of the experiments and provides an in-depth analysis. This section critically evaluates the performance of the proposed methodologies, drawing meaningful insights from the obtained data. Lastly, Sect. 6, titled "Conclusion," encapsulates the key findings and summarizes the contributions. This organized structure guides the reader through a coherent progression from the foundational literature to the experimental methods, results, and concluding insights.

2 Literature review

In the dynamic intersection of software-defined networking (SDN) and the Internet of Things (IoT), researchers have made significant contributions addressing various challenges. Keshari et al. [25] specifically concentrate on software-defined IoT networks, proposing an intelligent and energy-efficient strategy to manage traffic flow. Their study underscores the necessity of tailoring SDN solutions for IoT, emphasizing the importance of energy-aware optimizations. Another crucial facet of SDN is explored by Taurshia et al. [26], who delve into group key management, particularly for resource-constrained IoT devices. The study highlights the challenge of securing communication in IoT environments with limited resources. Utilizing SDN capabilities, the authors propose a lightweight group key management solution, showcasing SDN's broader applicability in enhancing security measures for resource-constrained devices.

Contributing to the evolving landscape of SDN applications and IoT, Mohammadi et al. [27] propose an efficient clustering scheme. Their study introduces the SDN-IoT framework, incorporating an improved Sailfish optimization algorithm for enhanced efficiency. By integrating SDN capabilities with optimization algorithms, the authors aim to create an intelligent clustering scheme tailored for IoT environments, addressing the unique challenges posed by IoT scenarios. In order to provide a comprehensive overview of the current state of SDN traffic management research, Xu et al. [28] present a detailed survey. Through a thorough review of existing literature, the authors offer insights into various approaches, methodologies, and challenges within the realm of SDN traffic management. This survey serves as a valuable resource for understanding the current landscape of SDN research, laying the foundation for the development of novel solutions.

2.1 Synthesis and contributions

These case studies collectively contribute to the evolving field of software-defined networking, offering unique perspectives on key aspects such as traffic flow control, group key management, and efficient clustering in the context of IoT. The studies underscore the adaptability and versatility of SDN, showcasing its potential to address diverse challenges in modern network architectures. As this literature review forms part of a broader investigation into a computationally intelligent framework for traffic engineering and congestion management in SDN, these case studies provide valuable context and inspiration for developing innovative solutions tailored to the dynamic demands of contemporary networks.

3 Methods/experimentation

The aim of this study is to address the issue of congestion in SDN caused by a high volume of traffic in specific regions of the developed network topology. To achieve efficient traffic management in SDN, this research proposes a novel mGRNN and a CA-HPO algorithm as shown in Fig. 1. The SDN network is modeled with interconnected SDN controllers using the OpenFlow protocol to exchange control information. The method employs a simulation-based approach to evaluate the proposed SDN traffic prediction and routing optimization scheme. The simulations are performed using MATLAB R2021a and the image processing toolbox. The primary intervention in this research is the application of the proposed mGRNN and CA-HPO algorithm for traffic management in SDN. The mGRNN is used to predict traffic patterns and congestion-prone regions, while the HPO algorithm optimizes the routing decisions in the SDN to alleviate congestion. For comparison, two well-known optimization algorithms, namely genetic algorithm (GA) and particle swarm optimization (PSO), are used as baselines. The GA and PSO

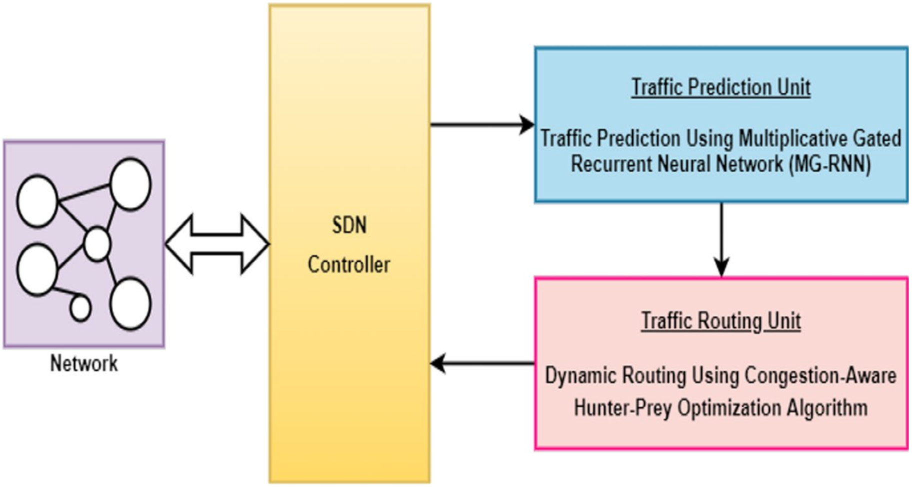

Network
SDN
Controller
Traffic Prediction Unit
Traffic Prediction Using Multiplicative Gated Recurrent Neural Network (MG-RNN)
Traffic Routing Unit
Dynamic Routing Using Congestion-Aware Hunter-Prey Optimization Algorithm

Fig. 1 Proposed congestion-aware SDN architecture. This figure illustrates the overall architecture of the proposed SDN traffic prediction and routing optimization. This architecture consists of two units: the traffic prediction unit and the traffic routing unit

algorithms are widely recognized in the literature for their effectiveness in optimization tasks. The work is validated to provide efficient results in congestion-aware routing.

3.1 Improvements

In the dynamic landscape of software-defined networking (SDN), effective traffic management is crucial for optimizing network performance. This study introduces several key improvements to enhance the reliability and applicability of the mGRNN (modified general regression neural network) and CA-HPO (combinatorial algorithm with hyperparameter optimization) algorithms. The objective is to advance the understanding and application of these algorithms in the context of SDN traffic management.

3.1.1 Summary of improvements

- *Sensitivity analysis* Conducting a sensitivity analysis on the parameters of both mGRNN and CA-HPO algorithms provides valuable insights into their behavior and impact on performance. This exploration allows for fine-tuning, ultimately improving the algorithms' adaptability to diverse network conditions.
- *Real-world dataset consideration* In our study, we aimed to enhance the external validity of our proposed approach by rigorously validating the mGRNN and CA-HPO algorithms using real-world datasets. To bridge the gap between simulated environments and practical network scenarios, we utilized two distinct SDN datasets available at Mendeley Data.

  The first dataset, curated by Wassie et al. [29], was employed in their research on "Traffic prediction in SDN for explainable QoS using deep learning approach." The second dataset, provided by Ahuja et al. [30], is specifically tailored for the study of DDoS attacks in the SDN context.

These datasets offer a diverse and comprehensive set of real-world scenarios, enabling us to thoroughly validate our algorithms in different traffic management contexts. For more details and access to the datasets, you can visit the following link: SDN datasets on Mendeley Data (https://data.mendeley.com/datasets/jxpfjc64kr/1).

- *Visualization* The incorporation of visualizations depicting predicted traffic patterns, congestion-prone regions, and routing decisions contributes to the clarity of results. Visual representations facilitate a more intuitive understanding of algorithmic outputs, aiding network administrators and stakeholders in making informed decisions.
- *Scalability assessment* Evaluating the scalability of the proposed approach involves varying the size and complexity of the SDN network. This assessment is crucial for determining the algorithms' efficiency as the network expands, ensuring their viability in handling large-scale and intricate SDN infrastructures.
- *Comparative analysis* A comprehensive comparative analysis with existing state-of-the-art methods for SDN traffic management provides a benchmark for assessing the proposed approach's efficacy. This analysis helps identify the strengths and weaknesses of the mGRNN and CA-HPO algorithms in comparison with established techniques.
- *Robustness testing* Robustness testing introduces variations in network conditions and traffic patterns to assess the adaptability of the proposed approach. By subjecting the algorithms to diverse and challenging scenarios, the study aims to validate their resilience and ability to maintain optimal performance in dynamic environments.

