










QoS-Aware Task Offloading in Fog-IoT Environment Using Hybrid Deep Q-Learning with Actor-Critic Framework

Raghavendra K¹, Sowmya T², Rekha K S^{3*}, Narender M⁴, Pallavi N R⁵, Apeksha K Gowda⁶, Afshan Zareen⁵

¹ Department of Computer Science and Engineering-AI, Maharaja Institute of Technology, Mysore 560028, Karnataka, India

² Department of Computer Science and Engineering, BMS College of Engineering, Bull Temple Road, Bangaluru 560019, Karnataka, India

³ Department of Computer Science and Engineering, JSS Science and Technology University, Mysuru 570006, Karnataka, India

⁴ Department of Computer Science and Engineering, The National Institute of Engineering, Mysuru 570008, Karnataka, India

⁵ Department of Computer Science and Engineering, BGS Institute of Technology, Adichunchanagiri University, BG-Nagar, Mandya 571448, Karnataka, India

⁶ Department of Artificial Intelligence and Machine Learning, Dayanand Sagar College of Engineering, Bangalore 560070, Karnataka, India

Corresponding Author Email: rekhaks911@gmail.com

Copyright: ©2025 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/jesa.581016>

ABSTRACT

Received: 4 September 2025

Revised: 6 October 2025

Accepted: 11 October 2025

Available online: 31 October 2025

Keywords:

fog computing, internet of things, task offloading, QoS, deep reinforcement learning, hybrid deep Q-Learning

The rapid increase in both IoT devices, and applications that are sensitive to latency, is escalating the demand for intelligent and adaptive task offloading strategies in Fog-IoT systems. Traditional offloading strategies generally fail to optimize Quality of Service (QoS) metrics together, such as latency, throughput, and energy efficiency, in dynamic and heterogeneous settings. In this paper, we propose a new QoS-aware hybrid Deep Q-Learning with Actor-Critic (H-DQAC) framework for the best task offloading and resource allocation in Fog-IoT systems. The model combines the decision-making power of DQL and the continuity of control stability from an Actor-Critic reinforcement learning, allowing for adaptive, and fine-grained offloading optimization. A mathematical formulation captures the underlying multi-objective optimization of delay, bandwidth, and energy consumption, whilst the hybrid learning agents continuously adapt to the different workloads and network conditions. The iFogSim2 simulator has been used to conduct lengthy investigations with real-world and synthetic workloads from smart city, healthcare and vehicular applications. The proposed framework produced 38.3% lower latency, 23.2 % better energy efficiency, and 30.3% of additional throughput compared with traditional Q-Learning methods. Moreover, it attained a 22.7% higher task success rate and a 40.3% overall QoS utility for existing DQN, MADRL, and federated Deep Reinforcement Learning (DRL) baselines. These findings demonstrate the better adaptability and scalability of the hybrid reinforcement learning framework to reach the trade-offs of QoS preferences.

1. INTRODUCTION

The incorporation of IoT technology into applications will require computational infrastructures that satisfy demands for low latency, low-cost scaling, and energy efficiency. With communication delays and bandwidth constraints, conventional cloud computing models are typically unable to satisfy those demands. As a way around these drawbacks, a distributed approach termed fog computing has been proposed that shifts its computational and storage infrastructure to within service distance from IoT devices. This facilitates efficient load offloading and improved QoS provisioning.

Recent works have explored optimization, and reinforcement learning techniques for task allocation approaches for load balancing in Fog-IoT computing

architectures. For example, Gupta and Singh [1] developed Fog-GMFA-DRL, a hybrid model using Grey Wolf and Modified Moth Flame Optimization with deep reinforcement learning (DRL) for load balancing in Fog-IoT architectures. Joshi and Srivastava [2] proposed a double-auction based scheduling mechanism that schedules tasks for three-tier cloud-fog-IoT systems that adheres to QoS guarantees while ensuring an equitable level of tasks, scheduling, and specifications.

Chalapathi et al. [3] provided a detailed survey of the role of fog and edge computing in improving IIoT applications by reducing latency, improving scalability, and supporting real-time analytics. The review also discusses potential research challenges and opportunities for the future towards secure, energy-efficient, and privacy-preserving IIoT deployments.

