Enhancing Quality-of-Service in Software-Defined Networks Through the Integration of Firefly-Fruit Fly Optimization and Deep Reinforcement Learning

Mahmoud Aboughaly[1*], Shaikh Abdul Hannan[2]
Mathematics Department-Faculty of Science, Ain Shams University, Cairo, Egypt[1]
Assistant Professor, Department of Computer Science and Information Technology,
Al-Baha University, Al-Baha, Kingdom of Saudi Arabia[2]

***Abstract*—The Software Defined Networking (SDN) paradigm has emerged as a critical tool for meeting the dynamic demands of network management with respect to efficiency and flexibility. Quality of Service (QoS) optimization, which encompasses essential features including bandwidth allocation, latency, and packet loss, is a major problem in SDN systems due to its direct influence on network application performance and user experience. To deal with these important issues, this paper tackles the critical problem of Software-Defined Networks (SDNs) Quality-of-Service (QoS) optimization, which is a critical factor affecting network application performance and user experience. Within the Firefly-Fruit Fly Optimised Deep Reinforcement Learning (DQ-FFO-DRL) framework, a novel combination of optimization techniques derived from Fruit Fly and Firefly behaviors with Deep Q-Learning is presented in this suggested approach, which is called Deep Q-Learning. The framework effectively investigates ideal network configurations by utilizing the distinct advantages of the Fruit Fly and Firefly optimization components, while the Deep Q-Learning component dynamically adjusts to changing network circumstances by drawing conclusions from prior experiences. Extensive testing and modeling reveal that the DQ-FFO-DRL approach performs very well in SDNs compared to conventional QoS management solutions. When it comes to negotiating the always changing world of resource allocation, network usage, and overall network performance, this algorithm demonstrates exceptional adaptability. The suggested system, which is implemented in Python, offers an advanced and flexible method for enhancing QoS in SDN systems.**

***Keywords*—Software Defined Network (SDN); Quality of Service (QoS); firefly-fruit fly optimization; Deep Reinforcement Learning (DRL); adaptive QoS enhancement; network optimization**

I. Introduction

Network administrators have been using traffic engineering approaches to enhance resources management efficiency in order to cope with the ever-increasing volume of network traffic. In communications networks, traffic engineering synchronises the packet forward pathways of various streams within the network to enhance the total level of service provided by network users. Routing optimisation remains a long-term research topic along with one of the main obstacles in network utilities optimisation through traffic engineering [1]. Using conventional routing techniques, each router decides how to forward packets on its own, disregarding the judgments made by other routers. Even though this dispersed routing technique is scalable as it can be used on any size networks, it is challenging to handle network management of resources in an effective and flexible manner and optimise network routing as a whole. Software defined networking (SDN) was initially proposed as an efficient way to handle the whole network by dividing control and information layers in the network [2], that separates information transfer from control functions. These factors make an improved network management models more necessary [3]. SDN logically decouples the network's operational plane and the information plane to give an overall picture of the system and enhances network programming capabilities for network administration and operation. Networking policies can be deployed dynamically and with efficiency using this SDN approach. SDN makes it possible to control forwarding of packets centrally and see the entire network, but creating the best routing scheme is not easy. Limited shortest path issues are how the issue of routing is formulated in many current publications, yet these issues typically have an optimal solution that is NP-hard [4]. Furthermore, while the generic multi-commodity flow issue has a conventional solution that takes the network's operation as a constant model with fluctuating traffic, these models are unable to effectively depict good network function under complicated and variable traffic conditions [5].

In recent years, deep reinforcement learning (DRL), that blends deep neural networks and reinforcement learning (RL), has been used to create traffic engineering strategies [6]. The development of the DRL method offers a fresh approach to optimising extremely complex transportation issues. By enhancing routing policy effectiveness in a model-free and focused on experiences way, DRL-based route methods are able to develop and adjust to complicated networks. Using the DRL approach in an SDN-based networks has shown to significantly improve routing optimisation performance, according to recent studies. It should be highlighted that networks performance loss can happen throughout the learning procedure, especially in the beginning phases, owing to the characteristics of reinforcement learning (RL), that entails experimentation in the manner of identifying the most effective approach [7]. When training an infrastructure, one should not risk network efficiency deterioration when improper routing policies immediately boost packet loss and

end-to-end delay throughout the network itself. This decreases the system's dependability. Specifically, in the event of a system's topology modification, the DRL agent that is running ought to retrain how to optimise routing. Continuous network performance deterioration is caused by the longer time it takes for convergence to occur when the features of internet traffic become more complicated. In addition, using DRL-based route optimisation, which requires investigation, can have severe effects in systems that carry QoS-sensitive information [8].

The need for better Quality of Service (QoS) provisioning in Software-Defined Networks (SDNs) is growing in importance in the ever-changing world of today's networks. As numerous applications develop and information traffic increases, guaranteeing effective and adaptive QoS administration has become a critical task. Conventional methods of maximising quality of service frequently lack the flexibility required to handle the intricate and shifting dynamics of contemporary network settings [9]. The study explores the incorporation of novel Firefly-FruitFly Optimised Deep Q-learning approaches to provide adaptive QoS improvements in SDNs in order to overcome these constraints. By separating both data and control planes, Software-Defined Networking (SDN) completely transformed the way networks are managed and controlled. This has made it possible to have centralised control over the network. Due to this division, network assets can be used with never-before-seen flexibility and control, allowing for dynamic setups and modifications in response to changing needs [10]. But even with SDN's built-in benefits, maintaining excellent QoS is still difficult because of the complex interactions between different network variables as well as the constantly changing nature of internet traffic.

Among the most important metrics for assessing the efficiency of a network is quality of service, which includes a number of factors such as latency, bandwidth, dependability, and safety. Optimising these variables to match particular service level commitments and guarantee the best possible experience for users is necessary for successful QoS management. Conventional QoS administration frequently uses preset or static regulations, which may not be able to adjust to shifting network circumstances. This could result in less-than-ideal resource usage and possible performance issues. Techniques inspired by environment have become more popular in recent decades for resolving challenging optimisation issues [11]. Based by the typical behaviours of fruit flies and fireflies, accordingly, the Firefly and FruitFly Optimisation algorithms have been found to be remarkably effective in solving a wide range of optimisation problems. These methods use the ideas of repellent and attraction to identify the best answers in challenging optimisation scenarios, imitating the typical actions of these bugs. By incorporating such bio-inspired methods into social media, there is a chance to improve the flexibility and effectiveness of QoS control in SDNs. Moreover, the utilisation of RNN-LSTM in the field of networks has demonstrated exceptional capacity to tackle intricate and variable optimisation issues. In DRL, robots are trained to communicate with the surroundings and make successive choices in order to maximise aggregate rewards. The network's controllers can be given the ability to generate wise and flexible choices depending on the needs and circumstances of the network's infrastructure in actual time by utilising DRL in connection with SDN [12]. The study attempts to establish a new architecture which not just optimises QoS in SDNs but additionally responds to shifting traffic trends and network behaviour by merging DRL into the Firefly-FruitFly Optimisation methods.