Incorporating these improvements collectively aims to advance the field of SDN traffic management, providing more nuanced insights into the capabilities and limitations of the mGRNN and CA-HPO algorithms in real-world scenarios.

4 Proposed methodology

Traditional networks have limitations in terms of function expansion and configuration. To enhance network management convenience, the development of software-defined networking (SDN) has been initiated. SDN is a new network architecture that separates network control and forwarding functions, simplifying network management and improving network programmability and flexibility. SDN enables better utilization of network resources, controls over network infrastructure expansion, and protects the underlying network complexity for upper-level users. It is continuously evolving based on traditional networks, significantly improving the utilization of network resources.

Network traffic exhibits variations in both time and space. In terms of time, network traffic varies throughout the day, with higher traffic during daytime compared to nighttime. SDN controllers are interconnected and exchange control information through the OpenFlow protocol. The SDN network is divided into application, control, and data layers. The control layer consists of one or more controllers that connect the data forwarding layer to the application layer through an interface. The control layer has centralized control over the network topology and can design approaches to manage data transmission paths. The data forwarding layer consists of switches, hosts, and other underlying network devices, which implement the data plane. The data layer does not have the capability to choose forwarding paths but relies on the control layer's provided paths for communication. Network flows are dynamic variables that change continuously in a fine-grained perspective but remain stable over time.

Although mitigation can improve network performance by supporting current burst network data flows and reducing overall network delay, it cannot guarantee improvement within a specific time frame. When the SDN network employs optimized routing configurations, especially during burst flows, the load flow may exceed the network capacity. To prevent congestion and achieve traffic control goals, suitable transmission resources must be allocated for data traffic based on transmission requirements. Additionally, restricting traffic flow from entering bottleneck links becomes essential. In this context, a dynamic and efficient traffic engineering (TE) scheme, such as SDN-based traffic prediction and routing optimization, has been developed. Figure 1 illustrates the overall architecture of the proposed SDN traffic prediction and routing optimization. This architecture consists of two units: the traffic prediction unit and the traffic routing unit.

4.1 SDN controller

SDN separates the data plane and the control plane, transferring network intelligence to the controller, where all calculations are performed, and various applications and features can be added as required. In this regard, a lightweight carrier-grade controller is proposed, focusing on essential modules. These modules include the link discovery module, topology module, storage module, strategy-making module, flow table module, and control data module. The topology manager and link discovery modules play a crucial role in providing routing services. The link discovery module is responsible for discovering and maintaining the state of the network's physical links. There are two methods of link discovery: link discovery between OpenFlow Nodes (OpenFlow switches) using the standard link layer discovery protocol (LLDP) and link discovery between edge OpenFlow Node and Host. When any unknown traffic enters the OpenFlow domain, the controller initiates the link discovery process. The information gathered by the link discovery module is used to build the neighbor database in the controller, which captures all the OpenFlow neighbors. As a result, the topology manager creates and maintains topology information in the controller, as well as calculating network paths. Based on the information obtained from the link discovery module, this module uses the neighbor database to construct network topologies. At the controller, the Topology Manager creates the global Topology Database, which contains information on the shortest (and alternate) paths to every OpenFlow node or host.

Our proposed methodology builds upon and extends prior work in traffic prediction, particularly leveraging concepts from the multiplicative recurrent neural network (mRNN) model [31]. Recognizing the strengths of both mRNN and gated recurrent unit (GRU) architectures, our model, termed mGRNN, represents a hybrid approach that combines these frameworks. This combination results in a novel architecture with improved expressiveness and adaptability. It extends the work of Lohrasbinasab et al. [32] by introducing a model that incorporates distinct recurrent transition functions and memory cells, enhancing the model's capability to capture intricate patterns in traffic data.

It complements the attention mechanism and spatiotemporal features utilized by Hu et al. [33] by introducing a hybrid architecture that integrates the strengths of mRNN and GRU, offering improved adaptability to varying network conditions. It extends the GRU-based predictive model proposed by Patil et al. [34] by combining the factorized hidden-to-hidden transition of mRNNs with the gating framework of GRUs, resulting in a more robust and expressive traffic prediction model. This novel methodology represents a significant advancement in the field of traffic prediction, offering a unique and effective solution for dynamic software-defined network (SDN) environments (Fig. 2).

4.2 Traffic prediction unit

To prevent congestion and enhance network performance, it is crucial to predict the future growth of network traffic. In this regard, a novel mGRNN is proposed for network traffic prediction. Network traffic prediction involves estimating future traffic based on past and current network traffic data. The current traffic matrix is estimated and sent as input to the traffic prediction unit. A network traffic matrix represents the amount of traffic between all pairs of source–destination nodes in a network at a specific time. The nodes within a traffic matrix can comprise of Points of Presence, routers, switches, or links. In OpenFlow SDNs, the controller makes use of packet in messages to construct a comprehensive view of the network. When a new flow arrives at a switch, it is compared against forwarding rules to estimate the appropriate forwarding path. If the flow does not match any rule, the switch forward either the first packet or just the packet header to the SDN controller. Furthermore, the SDN controller can request packet counts from switches, which track the count of packets and bytes processed by each switch.

In the proposed approach, the current and past traffic matrices are used as input to the mGRNN model to forecast the traffic matrix for the next state. The network operates by allowing information to flow both forward and backward, traversing input nodes, hidden nodes, and output nodes through recurrent cycles. Furthermore, the proposed model

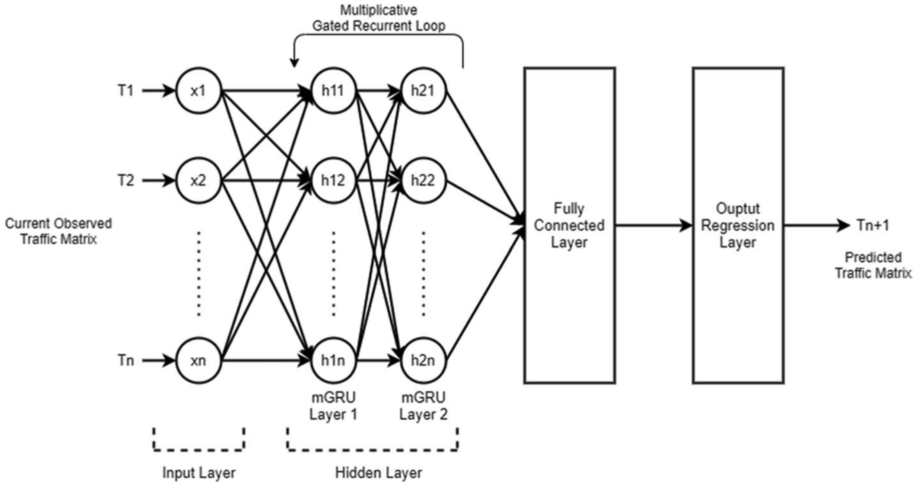


Fig. 2 Proposed traffic prediction network. This figure illustrates the functioning of the proposed mGRNN-based traffic prediction network

incorporates distinct recurrent transition functions for each possible input, enhancing its expressiveness. Additionally, the model includes memory cells that utilize past information during the learning process. The effectiveness of the prediction relies on the quality of the learning process. These unique characteristics differentiate the proposed mGRNN traffic prediction model and position it as superior to existing traffic prediction models.