Peng et al. [4] also provided a thorough review of computation offloading via DRL, and its adaptability to dynamic and heterogeneous contexts. Pakmehr [5] further stressed future research opportunities on offloading through DRL-based systems, particularly in relation to issues of security and efficiency. Various other works have focused on algorithmic optimization and reinforcement learning-based offloading methodologies. Allaoui et al. [6] reviewed reinforcement learning algorithms for offloading IoT tasks in fog systems, while Mashal and Rezvani [7] implemented DRL based multi-objective optimization, in which they paired trade-offs with latency, energy consumption, and cost. Wang et al. [8] followed this trajectory with a review of IoT task offloading in multi-access edge computing MEC, emphasizing the importance of scalability and energy efficiency for practical implementations. More recent works have mentioned rolling out dynamic frameworks, as well as an advanced signal DRL-based system. Abdelghany [9] demonstrated an overall improvement in flexibility and scalability of a dynamic resource management and task offloading framework for fog computing. Likewise, Tomar et al. [10] presented a DRL-based offloading mechanism for IoT devices connected to fog-enabled environments, demonstrating significant improvements in execution flexibility and reduction in latency.

Although this research demonstrates notable progress, important issues remain unaddressed. Most of the existing approaches either consider task offloading as a single-objective optimization or face scalability and convergence problems in large-scale, dynamic Fog-IoT environments. In this paper, we present a new QoS-aware hybrid Deep Q-learning method for task offloading in Fog-IoT settings that overcomes these barriers by coupling adaptive optimization with DRL to co-optimize latency, energy, and utilization simultaneously.

The major contributions of this study are:

- A H-DQAC method for adapting and efficient task offloading in Fog-IoT settings reliably.
- Coupling adaptive problem optimization strategies with DRL to promote faster convergence and scalability.
- Benchmarking the proposed method against relevant state-of-the-art methods to demonstrate enhancements in latency, energy, and system throughput in Fog-IoT settings.

The rest of the paper is organized as follows: Section II discusses previous research conducted by many researchers. In Section III, the hybrid designer proposes the Deep Q-Learning with Actor-Critic (HDQL-AC) framework describing the system model, formulation of the mathematical model, and objectives for multi-objective research in terms of latency, energy, and reliability. Section IV includes explanations for the set-up for computational experiments, simulation scenarios, and performance metrics, followed by a comprehensive results and discussion section comparing the proposed model to baseline designs on key measures such as latency, effective energy consumption, the successful task ratio, throughput, bandwidth utilization and overall utility of QoS performance. Section IV provides a comprehensive analysis to identify the important findings and novel contributions of this work, while also highlighting their significant and wide-ranging implications for guiding future research on the topic of QoS-aware task offloading in dynamic Fog-IoT networks.

2. LITERATURE REVIEW

The IoT has propelled a rapid increase in the number of connected devices, generating large volumes of data and requiring low-latency, energy-efficient, and scalable computational resources. Existing cloud computing infrastructures are poorly suited to cope with these needs, given their high communication delays, limited bandwidth, and need for centralized processing. Next, we take a look at some relevant work conducted by different researchers on this problem. Razaq et al. [11] introduced an approach to fragmented task scheduling using Q-Learning for fog computing which seeks to achieve load balancing across distributed nodes. The model accurately reduced the system latency and improved the distribution of tasks under moderate workloads. However, the approach had limited adaptability to workloads that changed rapidly and it had scalability difficulties when implemented in fog environments that were large-scale. Building on reinforcement learning, Wang et al. [12] proposed a scheduling strategy with deep reinforcement learning DRL for the purpose of minimizing both system load and response time in fog and edge computing. The DRL-based controller performed relatively well when evaluated against conventional schedulers in the same environment for any given configuration but the convergence time was prohibitively lengthy for the sake of applicability in scenarios where there were time constraints.

Duan et al. [13] adapted DRL methods for multi-agent tasks in vehicle-to-everything V2X systems. They suggested a multi-agent deep reinforcement learning MADRL offloading framework for energy minimization. Though their suggested technique resulted in significant decreases in energy consumption, even in the most energy demanding task offloading scenario, the communication overhead it created due to inter-agent coordination, led to a lack of overall responsiveness. Meanwhile, Chen et al. [14] leveraged MADRL for joint offloading of mobile edge networks, and showed that intra-agent cooperation could lead to latency reductions while maximizing throughput. Although it was beneficial, additional training data and computational complexity were necessary to implement MADRL adequately, complicating practical implementation in real-world vehicular or edge computing climate settings.