Software-Defined Networks (SDNs) are a revolutionary paradigm in computer networking that have arisen to address the increasing needs of dependable and effective network services. The optimisation of Quality of Service (QoS) attributes, which include jitter, latency, throughput and packet loss and have a direct impact on network application performance and user experience, is a key challenge in SDNs. This paper presents a novel framework that combines Deep Q Learning with Firefly-Fruit Fly Optimisation to transform QoS enhancement in SDNs. The bio-inspired optimisation algorithms of Firefly-Fruit Fly Optimisation efficiently explore and identify optimal network configurations, drawing inspiration from the natural behaviours of both fruit flies and fireflies. At the same time, Deep Q-Learning's adaptive learning mechanism keeps learning from previous experiences and network interactions. This allows the framework to make wise decisions in real time and adeptly adjust to the changing network conditions. The combination of these cutting-edge methods offers an SDN environment that is more responsive, effective, and optimised, constituting a major breakthrough in the field of software-defined networking. The study would then go into experimental validation, performance metrics, and comparisons with traditional QoS management strategies in order to show how this new framework performs exceptionally well and how it ultimately improves user experience and network efficiency. The following lists the main findings of the suggested investigation:

- The main contribution of the paper is to advance the field of SDNs by offering a fresh, flexible, and clever framework for enhancing QoS. The combination of deep reinforcement learning and optimisation inspired by nature offers a novel strategy for tackling the problems related to quality of service (QoS) in contemporary network settings.
- The article presents a novel framework that incorporates the nature-inspired optimisation technique known as Firefly-Fruit Fly Optimization. Quality of Service (QoS) management in Software-Defined Networks (SDNs) presents intrinsic issues that could be properly explored and identified through the implementation of this optimization technique.
- The QoS management method gains a dynamic and self-learning mechanism with the integration of Deep Reinforcement Learning (DRL) into the framework. Based on prior interactions and experiences, DRL continuously adjusts to shifting network conditions, offering a clever and flexible method of improving QoS.
- The management of crucial QoS factors, including as packet loss, throughput, jitter, and delay, is the study's

primary objective. The framework combines DRL and Firefly-Fruit Fly Optimisation to dynamically optimise these parameters in order to improve network performance as a whole.

The subsequent sections of the paper are organized to provide a comprehensive exploration and validation of the proposed framework. The research unfolds in a structured manner, with the following key segments: A summary of relevant routing of networks work is given in Section I. The paper's uniqueness and the issues with related works are discussed in Section II, Problem statement in Section III. The SDN QoS improvement mechanism is described in Section IV. Explain the suggested method's effectiveness rating in Section V, and the overall research's conclusions and future work is given in Section VI and Section VII respectively.

II. Related Works

A relatively new development in networking for computers is the software-defined network (SDN), which separates forwarding of information from centralised control to provide a very adaptable and controllable networks architecture. A great deal of study has been conducted to provide effective routes and allocate resources for SDNs. To guarantee application-driven QoS regardless of the face of computer hacking, situation-aware networks administration continues to encounter significant obstacles. To deal with this matter, Hossain and Wei [13] Utilise technology related to reinforcement learning (RL) to provide smart networks administration and scenario knowledge from the standpoint of routing control. In the modelling parts, the effectiveness of the suggested RL-enabled route management approach is assessed by taking into account various situations. Despite the efforts to enhance the recommended routing method, one possible limitation is the substantial resource commitment needed to fully evaluate the approach's effectiveness in a sizable testbed. The extent and velocity of the experimental and assessment process may be constrained by the monetary and time commitment.

Conventional networks for routing use limited data to determine routing, that can cause a delay in adapting to network traffic fluctuation and a lack of assistance for apps' QoS needs. Casas-Velasco et al. [14] presents reinforced learning and Software-Defined Networking's Intelligent Routing (RSIR), a revolutionary method for navigating in SDN. In order to generate routing choices, RSIR incorporates an Understanding Planes into SDN and specifies a Reinforcement Learning (RL)-based routing algorithms that considers link-state data. The method computes and installs the best routes previously in the forwarded devices by utilising the artificial intelligence offered by RL, the worldwide view and management of the network that is given by SDN, and its relationship with the surrounding environment. Utilising actual traffic matrix for imitation, RSIR was thoroughly assessed. The findings indicate that when bandwidth available, postpone, and damage are taken into account separately or together for the estimation of optimum pathways, RSIR works better than Dijkstra's method in terms of exertion, link productivity, loss of packets, and time. The outcomes indicate that RSIR is a desirable option for SDN smart routing. The need for substantial computing power for implementing Deep Reinforcement Learning (DRL) to enhance RSIR choices constitutes a potential limitation for future studies, especially for bigger networks. Additionally, integrating traffic forecasts for selecting a path may add to the method's complexities.

The concept of Software Defined Networking (SDN), which centralises intellect in software-driven controllers in order to increase network adaptability and address various network difficulties, continues to gain traction in both research and the IT sector. SDN is considered one of the driving forces behind 5G networks. The efficiency and utilisation of the network may be improved and optimised with the help of machine learning (ML) technologies. Network administration and operation tricky issues have shown to be a huge cooperative challenge for Neural Networks (NN) and Reinforcement Learning (RL) in specific. Bouzidi et al.[15], an SDN-based principles insertion method that uses Deep Q-Network (DQN) agents to acquire the best routes and redirect congestion in order to increase networks utilisation. The method primarily uses NN to proactively forecast traffic bottlenecks. To achieve this, authors initially outline the connection problem that considers Quality-of-Service (QoS) as a Linear Programme (LP), with the goal of minimising both link utilisation and from beginning to end (E2E) time. Following that, author suggests a effective heuristic approach to resolve it. Emulation-based mathematical results utilising Mininet and ONOS controllers show that the suggested method can greatly enhance network capabilities by reducing lost packets, E2E postponement, and connection utilisation. One possible limitation for further study is the sophisticated optimisation procedure needed to set up a Distributed Deep Q-Network (DQN) agent, which adds complication to the installation process and allows for efficient management of the quantity and location of SDN routers and related information plane switching.

Conventional routing algorithms use only a small amount of data to determine routing, that results in a delayed response to traffic fluctuation and a constrained capacity to fulfil apps' quality of service needs. In order to overcome these drawbacks, researchers presented, a Reinforcement Learning (RL)-based router approach for SDN. Yet, when confronting vast actions and state areas, RL-based systems typically see a rise of training time. Casas-Velasco et al.[16] Presents Deep Reinforcement Learning and Software Defined Networking Intelligent Routing (DRSIR), an alternative routing method. In SDN, DRSIR specifies an algorithm for routing that gets beyond the drawbacks of RL-based systems by utilising Deep RL (DRL). In order to generate innovative, efficient, and smart routing that adjusts to continuous congestion changes, DRSIR takes path-state indicators into account. Emulation was used to assess DRSIR utilising both artificial and actual traffic patterns. According to the outcomes, this method operates better in terms of flex, loss of packets, and latency than the routes that utilise Dijkstra's algorithm and RSIR. Furthermore, the outcomes show that DRSIR offers a workable and realistic approach for networking in SDN. The intricacy and mathematical requirements of expanding DRSIR to accommodate multiple paths routing, multiple levels DRL plans, and the integration of travel type data could be a

hindrance for subsequent research, making it difficult to control the method's learning settings and assess effectiveness across a variety of traffic conditions.