The employment of mGRNN for the traffic prediction involves two distinct phases: (a) the training phase and (b) the testing phase. During the training phase, the mGRNN is supervised and learns from the data by obtaining the training data at the input layer. The mGRNN dynamically adjusts its parameters to attain the desired output value for the given input set. The backpropagation algorithm is employed to train the mGRNN. This algorithm propagates the error backward, from the output layer to the input layer, uninterruptedly modifying the weights until the output error reaches a predefined threshold. Consequently, the mGRNN learns to identify patterns between input sets and their corresponding target values. On the other hand, the testing phase involves testing the mGRNN. A new, unseen traffic matrix input is presented to the mGRNN, and the output predicted next state traffic matrix is calculated, enabling the prediction of outcomes for novel input data.

The concept of mGRNN is inspired by the multiplicative recurrent neural network (mRNN) model [31]. The model combines the advantages of mRNN and gated recurrent unit (GRU) cell model. The mRNN model is specifically developed to enable adaptable transitions based on input changes. Recognizing the complementary characteristics of the GRU and mRNN architectures, a hybrid model called mGRNN is introduce, which combines the factorized hidden-to-hidden transition of mRNNs with the gating framework of GRUs. By incorporating connections from the mRNN's intermediate state m_t to each gating unit in the GRU, a system is created that merges the strengths of both architectures as follows.

$$m_t = W_{mx}x_t \odot W_{mh}h_{t-1} \tag{1}$$

$$r_t = \sigma(W_{rx} * [h_{t-1}, x_t] + W_{rm}m_t) \tag{2}$$

$$z_t = \sigma(W_{zx} * [h_{t-1}, x_t] + W_{zm}m_t) \tag{3}$$

$$h_{t'} = \tanh(W_h * [r_t * h_{t-1}, x_t]) \tag{4}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t' \tag{5}$$

The objective of this architecture is to integrate the adaptable input-dependent transitions found in mRNNs with the ability of GRUs to retain and utilize information over longer sequences. By leveraging the gated units of GRUs, it becomes more manageable to regulate or bypass the intricate transitions that arise from the factorized hidden weight matrix. The model thus captures complex dependencies and retains important information contributing to enhanced accuracy and robustness in the performance of traffic prediction (Fig. 3).

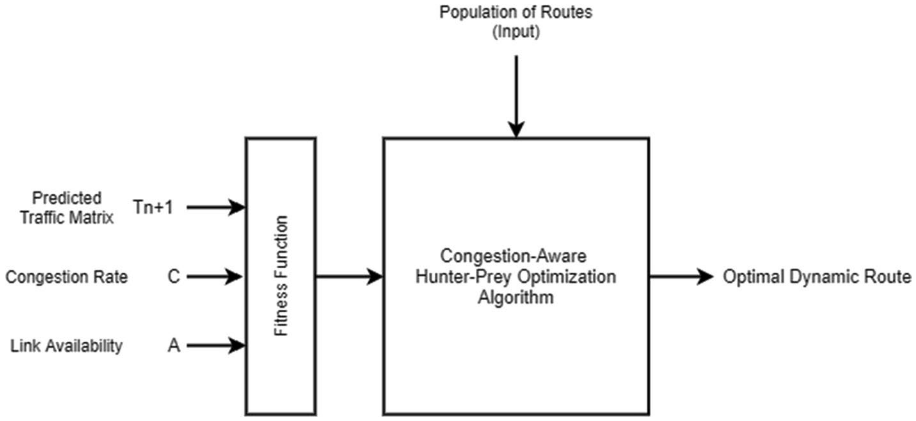


Fig. 3 Proposed traffic routing unit. This figure illustrates the functioning of the proposed CA-HPO-based traffic routing unit

4.3 Traffic routing unit

To meet the required QoS, the traffic flow should be routed following the best routing procedure. As the traffic flow relies upon the type of data transported in the network, the selection of the best routing path to that flow improves the QoS. Accordingly, the congestion-aware hunter prey optimization (HPO) algorithm was proposed that dynamically determines the optimal path. The input given in the traffic routing unit is the possible set of routes. The output optimal dynamic route is calculated based on three factors such as the predicted traffic matrix, congestion rate, and link availability of the route chosen to provide an optimal congestion-aware route for data transmission.

Initially, the population of the search agents are the possible set of routes which are defined as $X = \{x_1, x_2, \ldots, x_n\}$, and the fitness values of each member of the population are denoted as $F = \left\{f_1, f_2, \ldots, f_n\right\}$. The movement and direction of the population within the search space are controlled and directed by a set of rules and strategies inspired by the HPO algorithm. This process continues iteratively until maximum search iteration is reached. In each iteration, the position of each population member is updated according to the algorithm's rules, and the newly determined position is evaluated using the objective function. As a result, the solutions gradually improve with each iteration.

$$f_i = A_i + \frac{1}{C_i} + \frac{1}{|T_{i+1}|} \tag{6}$$

where A_i is the sum of availabilities of links of the route i (link availability 0 indicates that the link is unavailable and 1 indicates that the link is available), C_i is the congestion rate of the route, and T_{i+1} is the traffic matrix predicted by the traffic prediction unit.

The search mechanism typically involves two major phases: exploration and exploitation. Exploration refers to the algorithm's inclination toward highly random behaviors, resulting in significant changes in the solutions. These changes facilitate further exploration of the search space, supporting in the process of discovering promising areas. Once promising regions are determined, random behaviors are reduced to focus the algorithm's search around these favorable regions, which is called as exploitation. For the hunter search mechanism, the equation governing the update of the hunter's position is described as follows.

$$x_i(t+1) = x_i(t) + 0.5\left[2CZP_{\mathrm{pos}(i)} - x_i(t) + \left(2(1-C)Z\mu_{(i)} - x_i(t)\right)\right] \tag{7}$$

where $x(t)$ represents the current hunter position, $x(t+1)$ represents the hunter next position, P_{pos} represents the prey position, μ represents the mean of all positions, and Z is an adaptive parameter computed using the following equation.

$$Z = R_1 \otimes idx + \vec{R}_2 \otimes (\sim idx) \tag{8}$$

where R_1 is a random number in the range [0,1] and $\vec{R}_2$ is a random vector in the range [0,1], idx is the index numbers of the random vector $\vec{R}_3$ which is also in the range [0,1] and satisfies the condition ($P{=}{=}0$). P is a random vector with values 0 and 1 equal to the number of problem variables.

The balance parameter C computes the trade-off between exploration and exploitation. Its value gradually decrements from 1 to 0.02 as the iterations progress. The value of C is calculated using the Eq. (9).

$$C = 1 - \mathrm{it}\left(\frac{0.98}{\mathrm{it}_{\max}}\right) \tag{9}$$

where it denotes the current iteration and $\mathrm{it}_{\max}$ denotes the maximum number of iterations.

P_{pos} is then calculated using the Euclidean distance D as follows.

$$P_{\mathrm{pos}} = x_i \middle| i \text{ is the index of Max(end)sort}(D) \tag{10}$$

where

$$D = \left(\sum_{j=1}^{d} (x_i - \mu_i)^2\right)^{\frac{1}{2}} \tag{11}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{12}$$

The update equation of the prey position for each iteration is given by,

$$P_{\mathrm{pos}} = x_i \middle| i \text{ is sorted } D(kbest) \tag{13}$$

where $kbest = \mathrm{round}(C \times N)$, N represents the total number of search agents.