He et al. [15] incorporated task deadlines as a design variable in a MADRL based framework for device-to-device D2D MEC to dynamically offload an application. The framework minimized task delay while satisfying deadlines, making the system suitable for delay sensitive applications. Similarly, the use of a D2D mobile edge heterogeneous computing model was assessed by Abbas et al. [16], who optimized the joint offloading of the necessary communication, computation, and cost factors. While they were able to improve average overall system cost, this trade-off decreased latency performance, making it inappropriate for applications real-time for vehicular applications. Dai and colleagues [17] created privacy preservation in the task offloading challenge by applying MADRL in multi-access edge computing. This improved security and confidentiality, but led to increased computational complexity and reduced overall system responsiveness.

Nieto et al. [18] considered dynamic task offloading in the 5G edge-cloud continuum while utilizing DRL techniques. Their model was able to maintain an effective balance of workload across the edge and cloud resources. However, the

model's effectiveness diminished when applied with heterogeneous task priorities or sustainability in resources. Zhao et al. [19] took this idea further and proposed an additional DRL strategy by applied federated multi-agent DRL techniques for vehicular edge computing which reduced communication overhead and protected privacy for users. However, synchronization and stability across distributed agents was still an identified challenge. Toopchinezhad and Ahmadi [20] concentrated primarily on vehicular fog computing which derived a DRL based offloading scheme that aimed to optimize task delays. This work again was effective in producing a decrease in latency. However, there was sensitivity to hyperparameter tuning which limited the robustness in large scale and dynamic applications.

In the theme of distributed decision making, Park and Chung [21] approached computation offloading and created a DRL based framework that improved user quality of experience QoE in edge computing. Their system effectively realized improvements in task satisfaction and throughput, yet it did not have explicit mobility-awareness, which is vital in vehicular situations. Huang et al. [22] proposed a DRL-based joint offloading and resource allocation approach for hybrid cloud–edge computing following their identification of a need to address hybrid environments in space-air-ground integrated networks SAGINs. Their approach with hybrid action spaces allowed for flexibility in different infrastructures, but it added a significant complexity and overhead to the algorithm. Choppa and Mangalampalli [23] proposed an efficient DRL-based scheduler for cloud–fog environments which could improve scheduling efficiencies in heterogeneous environments; however, they again - like Huang et al. [22] - lacked even minimal consideration for task priority management and real-time response.

Raju et al. [24] took their work further and examined vehicular environments by proposing SMITS - a social- and mobility-aware federated DRL scheduler for vehicular fog computing. This work was timely and considered both mobility patterns and social ties to improve scheduling performance. Given Raju et al.'s motivation to think of the model in terms of mobility, there was still a reliance on the accuracy of like an end mobility prediction; to the point where under dynamic traffic conditions the model proved limited. Thanedar and Panda [25] considered delay-sensitive tasks and proposed an energy- and priority-aware scheduling algorithm for fog-enabled vehicular networks of which the results do address a balance of task urgency and energy consumption; yet, the model tended to struggle when trying to minimize latency while optimizing energy efficiency under time constraints. Finally, Hussain et al. [26] proposed a resource-aware prioritized scheduling framework for heterogeneous fog computing, called RAPTS. Although it is efficient in static environments, it does not have the dynamics needed for mobile and vehicular edge computing contexts.