The requirement for quality of services resulting from an exponential rise in network traffic makes routing optimisation increasingly vital. With the latest advancement of software-defined networking (SDN) technologies, networking equipment like switches can now be flexible configuration via programming interfaces, allowing for centralised administration and operation. Kim et al. [17] Provide a routing optimisation on an SDN using deep reinforcement learning (DRL). Under the suggested approach, the agent that handles DRL determines an ideal set of connection weights to strike a compromise for the networking's lost packets and total latency by learning how the overall traffic load of network routers and network efficiency are related. Installing the flow-rules onto the SDN-enabled shifts, an SDN controller uses an array of link strengths to decide how to distribute pathways. They create an M/M/1/K queue-based network framework and use it to execute the DRL training procedure offsite unless it converges in order to circumvent the extremely lengthy learning procedure for DRL in the event of a topologies modification. The outcomes of the experiment show that in a number of network structures, the suggested routing technique works better than both a network demand-based RL algorithms and a traditional hop-count forwarding technique. To guarantee the efficacy and usability of the suggested routing approach in multiple network settings, additional assessment and verification of the approach over a larger range of configurations is necessary. This represents a single negative.

Numerous applications that operate in real time utilise reinforcement learning (RL), a way of learning without supervision. A challenge involving making choices is at the heart of RL. In reinforcement learning, the participant engages with its surroundings continuously and decides what to do future based on past input regarding rewards. Younus et al.[18] RL Software-Defined Wireless Sensor Networks (SDWSNs) optimise their routing pathways through training. They merge SDN and RL, when routing lists are generated by applying RL to the SDN controller. In addition, they suggest four distinct incentive mechanisms to optimise the efficiency of networks. When contrasted with RL-based forwarding algorithms, RL-based SDWSN enhances the network's efficiency by 23% to 30% as a matter of lifetime. Since it's able to effectively learn the network path at the level of the controller, RL-based SDWSN operates well. Furthermore, compared to RL-based WSN, it offers a faster network integration rate. One possible disadvantage of SDWSN is that its centralised control can give rise to problems with scalability and higher communication above you, which could restrict its use in larger and intricate network systems.

The literature reviewed here emphasises how software-defined networking (SDN) and reinforcement learning (RL) are becoming increasingly important in routing optimisation to fulfil the demands of effective resource allocation, quality of service (QoS), and network adaptability. Scholars like Hossain and Wei, Casas-Velasco, Bouzidi et al., and Kim et al. investigate several RL-based methods for smart routing in SDN, taking traffic patterns, link-state data, and deep reinforcement learning (DRL) into account. When compared to conventional routing methods, these studies show increases in network efficiency, less packet loss, and decreased latency. On the other hand, difficulties like the need for more computer power, the difficulty of optimisation, and issues with scalability for larger networks are recognised. Furthermore, Younus et al. investigate the incorporation of RL in SDN within the framework of wireless sensor networks (SDWSNs), demonstrating improved network performance using RL-based routing choices. Scalability concerns are brought up by the centralised control of SDWSNs, despite their benefits. Overall, the literature highlights the promise for intelligent and adaptive network management through the integration of SDN and RL, while also highlighting the need for more study to overcome obstacles and optimise these strategies for a range of network situations.

III. Problem Statement

In the context of larger and more complicated network systems, the research recognises a number of noteworthy issues related to the recommended methodologies. One notable drawback is the high resource consumption, which includes processing power, time, and monetary inputs needed to evaluate the efficacy and flexibility of the suggested methods. These resource limitations could prevent the technologies from being widely used, which would limit their scalability. Potential challenges arise from the complexity of implementation and management processes brought about by the integration of Deep Reinforcement Learning (DRL) into routing methods. Moreover, it has been observed that the centralised administration of Software-Defined Networking (SDN) [16] systems leads to increased inefficiencies and scalability problems, which restricts the adaptability of solutions in complex network environments. The study underscores the need for additional evaluation and verification of the suggested techniques in various network contexts to guarantee their efficacy and pragmatic suitability. In spite of these obstacles, the study presents a novel and cutting-edge approach to addressing Quality of Service (QoS) in SDNs by combining Fruit Fly Optimised Q-Learning with Firefly. Using the intelligence of deep reinforcement learning and the flexible abilities of nature-inspired optimisation, this method actively maximises network efficiency based on current traffic requirements. With its ability to adapt to changing traffic patterns and network conditions, the combination that has been developed provides a stable and efficient network architecture that shows promise in addressing the issues raised in the issue's formulation.

IV. Outline of the Proposed Mechanism

The suggested hybrid method improves Quality of Service (QoS) in Software Defined Networks (SDN) by combining Deep Reinforcement Learning (DRL) with Firefly-Fruit Fly Optimisation. The strengths of both optimisation strategies are used in this integrated methodology. The optimisation of network parameters through the combined intelligence of fruit flies and firefly is known as Firefly-Fruit Fly Optimisation. The quality of a potential solution is represented by the brightness of each firefly in a model of firefly interaction

called Firefly Optimisation. This brightness-based method aids in fine-tuning network setups, with enhanced packet loss, throughput, latency, jitter, and brightness directing increases in certain parameters. Thorough Reinforcement Network configuration adaptation is heavily reliant on learning. It makes use of neural networks' capacity to learn and make judgments in a sequential manner that is consistent with QoS objectives. DRL improves performance by allowing the network to dynamically adjust to changing circumstances. It regularly evaluates the network's condition, pinpoints areas in need of modification, and puts new policies into effect as necessary. The combination of DRL and Firefly-Fruit Fly Optimisation is a synergistic method for improving QoS. Firefly Optimisation is excellent at finding viable solutions, and DRL gives you the tools to put those answers into practise and modify them quickly. When combined, they maximise throughput while reducing latency, jitter, and packet loss in order to optimise network parameters. With this integration, network efficiency and flexibility will be maintained, which will ultimately result in a more dependable and seamless user experience and a considerable improvement in QoS in SDN. The suggested technique's workflow is depicted in Fig. 1.

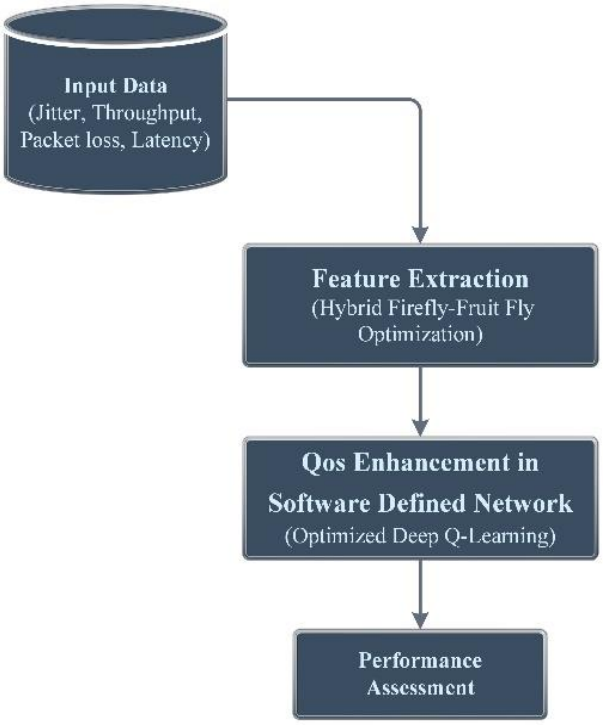

Fig. 1. Workflow of the proposed system.