When the prey is attacked, it instinctively attempts to escape and reach a safe spot. It is assumed that the optimal global position represents the best safe position for the prey, as it provides the highest chance of survival and potentially allows the hunter to pursue another prey. The following equation represents the final update of the prey's position.

$$x_i(t+1) = G_{\mathrm{pos}(i)} + CZ\cos(2\pi R_4) \times \left(G_{\mathrm{pos}(i)} - x_i(t)\right) \tag{14}$$

where G_{pos} represents the optimal global position and R_4 is a random number within the range of -1 to 1. Thus the final optimal dynamic route determined by the proposed congestion-aware HPO-based traffic routing algorithm is G_{pos}.

5 Results and discussion

The experiment was performed in MATLAB R2021 using the proposed approach and image processing toolbox. The proposed model was trained on a Windows 10 system with an Intel(R) Core$^{\text{TM}}$ i7-8650U processor, 16 GB of random access memory (RAM), and an NVIDIA GeForce MX150 graphics processing unit (GPU).

5.1 SDN deployment

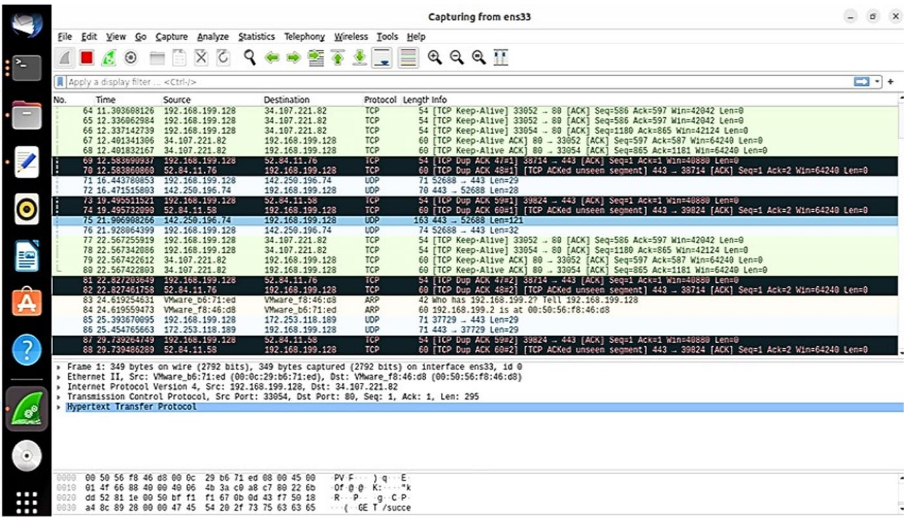
The data packets are captured using the hypertext transfer protocol in the Wireshark platform. The transmission of packets from one node to another is captured using the interface. The information on data transmission and traffic during routing was saved as output csv file format. The traffic files of all the nodes in the network for a specific period of time are also recorded and saved. These files are then utilized for further processing of traffic prediction and optimal route selection in the framework. The set of the traffic prediction was done during the data transmission and saved as shown in Fig. 4.

5.2 Performance measures

The performance of the proposed framework was assessed using various performance metrics such as mean absolute error (MAE), mean square error (MSE), and root-mean-square error (RMSE).

5.2.1 Mean absolute error (MAE)

Absolute error is the difference between the predicted observation (Data) and the actual values of that observation. MAE takes the average of absolute errors for a group of



```
No.   Time          Source            Destination       Protocol Length Info
   64 11.303608126  192.168.199.128   34.107.221.82     TCP        54 [TCP Keep-Alive] 33052 → 80 [ACK] Seq=586 Ack=597 Win=42042 Len=0
   65 12.336062984  192.168.199.128   34.107.221.82     TCP        54 [TCP Keep-Alive] 33052 → 80 [ACK] Seq=586 Ack=597 Win=42042 Len=0
   66 12.337142739  192.168.199.128   34.107.221.82     TCP        54 [TCP Keep-Alive] 33054 → 80 [ACK] Seq=1180 Ack=865 Win=42124 Len=0
   67 12.401341306  34.107.221.82     192.168.199.128   TCP        60 [TCP Keep-Alive ACK] 80 → 33052 [ACK] Seq=597 Ack=587 Win=64240 Len=0
   68 12.401832167  34.107.221.82     192.168.199.128   TCP        60 [TCP Keep-Alive ACK] 80 → 33054 [ACK] Seq=865 Ack=1181 Win=64240 Len=0
   69 12.583690937  192.168.199.128   52.84.11.76       TCP        54 [TCP Dup ACK 47#1] 38714 → 443 [ACK] Seq=1 Ack=1 Win=40880 Len=0
   70 12.583860860  52.84.11.76       192.168.199.128   TCP        60 [TCP Dup ACK 48#1] [TCP ACKed unseen segment] 443 → 38714 [ACK] Seq=1 Ack=2 Win=64240 Len=0
   71 16.443780853  192.168.199.128   142.250.196.74    UDP        71 52688 → 443 Len=29
   72 16.471515803  142.250.196.74    192.168.199.128   UDP        70 443 → 52688 Len=28
   73 19.495511521  192.168.199.128   52.84.11.58       TCP        54 [TCP Dup ACK 59#1] 39824 → 443 [ACK] Seq=1 Ack=1 Win=40880 Len=0
   74 19.495732090  52.84.11.58       192.168.199.128   TCP        60 [TCP Dup ACK 60#1] [TCP ACKed unseen segment] 443 → 39824 [ACK] Seq=1 Ack=2 Win=64240 Len=0
   75 21.906908266  142.250.196.74    192.168.199.128   UDP       163 443 → 52688 Len=121
   76 21.928064399  192.168.199.128   142.250.196.74    UDP        74 52688 → 443 Len=32
   77 22.567255919  192.168.199.128   34.107.221.82     TCP        54 [TCP Keep-Alive] 33052 → 80 [ACK] Seq=586 Ack=597 Win=42042 Len=0
   78 22.567342086  192.168.199.128   34.107.221.82     TCP        54 [TCP Keep-Alive] 33054 → 80 [ACK] Seq=1180 Ack=865 Win=42124 Len=0
   79 22.567422612  34.107.221.82     192.168.199.128   TCP        60 [TCP Keep-Alive ACK] 80 → 33052 [ACK] Seq=597 Ack=587 Win=64240 Len=0
   80 22.567422803  34.107.221.82     192.168.199.128   TCP        60 [TCP Keep-Alive ACK] 80 → 33054 [ACK] Seq=865 Ack=1181 Win=64240 Len=0
   81 22.827203649  192.168.199.128   52.84.11.76       TCP        54 [TCP Dup ACK 47#2] 38714 → 443 [ACK] Seq=1 Ack=1 Win=40880 Len=0
   82 22.827461758  52.84.11.76       192.168.199.128   TCP        60 [TCP Dup ACK 48#2] [TCP ACKed unseen segment] 443 → 38714 [ACK] Seq=1 Ack=2 Win=64240 Len=0
   83 24.619254631  VMware_b6:71:ed   VMware_f8:46:d8   ARP        42 Who has 192.168.199.2? Tell 192.168.199.128
   84 24.619559473  VMware_f8:46:d8   VMware_b6:71:ed   ARP        60 192.168.199.2 is at 00:50:56:f8:46:d8
   85 25.393670095  192.168.199.128   172.253.118.189   UDP        71 37729 → 443 Len=29
   86 25.454765663  172.253.118.189   192.168.199.128   UDP        71 443 → 37729 Len=29
   87 29.739264749  192.168.199.128   52.84.11.58       TCP        54 [TCP Dup ACK 59#2] 39824 → 443 [ACK] Seq=1 Ack=1 Win=40880 Len=0
   88 29.739486289  52.84.11.58       192.168.199.128   TCP        60 [TCP Dup ACK 60#2] [TCP ACKed unseen segment] 443 → 39824 [ACK] Seq=1 Ack=2 Win=64240 Len=0

▸ Frame 1: 349 bytes on wire (2792 bits), 349 bytes captured (2792 bits) on interface ens33, id 0
▸ Ethernet II, Src: VMware_b6:71:ed (00:0c:29:b6:71:ed), Dst: VMware_f8:46:d8 (00:50:56:f8:46:d8)
▸ Internet Protocol Version 4, Src: 192.168.199.128, Dst: 34.107.221.82
▸ Transmission Control Protocol, Src Port: 33054, Dst Port: 80, Seq: 1, Ack: 1, Len: 295
▸ Hypertext Transfer Protocol

0000  00 50 56 f8 46 d8 00 0c  29 b6 71 ed 08 00 45 00   PV F    ) q   E
0010  01 4f 66 88 40 00 40 06  4b 3a c0 a8 c7 80 22 6b   Of @ @  K:    "k
0020  dd 52 81 1e 00 50 bf f1  f1 67 0b 0d 43 f7 50 18   R   P    g  C P
0030  a4 8c 89 28 00 00 47 45  54 20 2f 73 75 63 63 65      (  GE T /succe
```