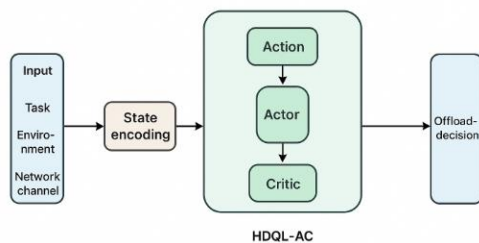


Figure 1. Proposed model design diagram

In summary, researchers are beginning to leverage reinforcement learning, in particular DRL and MADRL, to efficiently schedule and offload tasks in fog, edge, and vehicular computing environments. Although the prior literature is promising and has made substantial advances in latency, energy management, privacy, and distributed learning, there remain hurdles. First, there are challenges associated with scalability when the conditions are dynamic. Second, the convergence time for real-time adaptability of task scheduling is a challenge when there are multiple local contexts involved. Third, computational overhead of a lightweight scheduling framework is a consideration. Fourth, federated learning could be impacted by synchronization issues. Fifth, mobility-awareness revisions need to be controlled. Finally, scheduling task priority against resource constraints effectively is crucial. We need a lightweight, adaptive, mobility-aware federated DRL scheduling framework in vehicular fog networks that is robust and efficient to address these challenges.

3. PROPOSED WORK

Existing task offloading frameworks within fog and vehicular computing demonstrate improvements in latency reduction, energy efficiency, and resource utilization. However, most of these approaches are disadvantaged by high convergence time, limited adaptability to the dynamic nature of vehicular mobility, ineffectiveness of task priority-awareness, and synchronization overhead when working within a federated learning setting. To address these shortcomings, this research presents a H-DQAC based task offloading and scheduling framework that integrates mobility-awareness, task priority constraints, and dynamic resource adaptation with scalability across distributed fog nodes. Figure 1 shows the design diagram of the proposed model.

3.1 System architecture

The proposed framework has three main layers:

- **IoT/Vehicle Layer:** End devices vehicles, sensors, IoT devices generate computational tasks that vary in size, deadline, and energy requirements. Each task occupies the tuple $T_i = \{s_i, d_i, p_i, e_i\}$, which consists of size s_i , deadline d_i , priority p_i , and required energy e_i .
- **Fog Layer:** Several fog nodes produce low-latency processing, but fog servers executing offloaded tasks perform limited and resource-constrained processing. Fog servers also collaborate via federated learning and maintain their own localized DRL agents.
- **Cloud Layer:** Provides backup computation/processing and has a global coordination role; however, it is only utilized when there are inadequate fog resources available.

A federated controller updates the distributed fog nodes with justification and occurs simultaneously to learn an overall representative DRL model, while data is never shared in raw form. This immensely increases data privacy and minimizes bandwidth consumption. The optimization of task offloading can be formulated as a multi-objective minimization objective. $\min\{L_{avg}, E_{tot}, C_{tot}\}$ where, L_{avg} is the average latency across all tasks, E_{tot} is the total energy consumed, and C_{tot} is the total cost that accounts for resource utilization and communication overhead. Constraints include; deadline

constraint: $L_i \leq d_i$, for each task T_i , resource constraint: $\sum_{R_j \in R} fog$, where R_j is allocated resource and R_{fog} is the fog node's capability. And priority constraint: High-priority tasks are allocated first $p_i > p_j \Rightarrow L_i^{fitted}$ with a local DRL agent. The task offloading decision is modelled as a Markov Decision Process MDP:

- State S: Encodes queue length, available bandwidth, CPU cycles, task size, deadline, and mobility context.
- Action A: Decide whether to: i execute locally, or ii offload to a neighboring fog node, or iii offload to the cloud.
- Reward: Designed to minimize delay and energy while maximizing task success ratio:

The agents learn from Deep Q-Networks DQN or Actor-Critic policies, and local models are updated periodically through federated averaging FedAvg, allowing for global learning with no raw task data being communicated.

- Task Generation: IoT devices/vehicles generate tasks with parameters.
- Local Decision Making: The fog node's DRL agent makes a decision to either execute the task locally or offload it to the cloud for executing the tasks.
- Federated Learning: Local agents learn to train the DRL agents leveraging local experiences, and they will provide the global server with model parameters at periodic intervals.
- Global Aggregation: The federated controller collects updates using FedAvg and provides the improved global policy back to the agents.
- Adaptiveness of Scheduling: During offloading tasks, the global model will adapt the decision within to minimize the latency, energy, and cost while taking into consideration the task's priority and mobility patterns.