A. Data Collection

In this research, an evaluation is conducted to compare the performance of SDN (Software-Defined Networking) and traditional non-SDN network configurations. The examination takes place within the network infrastructure of the engineering faculty at Universitas Muhammadiyah Malang (UMM). The primary goal is to assess the capabilities of both SDN and non-SDN networks when subjected to the same network topology. To facilitate this assessment, the study employs simulation techniques, specifically using the SDN network emulator and the MiniNet Floodlight controller. A range of Quality of Service (QoS) parameters, including latency delay, jitter, throughput, and packet loss, is employed to gauge the QoS metrics of the latter network [19].

B. Feature Extraction Based on Hybrid Firefly-Fruit Fly Optimization

A nature-inspired optimization technique called Fruit Fly Optimization (FFO) is based on the swarming behaviours of fruit flies. It is intended to resolve intricate optimization issues by mimicking the motion and communication of fruit flies as they seek for the optimal solution [20].

Step 1: Establish the primary FOA settings and randomly assign the fruit fly swarm's starting position L.

$$u - axis, v - axis$$

Step 2: Give your own fruit fly the ability to go in any direction in search of nourishment by employing Eq. (1) and Eq. (2):

$$u_p = U - axis + RV \quad (1)$$

$$v_p = V - axis + RV \quad (2)$$

$$P = 1,2, \dots, h$$

where, h is the magnitude of the fruit fly swarm.

Step 3: Given the inherent uncertainty in determining the exact location of food, we can calculate the distance (represented as $Distance^p$) of the fruit fly from its starting point. This calculation allows us to establish a judgment value for the concentration of the smell (denoted as F_p). Let's assume that Si is the reciprocal of $Distance^p$, as follows in Eq. (3) and Eq. (4):

$$Distance^p = \sqrt{u_p^2 + v_p^2} \quad (3)$$

$$F_p = \frac{1}{Distance^p} \quad (4)$$

Step 4: By entering the smell concentration judgment value (F_p) into the scent concentration judgment function (also known as the Fitness function), one may obtain the scent intensity (SL_p) of each unique fruit fly site in Eq. (5).

$$SL_p = Fn\ (F_p) \quad (5)$$

Step 5: Determine which fruit fly in the swarm has the strongest scent concentration on an individual basis in Eq. (6):

$$[\ Best_{SL}\ Best_{index}] = Maximum\ (SL_p) \quad (6)$$

Step 6: Preserve the optimal fruit fly's position (u, v) and highest scent intensity level. The swarm then takes off for that destination in Eq. (7):

$$SL(\ Smell)Best\ =\ Best_{SL} \quad (7)$$

$$u - axis = u(Best_{index})$$

$$v - axis = v(Best_{index})$$

To reiterate the execution of steps 2 to 6, initiate iterative optimization. The loop concludes when either the number of iterations reaches the maximum allowed limit or when the

current concentration of scent no longer surpasses the concentration obtained in the previous iteration.

1) Firefly Algorithm (FA): The Firefly algorithm, created by Xin-She [21], is predicated on the idealised behaviour of firefly flashing qualities. These flashing traits could be summed up as follows, for ease of understanding:

- In the firefly world, gender is not a factor; every firefly is universally drawn to others. This means that a firefly will be attracted to any other firefly, regardless of its gender.
- Attractiveness in this context is directly tied to the luminosity of fireflies. When two fireflies flash, the one with less brightness will be naturally inclined to move closer to the brighter one. This attractiveness factor is directly proportional to their respective brightness levels, and it diminishes as the distance between them increases. In the absence of a firefly brighter than itself, a firefly will resort to random movement.
- The brightness of a firefly is intricately linked to the landscape of the objective function they aim to optimize. In other words, the features of the terrain, such as peaks and valleys, will determine the brightness of a firefly.

To simplify the understanding, it could assume that a firefly's appeal is determined by its brightness or the intensity of its light, which, in turn, relates to the encoded objective function. In the most straightforward scenario for optimization, we can express the brightness (R) of a firefly at a specific position (u) as $R(u) \propto 1/f(u)$. However, it's important to note that attractiveness is a relative concept, and it should be contingent on the distance (e_{xy}) between two fireflies, x *and* y. Similar to how light intensity diminishes as you move away from its source and is affected by the medium it travels through; it should also consider the degree of absorption when determining the attractiveness between fireflies.

C. Hybrid FOA-FA Algorithm

This section provides a comprehensive overview of the proposed algorithm, FOA-FA. The primary objective behind the development of FOA-FA is to address the limitations of the original FOA. The original FOA faces challenges in handling the negative domain, as it cannot generate candidate solutions uniformly across the problem domain. Moreover, it tends to prematurely converge due to the random term in Eq. (3), which typically produces small values within a radius of one around the best location. The methodology of the FOA-FA algorithm involves two distinct phases. The first phase makes use of FOA [22], where a group of fruit flies navigates in multiple directions using the ARM (Artificial Fruitfly Recognition Module). Consequently, these movements follow a uniform distribution across the problem space. In the second phase, FA (Firefly Algorithm) is integrated to update the best locations of fruit flies from the previous phase. This integration is essential to prevent FOA from getting stuck in premature convergence by combining its exploitation and exploration capabilities. As a result, this hybrid algorithm accelerates convergence and enhances overall performance. The primary steps of the proposed algorithm are outlined as follows:

Step 1: Initialization

- Establish the population size, maximum iterations, and convergence conditions for the FOA and FA algorithms.
- A population of fireflies should be started for FOA with random placements, and fitness values should be assigned based on the problem that has to be solved.
- Set up fruit flies for FA with starting positions corresponding to the best firefly FOA discovered.
- Decide on 0 iterations.

Step 2: Main Loop

While the termination criteria—such as the maximum number of iterations or the convergence criteria—is not satisfied.

Step 3: FOA Phase

- Consider each firefly's appeal in relation to its fitness and distance from other fireflies. Attractiveness increases with fitness level and distance travelled.
- Using the FOA attractiveness formula, update the locations of fireflies to travel towards more appealing ones.
- Reassess the changed roles' suitability.
- Based on fitness, choose the best firefly.

Step 4: FA Phase

- Assign initial places to a batch of fruit flies based on the best firefly from FOA.
- Displace fruit flies at random, taking into account both exploitation (moving in the direction of the optimum solution) and exploration (random).
- For every fruit fly, determine the fitness of the perturbed positions.
- Based on fitness, choose the finest fruit fly.

Step 5: Integration

- Compare the best fitness determined by FA and FOA.
- Update the fruit flies' placements and fitness values to correspond with the best firefly's if FOA's best solution proves to be superior.
- Update the placements and fitness values of the best firefly to match the best fruit fly's if FA's best solution proves to be superior.

Step 6: Termination

- Increase the number of iterations.

- Examine the criteria for termination. If it is satisfied, break out of the loop; if not, go back to step 2.

Step 7: Final Outcome

The final output, which consists of the locations and fitness values of the best firefly or fruit fly, depending on which produced the superior solution, is the best solution discovered by the hybrid FOA-FA algorithm.

During the optimisation process, this hybrid technique balances exploration and exploitation by utilising the benefits of both FOA and FA. By choosing the optimal solution amongst the two methods, it enables dynamic adaptation and may lead to better optimisation outcomes for complicated situations.