Fig. 4 SDN deployment. This figure shows the SDN deployment, where the data packets are captured using the hypertext transfer protocol in the Wireshark platform. This shows the set of the traffic prediction done during the data transmission

predictions and observations as a measurement of the magnitude of errors for the entire group. The expression for MAE was given as follows;

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} \left|\hat{x}_i - x_i\right|}{n} \tag{15}$$

5.2.2 Mean square error (MSE)

MSE is the simplest and most commonly used to calculate the loss function. The loss function is a method of evaluating how well the algorithm models the dataset. A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite. The expression for MSE was given as follows:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n} \left(\hat{x}_i - x_i\right)^2}{n} \tag{16}$$

5.2.3 Root-mean-square error (RMSE)

RMSE is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point. The expression for RMSE and RRMSE was given as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(\hat{x}_i - x_i\right)^2}{n}} \tag{17}$$

$$\mathrm{RRMSE} = \sqrt{\frac{\left(\frac{\sum_{i=1}^{n} \left(\hat{x}_i - x_i\right)^2}{n}\right)}{\sum_{i=1}^{n} \left(\hat{x}_i\right)^2}} \tag{18}$$

where $\hat{x}_i$ is the predicted observations; $\hat{x}_i$ is the true or actual values; n is the total number of observations.

5.3 Performance evaluation

5.3.1 Traffic prediction

The performance of the proposed traffic prediction model was compared with the existing methods such as SVM, KNN, DT, ANN, and RNN. Table 1 presents the comparisons of values of MAE, MSE, and RMSE of the proposed and existing method.

Figure 5 graphically shows the MAE value of the proposed and existing models. It was seen that the MAE value of the proposed model was less as compared to the existing model such as SVM, KNN, DT, ANN, and RNN. The MAE value of the proposed model was 4.2–91% lesser than the existing models.

Figure 6 graphically shows the MSE value of the proposed and existing methods. The proposed mGRNN model shows the least MSE value of 0.00905. The MSE value

Table 1 Performance of proposed and existing methods

Models	MAE	MSE	RMSE	RRMSE
Decision tree	0.00792	0.039437	0.07708	0.065079
KNN	0.00453	0.030577	0.03946	0.0852192
SVM	0.00482	0.061588	0.058821	0.072883
ANN	0.08065	0.01002	0.12840	0.05929
RNN	0.08766	0.01468	0.12623	0.06296
Proposed MG-RNN model	0.0026	0.00905	0.029	0.0127

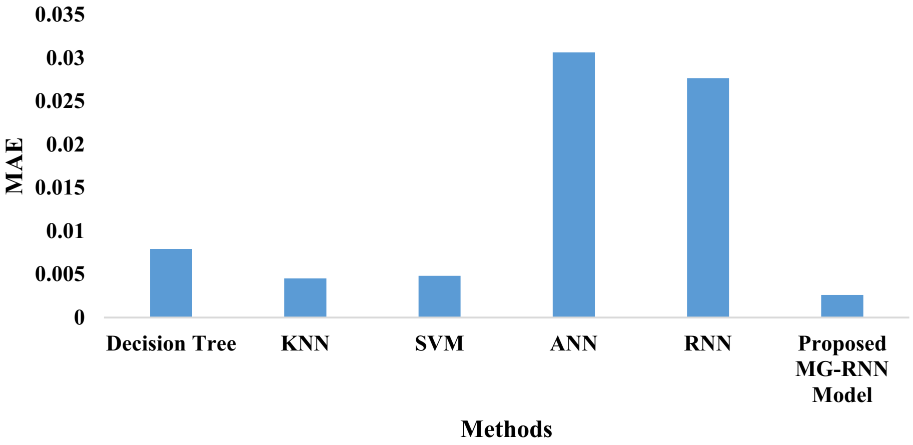


Fig. 5 MAE value of the proposed and existing methods. This figure graphically shows the MAE value of the proposed and existing models. It was seen that the MAE value of the proposed model was less as compared to the existing model such as SVM, KNN, DT, ANN, and RNN. The MAE value of the proposed model was 4.2–91% lesser than the existing models

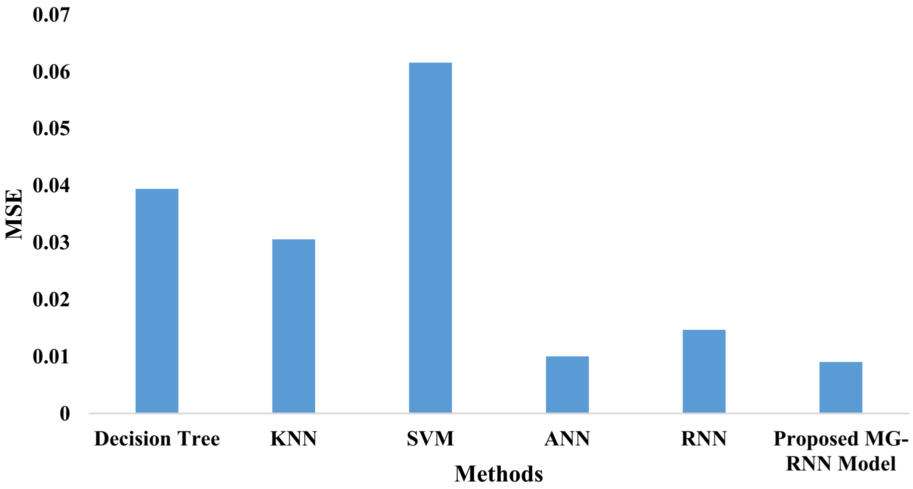


Fig. 6 MSE value of the proposed and existing methods. This figure graphically shows the MSE value of the proposed and existing methods. The proposed mGRNN model shows the least MSE value of 0.00905. The MSE value of the proposed model was 9.6–85.3% lesser than the existing model such as SVM, KNN, DT, ANN, and RNN