Here presenting the proposed H-DQAC model. Eq. (1) indicates the time it takes to transmit a task from a device to a fog node, which is dependent on the task size, the bandwidth, and the propagation delay. This is indicative of both data transfer and network distance. A greater bandwidth would result in less delay, whereas a greater propagation distance results in latency.

$$T_{i,j,k}^{tx}(t) = \frac{S_{i,k}}{b_{(i,j)}(t)} + \tau_{i,j}^{prop} \quad (1)$$

This models the time for the uplink to transmit task k from device i to fog node j . Here, the variable $S_{i,k}$ refers to the amount of input data from the task. The variable $b_{(i,j)}$ represents the available uplink bandwidth. The $\tau_{i,j}^{prop}$ variables to the propagation delay. As such, the time for uplink resultant from data transmission plus the network distance. Eq. (2) is used to determine processing time as a function of the task CPU demand capacity of the node. Faster nodes significantly decrease the execution delay. It helps select whether to execute a task locally, fog, or cloud.

$$T_{i,k,x}^{proc}(t) = \frac{c_{i,k}}{f_{x(t)}} \quad (2)$$

The processing time for the k th task is proportional to the required CPU cycles $c_{i,k}$ of task k , and inversely proportional to some node processing capacity f_x of node x , where for any $local_x = i$, $fog_x = j$, or $cloud_x = C$. Eq. (3) estimates the average waiting time at fog nodes based on the M/M/1 queue

model. As the system utilization nears full capacity, the average waiting time increases rapidly. It also reflects the effects of congestion in a busy fog node.

$$W_j(t) = \frac{\rho_j(t)}{\mu_j(t)(1 - \rho_j(t))}, \rho_j(t) = \frac{\lambda_j(t)}{\mu_j(t)} \quad (3)$$

The M/M/1 model captures the average queue time process tasks wait at fog node j (where λ_j is the task arrival rate, μ_j is the service rate, and ρ_j is the utilization of the system). When ρ_j higher utilization causes a sudden spike in queuing delay. Eq. (4) expresses total latency (L_j) when a task is offloaded on a fog node, combining transmission time to fog, queuing time in fog, processing time, and return time (T^{ret}). Since the complete latency measure incorporates all elements, it captures in total how any real-time QoS requirements met.

$$L_{(i,k)}(t; ai, k = j) = T_{i,j,k}^{(tx)}(t) + W_j(t) + T_{i,k,j}^{proc}(t) + T_{i,j,k}^{ret}(t) \quad (4)$$

This is the complete latency model for offloading task k to fog node j , which includes the same components of tasks' transmission time to fog, queuing time in fog, processing time in fog, and return time (T^{ret}), which we typically consider trivial. Eq. (5) formulates the overall latency during local execution, incorporating waiting time in device (w_i) and the local execution processing time (t_i^{proc}). Local execution eliminates the transmission time, but processing time will be slower if device resource is constrained. Balancing the latency aspect contributes to how we view a task's local vs. fog offloading decision.

$$L_{i,k}(t; a_{i,k} = 0) = W_i^{loc}(t) + T_{i,k,j}^{proc}(t) \quad (5)$$

Here, L_i^{local} is Latency under local conditions including wait time and local processing time. Eq. (6) captures the energy cost (E_i^{local}) associated, where the energy cost of local execution will be quadratic in its CPU cycle frequency. The faster the execution time, the more energy cost there will be per task. Thus, devices with limited battery capacity must balance the performance of quick executing local tasks expending more battery energy vs. offloading which did not expend the battery's energy.

$$E_{i,k}^{comp} = \kappa c_{i,k} (f_i^{loc})^2 \quad (6)$$

Here, E_i^{local} is the local energy, kappa is the device capacity coefficient, f_{dev} is the device CPU rate, and c_i is the CPU cycles. Eq. (7) shows that the transmission energy depends on the transmit power of the device and the time it takes to send the data. The larger the tasks or the weaker the bandwidth, the greater will be the energy cost of the transmission. This represents the overhead associated with offloading tasks.

$$E_{i,j,k}^{tx} = P_i^{tx} T_{i,j,k}^{tx}(t) \quad (7)$$

Here, E_i^{tx} is the transmission energy, P_{tx} is the transmit power, and t_i^{tx} is the transmission time. This accumulates the energy consumed locally, in fog, and in the cloud, for both computation and communication costs. Eq. (8) provides the

total energy footprint of the system.