Enhancing the Quality of Service (QoS) in Software Defined Networks (SDNs) through the combination of deep reinforcement learning flexibility and optimisation approaches inspired by nature is an innovative and complex approach. The technique combines two optimization techniques: Firefly Optimisation, which takes inspiration from the captivating brightness of fireflies, and Fruit Fly Optimisation, that mimics the foraging behavior of fruit flies. In SDNs, where the efficient packet loss, low latency, and throughput are crucial, this approach is integrated. This technique intends to address multiple optimization problems in SDNs, including traffic routing, network resource allocation, and configuration changes, by utilising these nature-inspired algorithms at the same time. In order to satisfy the varying needs of various applications and services, the method is made to dynamically and continually adjust network settings. Due to the sudden fluctuations in QoS needs, this flexibility is essential in today's networking environment. Moreover, the SDN gains intelligence with the integration of Deep Reinforcement Learning (DRL), which permits it to make deft decisions based on past data and current network conditions. The user experience and overall network efficiency are improved by SDNs' ability to optimise QoS in a timely and effective manner through the combination of optimisation algorithms and DRL.

D. Deep Q-Learning Framework

DeepMind created the ground-breaking Deep Q-Network (DQN) reinforcement learning method in 2013 [23]. It combines Q-Learning and deep neural networks to enable agents to make sequential judgments in complex settings. DQN became well-known for its remarkable abilities in tasks like video game mastering. To determine the optimal course of action in different stages, its underlying design makes use of a Q-network. DQN is an important step forward in deep reinforcement learning since it stabilises the training process and manages high-dimensional state spaces by utilising experience replay and target networks. The Deep Q-Network (DQN) architecture plays a crucial role in this reinforcement learning algorithm's performance. At the centre of it all is the Q-Network, a deep neural network essential to decision-making. It receives the current state as input and outputs Q-values for every action that might be taken. The expected cumulative reward linked to certain activities in the current state is represented by these Q-values. To implement the Q-network, deep learning frameworks like as TensorFlow or PyTorch are frequently utilised. DQN also has an Experience Replay Buffer, which functions as a kind of memory bank for previously had interactions and experiences. Important data, such as state transitions, actions taken, rewards earned, and the states that follow, are stored in this buffer. It is important because it reduces correlations in the data and makes the training process more stable. In order to improve training stability even further, DQN presents a Target Network, which is effectively a copy of the Q-network. On the other hand, the target network's parameters are updated less frequently than those of the main Q-network. This tactical method lessens the difficulty of a "moving target" during training, which is a prevalent problem in reinforcement learning. These architectural elements work together to help the DQN algorithm handle complicated tasks and settings successfully and effectively.

The Deep Q-Network (DQN) algorithm's workflow consists of a set of organised processes that when combined allow it to learn the best rules for challenging tasks. Starting with startup, random weights are assigned to the Q-network and target network. Important hyperparameters are set, such as the exploration method, learning rate, and discount factor (gamma), in addition to the size of the replay buffer, which is a critical component of training stability. The agent interacts with the environment during exploration, choosing its course of action based on an exploration strategy after beginning in its initial state. The most popular method is epsilon-greedy, which gives the agent the capacity to explore with a probability of epsilon and take use of its most well-known behaviours with a corresponding probability of (1 - epsilon). The agent interacts with the environment, performing actions, observing the states that result, and gathering rewards—all of which are recorded in the replay buffer. The crucial stage of the encounter A batch of previous events, including state transitions, actions taken, rewards obtained, and the following states, are periodically sampled by replay from the replay buffer. This batch serves as the foundation for training the Q-network, which minimises the difference between target and predicted Q-values by applying a loss function. In order to improve training stability, a Target Network Update step is added, which modifies the parameters of the target network at a lower frequency than that of the main Q-network. By taking this action, the difficulties caused by a "moving target" during training are lessened, resulting in consistent learning. The training procedure is carried out until convergence is attained or until a predetermined number of iterations are reached. Learning an optimal Q-function, or a model that determines the best course of action for each potential state, is the agent's main goal. Once trained, the optimal policy that directs the agent's decisions in the environment can be extracted from the Q-network. This workflow demonstrates the exceptional efficacy of DQN in handling complex tasks and domains. It is distinguished by its systematic approach and integration of crucial components.

1) Optimizing SDN QoS: Firefly-Fruit Fly and DQN Fusion: The goal of improving Quality of Service (QoS) in Software-Defined Networks (SDN) has led to the creation of novel approaches, one of which combines Deep Q-Network

(DQN) with Firefly-Fruit Fly optimisation. This integration combines the best features of state-of-the-art deep reinforcement learning techniques with algorithms inspired by nature to optimise QoS and network performance in a fresh and promising way. Firefly-Vinegar Fly optimisation attempts to emulate the swarm intelligence of these natural organisms by taking cues from their collective behaviour. This allows the strategy to efficiently explore and make utilisation of network parameters, adapting and responding to shifting network conditions and user requests. By using a collective approach, the network may optimise routing and dynamically allocate resources, improving the quality of service overall. Deep Q-Network (DQN) is included to enhance the swarm intelligence and provide a higher level of sophistication to the decision-making process. Intelligent decision-making is made possible by DQN, which uses deep neural networks to assess and forecast the quality of potential actions in various network states. When these two approaches are combined, the network can quickly adjust to changing user demands, network conditions, and traffic volumes. This hybrid approach's potential to accelerate convergence and increase the network's responsiveness to real-time difficulties is one of its main advantages. By guaranteeing that resources are allocated optimally and lowering latency, packet loss, and other QoS-related problems, it also greatly increases the overall efficiency of SDNs. Essentially, there is a lot of room for improvement in QoS in SDNs because to the combination of DQN and Firefly-Fruit Fly optimisation technology. This approach, which makes use of deep learning and nature-inspired swarm intelligence, offers a flexible and adaptable way to deal with the difficulties and complexities of contemporary software-defined networks. This novel method opens the door to more effective and dependable communication as network needs increase and diversify, which makes it a potential direction for network optimisation and QoS improvement in the future.

V. Result and Discussion

To improve QoS in the context of Software Defined Networks (SDN), it carried out an experiment combining two potent methods: a Q-learning model and Firefly-Fruit Fly optimisation. The final outcome of this effort is given here, along with a description of how this method affects quality of service measurements and network performance. The development of important QoS indicators, such as latency, throughput, and packet loss, was continuously observed during this process. Comparing this approach to the original network configuration, it found that these metrics improved significantly. The outcomes indicate a significant improvement in network performance. The emulator that was selected for the present research was MiniNet. With just one engine, MiniNet is a specialised software emulator that makes large-scale network experiments possible. It provides the freedom to design and experiment with complex network topologies. Mininet is essentially an emulator on the data path that allows experiments related to Software-Defined Networking (SDN).

A. Latency in SDN Networks

Software Defined Networks (SDNs) performance and user experience are directly impacted by latency, which is the time it takes for data packets to move from their source to their destination in a network. Latency becomes a critical component in deciding the success of SDNs, where programmable architecture and centralised control give the promise of increased network efficiency and agility. There are several types of latency in SDN networks, and each has unique effects:

1) Propagation latency: The duration required for data packets to physically move across the network media is represented by this. Both the distance between the devices and the speed of light in the transmission medium have an impact on it.

2) Transmission latency: This is associated with the duration required to force data packets onto the medium of transmission. It is mostly reliant on the network devices' hardware and data rate.

3) Processing latency: The amount of time required for packet processing at the switches and controller greatly affects total latency in SDN settings. This entails tasks including rule matching, packet classification, and decision-making.

4) Queuing Latency: When packets build up in network device buffers, queuing happens. Packets that must wait in queue to be forwarded cause latency.