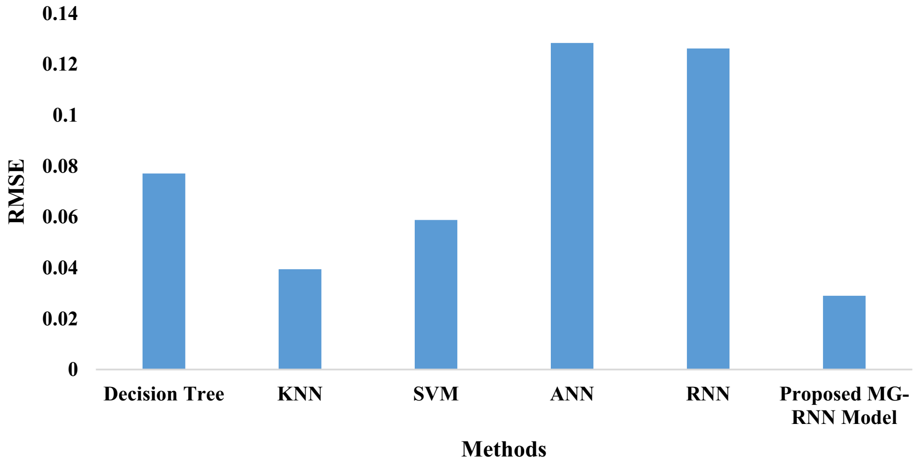


Fig. 7 RMSE value of the proposed and existing methods. This figure graphically shows the RMSE value of the proposed and existing models. It was seen that the RMSE value of the proposed model was less as compared to the existing models. The RMSE value of the proposed model was found to be 26.5–77.4% lesser than the existing models such as SVM, KNN, DT, ANN, and RNN

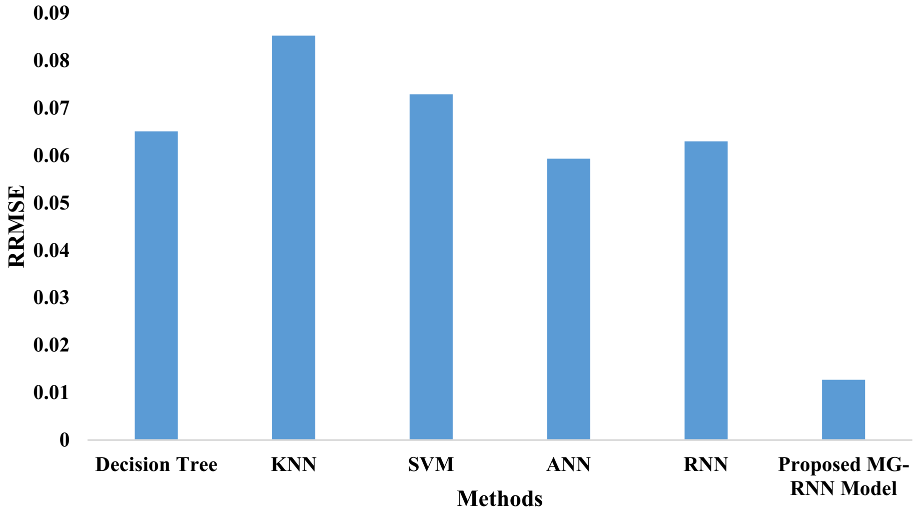


Fig. 8 RRMSE value of the proposed and existing methods. This figure graphically shows the RRMSE value of the proposed and existing models. It was seen that the RRMSE value of the proposed model was less as compared to the existing models. The RRMSE value of the proposed model was found to be 7.85–85.7% lesser than the existing models such as SVM, KNN, DT, ANN, and RNN

of the proposed model was 9.6–85.3% lesser than the existing model such as SVM, KNN, DT, ANN, and RNN.

Figure 7 graphically shows the RMSE value of the proposed and existing models. It was seen that the RMSE value of the proposed model was less as compared to the existing models. The RMSE value of the proposed model was found to be 26.5–77.4% lesser than the existing models such as SVM, KNN, DT, ANN, and RNN.

Figure 8 graphically shows the RRMSE value of the proposed and existing models. It was seen that the RRMSE value of the proposed model was less as compared to the existing models. The RRMSE value of the proposed model was found to be 7.85–85.7% lesser than the existing models such as SVM, KNN, DT, ANN, and RNN.

5.3.2 Comparison with other machine learning algorithms

In our rigorous evaluation, the proposed mGRNN model has been systematically compared with other machine learning algorithms commonly used in the context of SDN traffic prediction. The comparison extends to algorithms such as long short-term memory (LSTM), gated recurrent unit (GRU), and traditional machine learning algorithms.

The key aspects of this comparative analysis include:

- Accuracy: The mGRNN model exhibits competitive or superior accuracy compared to alternative machine learning algorithms. This is particularly significant in SDN environments where precise traffic prediction is essential for efficient network management.
- Training efficiency: The mGRNN model demonstrates efficient training, benefiting from the back propagation algorithm that continuously refines weights until a predefined error threshold is reached. This contributes to quicker convergence during the training phase.
- Generalization capability: The mGRNN model showcases strong generalization capabilities, effectively learning patterns and relationships in input sets. This is crucial for the accurate prediction of outcomes for novel input data during the testing phase.

5.3.3 Traffic routing

The performance of the proposed traffic routing model was compared with the existing GA and PSO models in terms of routing overhead (packets), Normal throughput, and average delay.

The normalized throughput of the network can be defined as:

$$\text{Normalized throughput} = \frac{N_{\mathrm{C}}}{N_{\mathrm{T}}} \tag{19}$$

where N_{C} is the number of data packets correctly received, and N_{T} is the total number of data packets sent.

The average delay can be defined as:

$$\text{Average delay} = \text{mean}\left(\frac{M \times L}{T_{\mathrm{r}}}\right) \tag{20}$$

where M, is the number of travelled links, L is the length of the data packet and T_{r} is the rate of data transmission.

Figure 9 presents the routing overhead of the various methods such as CA-HPO, GA, and PSO against various numbers of nodes. It was seen that the routing overhead decreases with the increase in number of nodes. The routing overhead of the proposed model was very less than the GA and PSO methods. The proposed CA-HPO

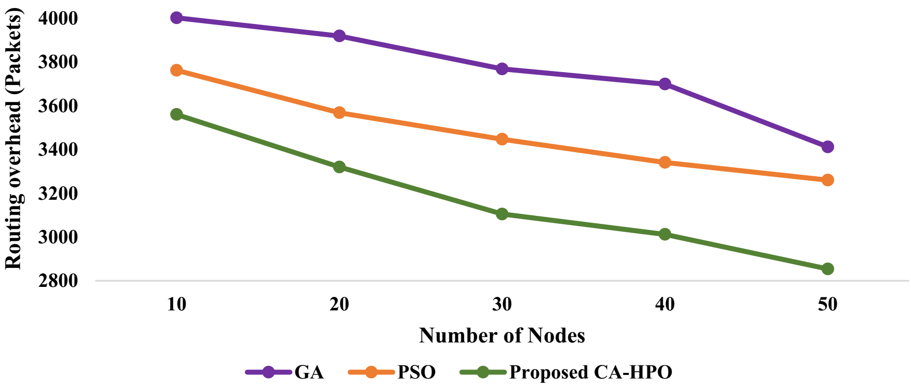


Fig. 9 Routing overhead of proposed and existing methods. This figure presents the routing overhead of the various methods such as CA-HPO, GA, and PSO against various numbers of nodes. It was seen that the routing overhead decreases with the increase in number of nodes. The routing overhead of the proposed model was very less than the GA and PSO methods. The proposed CA-HPO accomplished a lower routing overhead of 2854 at 50 nodes, were the routing overhead of the existing models such as GA and PSO, achieved 3412 and 3260, respectively