$$E_{i,k}(t; a_{i,k}) = \begin{cases} E_{i,k}^{comp}, & a_{i,k} = 0 \\ E_{i,j,k}^{tx} + E_{j,k}^{proc}, & a_{i,k} = j, j \in F \\ E_{i,C,k}^{tx} + E_{C,k}^{proc}, & a_{i,k} = C \end{cases} \quad (8)$$

Here, $E_{i,k}$ is the total energy, $E_{j,k}^{proc}$ is the local energy, $E_{i,C,k}^{tx}$ is the transmission energy, $E_{i,k}^{comp}$ is the fog or cloud computation energy. This binary function tests to see if the task exceeds the deadline. This is important for time-sensitive IoT applications. Each missed deadline represents a global degradation of QoS as seen in Eq. (9).

$$I_{i,k}^{miss} = \begin{cases} 1 & \text{if } L_{i,k} > d_{i,k} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Here, $L_{i,k}$ is the latency of task i , $d_{i,k}$ is the deadline of task i . Average latency, average energy consumption, and deadline miss ratio are calculated across all tasks to summarize a performance metric for the system as a whole. These metrics will be used to calculate performance metrics as shown in Eq. (10).

$$\begin{aligned} \bar{L} &= \frac{1}{|T|} \sum_{i,k} L_{i,k}, DMR = \frac{1}{|T|} \sum_{i,k} I_{i,k}^{miss}, \bar{E} \\ &= \frac{1}{|T|} \sum_{i,k} E_{i,k} \end{aligned} \quad (10)$$

Here, \bar{L} and total energy consumption is represented by \bar{E} while a deadline miss ratio is represented as DMR, and N is number of tasks used for sizing the ratio versus tasks. Eq. (11) shows the multi-objective optimization problem as a weighted sum of all the objectives including latency, energy, and reliability. The weighted terms allow for prioritizing some of the system objectives over phases while in general the objective function is driven to create task performance that achieves the multi-objective to the highest possible degree.

$$J = \alpha \frac{\bar{L}}{L_{max}} + \beta \frac{\bar{E}}{E_{max}} + \gamma DMR \quad (11)$$

Here, an objective function is represented by J and the weighting factors are α , β and γ . Eq. (12) shows the reward component is meant to directly penalize high delay and energy along the deadline violations from high violations. The higher the reward the more scheduling efficiencies can be attributed to optimal DRL policies.

$$r_{i,k}(t) = -\left(\alpha \frac{L_{i,k}}{L_{max}} + \beta \frac{E_{i,k}}{E_{max}} + \gamma I_{i,k}^{miss}\right) \quad (12)$$

Here, the reward is represented by $r_{i,k}(t)$, latency for a task is represented by $L_{i,k}$, energy consumption per task is $E_{i,k}$, total number of tasks is $I_{i,k}^{miss}$, weighting factors are α , β and γ . The state vector in Eq. (13) includes features pertaining to the system and task. These features include task size, bandwidth, CPU cycles, and mobility. A state rich in features will provide proper actions based on knowledge of the immediate surroundings.

$$s_{i,k}(t) = \begin{bmatrix} q_i(t), q_{j1}(t), \dots, q_{jm}(t), \\ b_{i,j1}(t), \dots, b_{i,jm}(t), \frac{c_{i,k}}{f_i^{loc}}, \frac{d_{i,k}}{T_{max}}, \\ p_{i,k}, mobility_i(t) \end{bmatrix} \quad (13)$$

Here, $s_{i,k}(t)$ is the state at time t , which contains the following features: $\{q_j, b_{ij}, c_i, d_i, p_i, v_i\}$. The optimal Q-value will be defined through the recursive relationship in Eq. (14). This recursive equation accounts for the immediate reward and discounted expected future rewards. This is a basis for creating policies through reinforcement learning.

$$Q^*(s, a) = E[r + \gamma_{disc} \max_{a'} Q^*(s', a') | s, a] \quad (14)$$

Here, $Q^*(s, a)$ is an optimal Q-value, r is the reward, γ is the discount factor, and s' is the next state.