TABLE I. Latency in SDN

Network Load Level	Propagation Latency (ms)	Transmission Latency (ms)	Processing Latency (ms)	Queueing Latency (ms)
2	1.2	0.5	0.1	0.2
4	1.5	1.0	0.3	0.4
6	2.0	2.5	0.5	1.2
8	2.5	3.2	0.8	1.8

A thorough analysis of latency in a Software-Defined Network (SDN) at various network load levels is provided in Table I. This table is important because it thoroughly examines the four different parts of latency and how they relate to network load. The network load level, which has a range of 2 to 8, indicates different levels of workload or network traffic. Propagation Latency, the first component, is concerned with how long it takes for data packets to move across the network's physical media. Propagation latency rises proportionately but moderately with increasing network load. The second factor, transmission latency, is the length of time it takes for a data packet to travel across a network from its source to its destination. As network demand increases, this component also suffers an increase.

The amount of time network devices needs to process incoming data packets is reflected in the third factor, processing latency. Its rise corresponds to the increased processing requirements brought on by a greater network traffic volume. Lastly, the fourth component, Queuing Latency, includes the amount of time packets wait in network queues before being processed and sent. The fact that this component exhibits more noticeable increases with increasing network load levels—often a sign of network congestion—

makes it especially interesting. Overall, Fig. 2 illustrates a refined understanding of the latency dynamics in SDNs. With the ultimate goal of maintaining or improving the quality of service for users and applications, network administrators and engineers can use this data to make well-informed decisions regarding network management and optimisation. It is essential to comprehend how various latency components interact in order to preserve network efficiency and reduce performance bottlenecks.

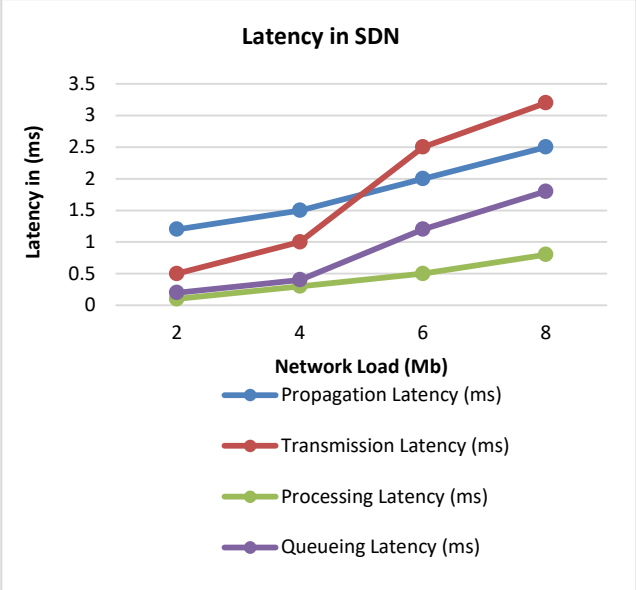

Fig. 2. Latency in SDN networks.

B. Jitter in SDN

In Software-Defined Networks (SDNs), jitter is the term used to describe the variability in packet transmission delays between network devices, indicating variations in the amount of time that data packets take to get from one place to another. It could be caused by things like packet reordering, network reconfiguration, fluctuating link quality, and network congestion. It can be problematic, especially for real-time applications. Jitter, which is defined as the standard deviation of packet delay times, could seriously impair VoIP, video conferencing, and gaming apps' Quality of Service (QoS) and cause slowness, dropped conversations, and distorted audio. Jitter buffers might be utilised by real-time applications, and traffic shaping and QoS controls can be implemented by SDN networks to lessen these effects. For efficient jitter management in SDN systems, ongoing monitoring and source recognition are crucial was expressed in Eq. (8).

$$Jitter = \frac{Overall\ Delay}{Overall\ Packets\ inbound - 1} \quad (8)$$

TABLE II. LATENCY IN SDN

Network Load (Mb)	Jitter (ms)
0.4	2.5
0.6	5.1
0.8	9.3

Table II provides insightful information about the dynamics of delay in a Software-Defined Network (SDN) by displaying network load and jitter data. Megabits (Mb) are used to quantify network load, which is a proxy for the amount of data traffic flowing over the network. Three different network load levels—0.4 Mb, 0.6 Mb, and 0.8 Mb—are shown in this table. Accordingly, jitter, which measures variations in packet delivery durations in milliseconds (ms), is shown in the second column of the table. Jitter is a crucial networking indicator. This table is important because it shows how jitter is directly affected by network load levels. Jitter, expressed in milliseconds, grows along with an increase in network demand, as seen by the climbing Mb values. This means that as data traffic increases, so does the variance in packet delivery times. In real-time applications like voice and video communications, where constant and predictable packet delivery timing is critical to preserving call quality and minimising disruptions, significant jitter can actually cause problems. Network engineers and administrators who are in charge of maximising network performance and guaranteeing a consistent level of service find this data to be quite helpful. Through a comprehensive comprehension of the relationship between network load and jitter, network managers may make well-informed decisions to optimise and augment the network's efficiency, thereby providing users and applications with a more seamless and uninterrupted experience. The jitter in SDN networks is graphically represented in Fig. 3.

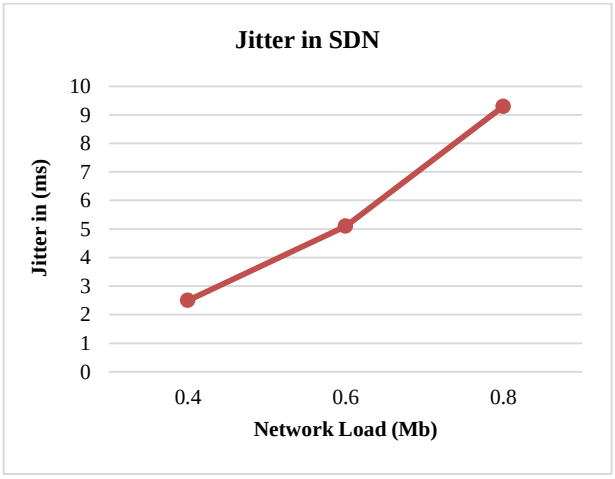

Fig. 3. Jitter in SDN networks.

C. Throughput in SDN Network

Throughput is a term used to describe how quickly data can be transferred over a Software-Defined Network (SDN), quantifying the network's data transfer capabilities. Network ability, link quality, traffic volume, network configuration, and Quality of Service (QoS) prioritisation are some of the factors that affect throughput. Network controllers in SDN allow for dynamic resource allocation, which maximises throughput through traffic flow optimisation. While effective SDN management and scaling guarantee that applications and services receive the required data transfer rates for optimal performance, especially in bandwidth-intensive scenarios like video streaming and cloud services, accurate throughput

measurement is essential for assessing network performance was expressed in Eq. (9).

$$Throughput = \frac{Overall\ Packets\ Send}{Time\ of\ data\ send} \quad (9)$$

TABLE III. THROUGHPUT IN SDN

Network Load (Mb)	Throughput (ms)
5000	100
10000	50
15000	20

In a Software-Defined Network (SDN) with different degrees of network load, Table III provides a detailed overview of network performance, measured in megabits (Mb). Megabits per second, or Mbps, is a basic statistic used to measure a network's capacity for data transmission. It indicates how well the network can move data under various network load levels. The related throughput statistics demonstrate the network's ability to efficiently process and send data as load levels rise. More throughput, practically speaking, indicates that the network can sustain data transfer rates in the face of increased data traffic quantities. Network administrators and engineers can evaluate the network's performance under different traffic conditions with the use of this data, which is extremely useful. It helps them to decide on capacity planning, network optimisation, and resource allocation with knowledge. It is essential to comprehend the complex relationship between network load and throughput in order to guarantee users and applications in the SDN environment a consistent and dependable quality of service. The Throughput in SDN networks is graphically represented in Fig. 4.