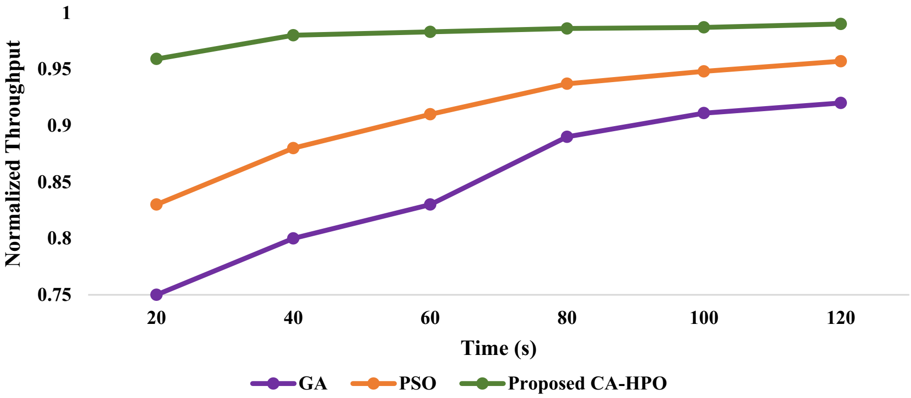


Fig. 10 Normalized throughout of the proposed and existing methods. This figure presents the normalized throughout of the various methods such as CA-HPO, GA, and PSO against various simulation times. It was observed that the normalized throughout of the proposed CAHPO algorithm was 3.4–7.6% higher than the normalized throughout of GA and PSO algorithms. The proposed CA-HPO algorithm showed a high speed in searching the optimized routing path with minimum RMSE

accomplished a lower routing overhead of 2854 at 50 nodes, where the routing overhead of the existing models, such as GA and PSO, achieved 3412 and 3260, respectively.

5.3.4 System performance

Figure 10 presents the normalized throughout of the various methods such as CA-HPO, GA, and PSO against various simulation times. The normalized throughput was increased with the increase in simulation time, which demonstrates that the network was enhanced. The normalized throughout of the proposed CA-HPO was 0.99 packets/seconds, whereas the normalized throughout of the GA and PSO was 0.99 packets/seconds and 0.99 packets/seconds. It was observed that the normalized throughout of the proposed CA-HPO algorithm was 3.4–7.6% higher than the normalized throughout of GA and PSO algorithms. The proposed CA-HPO algorithm showed a high speed in searching the optimized routing path with minimum RMSE. In the HPO technique, which includes prey and predator populations, a predator attacks the victim who wanders far from their group. The animals can place themselves farther from danger as the hunter moves closer to the distant target. It was assumed that the search agent's station has a great fitness value and was a secure area. With the inspiration of the animal hunting behavior, the HPO algorithm can search the optimized routing path.

Figure 11 presents the variation of average delay against the variation in simulation time of existing and proposed methods. The average delay of the proposed CA-HPO method was 58–65% lesser than GA and PSO methods. It was observed that average delay in data transmission was increased with increase in simulation time. This shows that the QoS of the network was enhanced.

5.3.5 Comparison with PSO and GA algorithms

In addition to the comparison with traditional machine learning algorithms, our research extends the evaluation to include particle swarm optimization (PSO) and genetic algorithms (GA).

The aspects under consideration in this comparison are:

- Optimization performance: The mGRNN model is evaluated against PSO and GA in terms of its optimization performance. This involves assessing how well the model adapts to changing conditions and refines its parameters to optimize traffic prediction accuracy.
- Convergence rate: The speed at which the mGRNN model converges to an optimal solution is compared with PSO and GA. This is crucial for determining the efficiency of each algorithm in finding the optimal configuration for traffic prediction.
- Robustness: The robustness of the mGRNN model in handling variations and uncertainties in SDN traffic data is compared with PSO and GA. This assessment

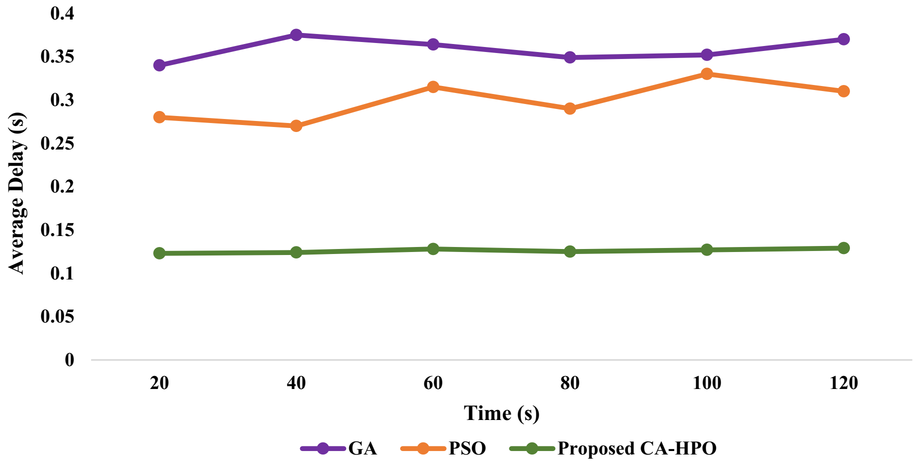


Fig. 11 Average delay of the proposed and existing methods. This figure presents the variation of average delay against the variation in simulation time of existing and proposed methods. The average delay of the proposed CA-HPO method was 58–65% lesser than GA and PSO methods. It was observed that average delay in data transmission was increased with increase in simulation time. This shows that the QoS of the network was enhanced

considers the ability of each algorithm to maintain prediction accuracy under different network scenarios.

In summary, the proposed mGRNN model stands out as a novel and effective approach for traffic prediction in SDNs, offering unique features that contribute to its accuracy and adaptability. The thorough comparison with other machine learning algorithms, as well as PSO and GA, provides a comprehensive understanding of the model's strengths and advantages in the context of our research objectives.

6 Conclusions

The evolution of AI and ML in route optimization for the SDN can be viewed as a decision-making policy that does not require a complex mathematical model when the model has been trained. The high computation cost would be the shortcoming of using meta-heuristic algorithms such as the ant colony optimization (ACO) algorithm, etc., for optimal routing in SDN. In comparison with various research articles such as Albakri et al. [35] and Thenmozhi et al. [36], the authors of this study have not conducted a comprehensive comparison of performance measures, including mean absolute error (MAE), mean squared error (MSE), root-mean-squared error (RMSE), and relative root-mean-squared error (RRMSE). The model proposed in this article demonstrates a significant enhancement in SDN traffic management with a lower error rate.

In this paper, a novel multiplicative gated recurrent neural network (mGRNN) and congestion-aware hunter prey optimization (HPO) algorithm was utilized for effective traffic prediction and routing for efficient traffic management in the SDN. It was seen that the proposed mGRNN can effectively predict the traffic of the SDN. The MAE, MSE, RMSE, and RRMSE of the proposed model were found to be 4.2–91%, 9.6–85.3%, 26.5–77.4%, and 7.85–85.7% lesser than the existing models like SVM, KNN, DT, ANN, RNN. The performance of the CA-HPO algorithm is also better in searching for optimal routing with improved QoS of the network.

Abbreviations

SDN	Software-defined networking
ML	Machine learning
DL	Deep learning
mGRNN	Multiplicative gated recurrent neural network
HPO	Hunter prey optimization
CA-HPO	Congestion-aware hunter prey optimization
GA	Genetic algorithm
PSO	Particle swarm optimization
TE	Traffic engineering
SAE	Sparse autoencoder
CNN	Convolutional neural network
LSTM-RNN	Long short-term memory-recurrent neural network
TM	Traffic matrix
GRU	Gated recurrent unit
SPF	Shortest path first
BCR	Box-covering-based routing
QoS	Quality of Service
LLDP	Link layer discovery protocol
RAM	Random access memory
GPU	Graphics processing unit
MAE	Mean absolute error
MSE	Mean square error
RMSE	Root-mean-square error
RRMSE	Relative root-mean-square error

SVM	Support vector machine
KNN	K-nearest neighbor
DT	Decision tree
ANN	Artificial neural network
RNN	Recurrent neural network

Acknowledgements
The authors would like to acknowledge the anonymous reviewers for their thoughtful comments.