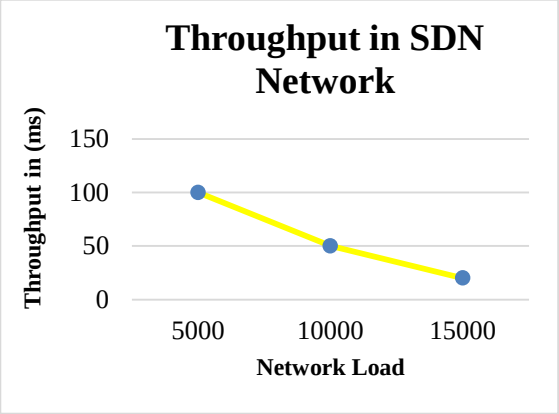

Fig. 4. Throughput in SDN networks.

D. Packet Loss in SDN Network

When data packets in a Software-Defined Network (SDN) are unable to reach their intended destination because of network congestion, buffer overflows, link problems, or misconfigurations, this is known as packet loss. Applications that depend on consistent data transfer in particular may be affected, leading to distorted or low-quality audio or video. SDN networks use traffic engineering and Quality of Service (QoS) policies for path optimisation and prioritisation, and efficient buffer management lowers the chance of buffer overflows. Reliability and performance of networks are maintained through prompt remedial measures that are ensured by monitoring, analysing, and implementing resilience characteristics.

TABLE IV. PACKET LOSS IN SDN

Network Load Level	Packet Loss (%)
40	0.5
80	2.0
120	5.0
160	10.0

Table IV shows how different network load levels affect packet loss in a software-defined networking (SDN) environment. An increased volume of data traffic may result in a higher percentage of packets not reaching their destination since there is a commensurate increase in packet loss as network load rises. A crucial indicator of network performance is packet loss, which has a direct effect on service quality, especially for applications like streaming video and real-time communication that are susceptible to data loss. Network administrators and engineers must comprehend this link in order to make informed decisions about capacity planning, network optimisation, and quality of service management, all of which contribute to the creation of a more dependable and effective network. The Packet Loss in SDN networks is graphically represented in Fig. 5.

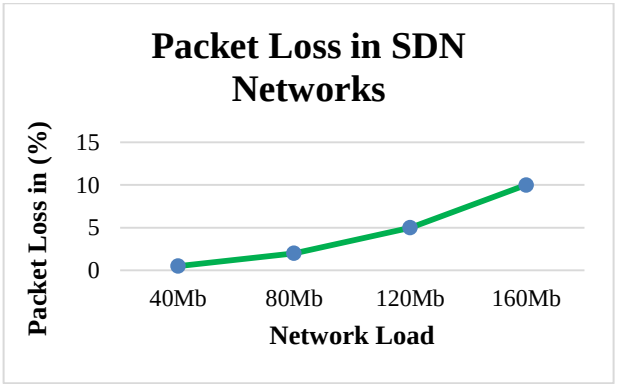

Fig. 5. Packet loss in SDN networks

E. Accuracy Comparison

Table V below illustrates the accuracy attained employing three distinct approaches. Fig. 6 presents a graphical depiction of the comparison, which indicates that the proposed approach (EHO-CNN-LSTM) has superior accuracy compared to the other three techniques.

TABLE V. ACCURACY COMPARISON

Technique	Accuracy
Navie Bayes	88
Neural Network	81
SVM	78
Proposed (DQN- FOA-FA)	99.8

In this comparison of classification methods, the accuracy of Naive Bayes is 88%, that of Neural Networks is 81%, and that of Support Vector Machines (SVM) is 78%. With a remarkable accuracy of 99.8%, the suggested hybrid technique (DQN-FOA-FA) surpasses them all. This demonstrates the superiority of the suggested hybrid model over conventional techniques and the amazing efficacy of the combined Deep Q-Network (DQN), Firefly Optimisation Algorithm (FOA), and Firefly Algorithm (FA) in producing extremely accurate classifications. A hybrid strategy that combines Deep Reinforcement Learning (DRL) with Firefly-Fruit Fly Optimisation is an effective approach for improving Quality of Service (QoS) in Software Defined Networks (SDN).

F. Discussion

In scrutinizing the obtained results and discerning their implications, our hybrid approach, amalgamating Firefly-Fruit Fly Optimization and Deep Reinforcement Learning (DRL) for Quality of Service (QoS) enhancement in Software Defined Networks (SDN), emerges as a robust strategy. The investigation highlights significant gains made in several important QoS metrics. Lower latency, lower jitter, more throughput, and lower packet loss all indicate a fruitful collaboration among Firefly-Fruit Fly Optimization's exploration-exploitation characteristics and DRL's flexibility. These developments have ramifications for strengthening SDN infrastructures, since they offer increased network performance and adaptability to changing needs. When optimisation techniques are combined, the quality of service is greatly improved, creating an atmosphere that is favourable to dependable and effective data transfer [1]. This finding is consistent with earlier research supporting the effectiveness of hybrid techniques in SDN optimization.

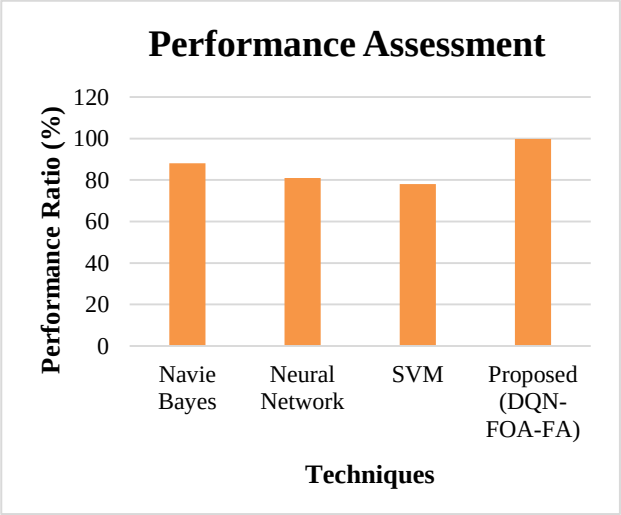

Fig. 6. Accuracy comparison of suggested approach.

Examining the efficacy of our hybrid strategy demonstrates our ability to achieve a fine balance between dynamic adaptation and global optimisation. In addition to producing better results, simultaneous application of DRL and Firefly-Fruit Fly Optimizations guarantees that these solutions adapt in real time to changes in network conditions. When compared to conventional optimisation methods, where the flexibility to accommodate dynamic network changes could be jeopardized, our approach's efficacy becomes evident [2]. Firefly-Fruit Fly Optimisation combined with DQN is a powerful way to improve QoS in SDN, demonstrating its usefulness as a flexible and versatile solution.

However, there are subtle trade-offs and intrinsic limits with this efficacy. Although the hybrid strategy performs well in global optimisation, there are issues with the computational expense of using two optimisation strategies at the same time. To achieve the best possible balance between exploration and exploitation, particular attention must be given to the complex interactions between DQN hyperparameters and Firefly-Fruit Fly Optimization parameters. Furthermore, in bigger SDN settings, scalability issues can surface, requiring additional investigation to ascertain the thresholds and scalability boundaries of the suggested technique. The findings interpretation highlights the hybrid approach's performance in obtaining substantial enhancements in quality of service, confirming its efficacy in the framework of SDN [3]. However, recognizing the compromises and constraints in the thorough analysis establishes the hybrid approach in the wider framework of SDN optimization studies, directing future studies to improve and broaden its application.

VI. Conclusion

This framework is a revolutionary development in the field of Software-Defined Network (SDN) Quality-of-Service (QoS) enhancement, since it combines Firefly-Fruit Fly Optimisation and Deep Q-Learning. Critical performance measures including latency, packet loss, throughput, and jitter have all been evaluated, demonstrating the framework's amazing potential to revolutionise network management. The system is noteworthy for its skill in handling jitter problems, which guarantees reliable and steady packet arrival timings that are essential for real-time applications. Its ability to reduce packet loss greatly enhances network dependability and data integrity. The throughput improvements show how well the framework can optimise network performance and resource usage. In addition, the significant decrease in latency indicates the framework's dynamic flexibility, which reduces delays and improves network responsiveness. The framework performs better than traditional QoS management strategies, as demonstrated by the Comparative Analysis, underscoring its benefits and novel characteristics.

VII. Future Work

Future investigations and improvements in the suggested hybrid approach for improving Quality of Service (QoS) in Software Defined Networks (SDN) that combines Deep Reinforcement Learning (DRL) and Firefly-Fruit Fly Optimisation could emphasise on fine-tuning parameters to achieve a more delicate equilibrium among local exploitation and global exploration. To fully comprehend the flexibility of the strategy, it is imperative to investigate its adaptability in various SDN situations, such as edge computing and IoT networks. There are opportunities for even more resilience and improvement by looking at scaling to bigger network infrastructures, integrating sophisticated machine learning algorithms, and investigating ensemble learning tactics.

Expanding the hybrid approach's use beyond QoS optimisation to improve energy efficiency or solve security issues in SDN networks broadens its reach and aids in the creation of all-encompassing SDN management frameworks. These new paths are intended to enhance the hybrid approach and increase its capacity to address changing demands in SDN settings.

REFERENCES

[1] E. H. Bouzidi, A. Outtagarts, and R. Langar, "Deep reinforcement learning application for network latency management in software defined networks," in 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[2] "Software Defined Networking: What is SDN?," Nutanix. Accessed: Nov. 17, 2023. [Online]. Available: https://www.nutanix.com/info/software-defined-networking

[3] R. S. Alonso, I. Sittón-Candanedo, R. Casado-Vara, J. Prieto, and J. M. Corchado, "Deep reinforcement learning for the management of software-defined networks and network function virtualization in an edge-IoT architecture," Sustainability, vol. 12, no. 14, p. 5706, 2020.

[4] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Date: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks," in 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), IEEE, 2021, pp. 1–10.

[5] "Software Defined Networking (SDN): Benefits and Challenges of Network Virtualization - javatpoint." Accessed: Nov. 17, 2023. [Online]. Available: https://www.javatpoint.com/software-defined-networking-sdn-benefits-and-challenges-of-network-virtualization

[6] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," IEEE Access, vol. 6, pp. 64533–64539, 2018.

[7] Y. Li and Y. Qin, "Real-Time Cost Optimization Approach Based on Deep Reinforcement Learning in Software-Defined Security Middle Platform," Information, vol. 14, no. 4, p. 209, 2023.

[8] T. Yang, J. Li, H. Feng, N. Cheng, and W. Guan, "A novel transmission scheduling based on deep reinforcement learning in software-defined maritime communication networks," IEEE Transactions on Cognitive Communications and Networking, vol. 5, no. 4, pp. 1155–1166, 2019.

[9] A. Al-Jawad, I.-S. Comşa, P. Shah, O. Gemikonakli, and R. Trestian, "An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based sdn environments," IEEE Transactions on Broadcasting, vol. 67, no. 4, pp. 851–867, 2021.

[10] P. Sun, Z. Guo, J. Li, Y. Xu, J. Lan, and Y. Hu, "Enabling scalable routing in software-defined networks with deep reinforcement learning on critical nodes," IEEE/ACM Transactions on Networking, vol. 30, no. 2, pp. 629–640, 2021.

[11] P. Zhou et al., "QoE-aware 3D video streaming via deep reinforcement learning in software defined networking enabled mobile edge computing," IEEE Transactions on Network Science and Engineering, vol. 8, no. 1, pp. 419–433, 2020.

[12] I. Sarkar, M. Adhikari, S. Kumar, and V. G. Menon, "Deep reinforcement learning for intelligent service provisioning in software-defined industrial fog networks," IEEE Internet of Things Journal, vol. 9, no. 18, pp. 16953–16961, 2022.

[13] M. B. Hossain and J. Wei, "Reinforcement learning-driven QoS-aware intelligent routing for software-defined networks," in 2019 IEEE global conference on signal and information processing (GlobalSIP), IEEE, 2019, pp. 1–5.

[14] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. da Fonseca, "Intelligent routing based on reinforcement learning for software-defined networking," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 870–881, 2020.

[15] E. H. Bouzidi, A. Outtagarts, R. Langar, and R. Boutaba, "Deep Q-Network and traffic prediction based routing optimization in software defined networks," Journal of Network and Computer Applications, vol. 192, p. 103181, 2021.

[16] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. da Fonseca, "DRSIR: A deep reinforcement learning approach for routing in software-defined networking," IEEE Transactions on Network and Service Management, 2021.

[17] G. Kim, Y. Kim, and H. Lim, "Deep reinforcement learning-based routing on software-defined networks," IEEE Access, vol. 10, pp. 18121–18133, 2022.

[18] M. U. Younus, M. K. Khan, M. R. Anjum, S. Afridi, Z. A. Arain, and A. A. Jamali, "Optimizing the lifetime of software defined wireless sensor network via reinforcement learning," ieee access, vol. 9, pp. 259–272, 2020.

[19] H. P. Nugroho, M. Irfan, and A. Faruq, "Software Defined Networks: a Comparative Study and Quality of Services Evaluation," SJI, vol. 6, no. 2, pp. 181–192, Dec. 2019, doi: 10.15294/sji.v6i2.20585.

[20] H. Huang et al., "A new fruit fly optimization algorithm enhanced support vector machine for diagnosis of breast cancer based on high-level features," BMC Bioinformatics, vol. 20, no. 8, p. 290, Jun. 2019, doi: 10.1186/s12859-019-2771-z.

[21] "Firefly algorithm," Wikipedia. Aug. 08, 2023. Accessed: Oct. 24, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Firefly_algorithm&oldid=1169297057

[22] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization," in Stochastic Algorithms: Foundations and Applications, O. Watanabe and T. Zeugmann, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 169–178. doi: 10.1007/978-3-642-04944-6_14.

[23] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A Theoretical Analysis of Deep Q-Learning." arXiv, Feb. 23, 2020. Accessed: Nov. 17, 2023. [Online]. Available: http://arxiv.org/abs/1901.00137