Author contributions
L. Leo Prasanth designed the algorithm, performed the simulation results, and drafted the manuscript under the supervision of Dr. E. Uma. All authors read and approved the final manuscript.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials
The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 7 August 2023 Accepted: 7 August 2024
Published online: 19 August 2024

References

1. B.A.A. Nunes, M. Mendonca, X.N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
2. F. Hu, Q. Hao, K. Bao, A survey on software-defined network and openflow: from concept to implementation. IEEE Commun. Surv. Tutor. **16**(4), 2181–2206 (2014)
3. W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, A survey on software-defined networking. IEEE Commun. Surv. Tutor. **17**(1), 27–51 (2014)
4. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, J. Turner, OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
5. T. Mahboob, Y.R. Jung, M.Y. Chung, Optimized routing in software defined networks–a reinforcement learning approach. In *Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019 13* (Springer, 2019), pp. 267–278
6. M. Perera, K. Piamrat, S. Hamma, Network traffic classification using machine learning for software defined networks. In *Journées Non Thématiques GDR-RSD 2020* (2020)
7. R. Thupae, B. Isong, N. Gasela, A.M. Abu-Mahfouz, Machine learning techniques for traffic identification and classifiacation in SDWSN: a survey. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society* (IEEE, 2018), pp. 4645–4650
8. H.K. Lim, J.B. Kim, J.S. Heo, K. Kim, Y.G. Hong, Y.H. Han, Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. (IEEE, 2019), pp. 046–051
9. P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: deep learning based encrypted network traffic classification in SDN home gateway. IEEE Access **6**, 55380–55391 (2018)
10. A. Azzouni, G. Pujolle, A long short-term memory recurrent neural network framework for network traffic matrix prediction (2017). arXiv preprint https://arxiv.org/abs/1705.05690
11. A. Azzouni, R. Boutaba, G. Pujolle, NeuRoute: predictive dynamic routing for software-defined networks. In *2017 13th International Conference on Network and Service Management (CNSM)* (IEEE, 2017), pp. 1–6
12. A. Azzouni, G. Pujolle, NeuTM: a neural network-based framework for traffic matrix prediction in SDN. In *NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium* (IEEE, 2018), pp. 1–5
13. Y. Zeng, H. Gu, W. Wei, Y. Guo, Deep-full-range: a deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access **7**, 45182–45190 (2019)
14. D. Zhang, F.R. Yu, R. Yang, A machine learning approach for software-defined vehicular ad hoc networks with trust management. In *2018 IEEE Global Communications Conference (GLOBECOM)* (IEEE, 2018), pp. 1–6
15. D. Arivudainambi, V.K. Ka, S. Sibi Chakkaravarthy, LION IDS: a meta-heuristics approach to detect DDoS attacks against software-defined networks. Neural Comput. Appl. **31**, 1491–1501 (2019)
16. F. Tang, Z.M. Fadlullah, B. Mao, N. Kato, An intelligent traffic load prediction-based adaptive channel assignment algorithm in SDN-IoT: a deep learning approach. IEEE Internet Things J. **5**(6), 5141–5154 (2018)
17. T.A. Tang, L. Mhamdi, D. McLernon, S.A.R. Zaidi, M. Ghogho, Deep recurrent neural network for intrusion detection in SDN-based networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)* (IEEE, 2018), pp. 202–206

18. Y. Yuan, C. Tian, X. Lu, Auxiliary loss multimodal GRU model in audio-visual speech recognition. IEEE Access **6**, 5573–5583 (2018)
19. L. Mou, P. Ghamisi, X.X. Zhu, Deep recurrent neural networks for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **55**(7), 3639–3655 (2017)
20. W. Huang, Y. Li, Y. Huang, Deep hybrid neural network and improved differential neuroevolution for chaotic time series prediction. IEEE Access **8**, 159552–159565 (2020)
21. P. Sun, A. Boukerche, Y. Tao, SSGRU: a novel hybrid stacked GRU-based traffic volume prediction approach in a road network. Comput. Commun. **160**, 502–511 (2020)
22. T.Y. Kim, S.B. Cho, Predicting residential energy consumption using CNN-LSTM neural networks. Energy **182**, 72–81 (2019)
23. W. Sun, S. Guan, A GRU-based traffic situation prediction method in multi-domain software defined network. PeerJ Comput. Sci. **8**, e1011 (2022)
24. L. Zhang, Q. Deng, Y. Su, Y. Hu, A box-covering-based routing algorithm for large-scale SDNs. IEEE Access **5**, 4048–4056 (2017)
25. S.K. Keshari, V. Kansal, S. Kumar, P. Bansal, An intelligent energy efficient optimized approach to control the traffic flow in software-defined IoT networks. Sustain. Energy Technol. Assess. **55**, 102952 (2023)
26. A. Taurshia, G.J.W. Kathrine, A. Souri, S.E. Vinodh, S. Vimal, K.C. Li, S.S. Ilango, Software-defined network aided lightweight group key management for resource-constrained Internet of Things devices. Sustain. Comput. Inform. Syst. **36**, 100807 (2022)
27. R. Mohammadi, S. Akleylek, A. Ghaffari, SDN-IoT: SDN-based efficient clustering scheme for IoT using improved Sailfish optimization algorithm. PeerJ Comput. Sci. **9**, e1424 (2023)
28. C. Xu, D. Qin, F. Song, A survey of SDN traffic management research. In *2022 11th International Conference on Communications, Circuits and Systems (ICCCAS)* (IEEE, 2022), pp. 231–236
29. G. Wassie, J. Ding, Y. Wondie, Traffic prediction in SDN for explainable QoS using deep learning approach. Sci. Rep. **13**(1), 20607 (2023)
30. N. Ahuja, G. Singal, D. Mukhopadhyay, DDOS attack SDN dataset. Mendeley Data (2020). https://data.mendeley.com/datasets/jxpfjc64kr/1
31. I. Sutskever, J. Martens, G.E. Hinton, Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (2011), pp. 1017–1024
32. I. Lohrasbinasab, A. Shahraki, A. Taherkordi, A.D. Jurcut, From statistical-to machine learning-based network traffic prediction. Trans. Emerg. Telecommun. Technol. **33**(4), e4394 (2022)
33. F. Hu, S. Zhang, X. Lin, Wu. Liu, N. Liao, Y. Song, Network traffic classification model based on attention mechanism and spatiotemporal features. EURASIP J. Inf. Secur. **2023**(1), 6 (2023)
34. S.A. Patil, L. Arun Raj, B.K. Singh, Prediction of IoT traffic using the gated recurrent unit neural network-(GRU-NN-) based predictive model. Secur. Commun. Netw. **2021**, 1–7 (2021)
35. A. Albakri, B. Alabdullah, F. Alhayan, Blockchain-assisted machine learning with hybrid metaheuristics-empowered cyber attack detection and classification model. Sustainability **15**(18), 13887 (2023)
36. R. Thenmozhi, P. Sakthivel, K. Kulothungan, Hybrid multi-objective-optimization algorithm for energy efficient priority-based QoS routing in IoT networks. Wirel. Netw. (2022). https://doi.org/10.1007/s11276-021-02848-z

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.