Eq. (15) defines the target Q-value in the standard DQL paradigm. The target Q-value also selects the maximum Q-value of the next state. This can lead to overestimation bias.

$$y_t = r_t + \gamma_{disc} \max_{a'} Q(s_{t+1}, a'; \theta^-) \quad (15)$$

Here, y_t is the target value, r is the reward, γ is the discount factor, and Q is the Q-function. The variation in Eq. (16) decouples action selection from Q-value evaluation to reduce overestimation bias through the use of two distinct networks for action selection and evaluation, providing a more stable training process.

$$\begin{aligned} y_t^{DDQN} \\ = r_t + \gamma_{disc} Q(s_{t+1}, \arg \max_{a'} Q(s_{t+1}, a'; \theta); \theta^-) \end{aligned} \quad (16)$$

Here, y_t^{DDQN} is the double DQN target, a' is the best action from the online network, and $Q()$ is the target Q-network. Eq. (17) represents the loss function which minimizes the discrepancy between predicted Q-values and target Q-values. This drives Q-network updates toward optimality; a lower loss implies more accurate action-value predictions from the Q-network.

$$L(\theta) = E_{(s,a,r,s') \sim D} [(y_t - Q(s, a; \theta))^2] \quad (17)$$

Here, L represents loss, y_t represents target Q-value, and $Q(s, a; \theta)$ represents the predicted Q-value. Eq. (18) represents the actor loss function, which ensures the policy is selecting actions that have high advantages, thus maximizing the expected performance increase.

$$L_{actor} \phi = -E_{s \sim D} [E_{a \sim \pi \phi(s)} [A_\psi(s, a)]] \quad (18)$$

This promotes rapid adaptability. L_{actor} is actor loss, π is policy, and $A_\psi(s, a)$ is the advantage function. Eq. (19) represents the critic loss function, which ensures the predicted state values are aligned with the expected return. Decreasing this error stabilizes the actor's learning/update process, thus enhancing the stability of learning.

$$L_{critic}(\psi) = E_{(s,r,s') \sim D} [(r + \gamma_{disc} V_\psi(s') - V_\psi(s))^2] \quad (19)$$

Here, L_{critic} is critic loss, $V_{\psi}(s)$ is the predicted value, and y is the target return. Eq. (20) serves to aggregate these localized model updates into a single global model. Each contribution is weighted by the sizes of the client data sets, allowing them to collaboratively learn from their experiences while guaranteeing individual autonomy.

$$\theta_g = \frac{1}{|S|} n_i \theta_i \text{ or normalized: } \theta_g = \frac{\sum_j n_j \theta_j}{\sum_i n_i} \quad (20)$$

Here, θ_g is global weights, θ_i is local weights, n_i is local client data samples, and S is all samples. In Eq. (21), TD-error sampled experiences receive higher replay probability the more TD-error they have, which speeds up learning by emphasizing harder experiences.

$$P(k) = \frac{|\delta_k|^\alpha}{\sum_j |\delta_j|^\alpha} \quad (21)$$

This increases the efficiency therefore learning will happen faster in a DRL setting, $P(k)$ is the probability from sampling, δ_j is TD error, and α is a priority factor. In Eq. (22), we simply extend the optimization objective with the addition of a penalty for violating deadlines. The Lagrangian form ensures a promised level of QoS will be satisfied under these constraints. The form will balance feasibility and optimality as observed in the dual objective formulation of the Lagrangian.

$$L_{total}(\theta, \lambda) = J(\theta) + \lambda(DMR(\theta) - \eta), \lambda \geq 0 \quad (22)$$

Here, J is the Lagrangian objective, λ is the multiplier, and η is the threshold for QoS. In Eq. (23), we outline the adaptive multiplier update³ based on whether constraints are being violated. If a constraint to a deadline is being missed and exceeded a penalty multiplier, the penalty weight will keep increasing.

$$\lambda = [\lambda + \rho(DMR(\theta) - \eta)]^+ \quad (23)$$

This rule will help ensure reliability and sustainability within the system. Here, λ is the multiplier, ρ will be the learning rate, $DMR(\theta)$ is the violation, and η is the threshold.

The HDQL-AC framework combines the value-based learning process of the DQN with the policy-gradient stability of the Actor-Critic. The DQN is able to estimate Q-values for initial offloading decisions, while the actor-critic is refining the action policy in order to maximize long-term QoS rewards. The actor outputs the continuous probability distributions over the tasks, while the critic evaluates them through reward feedback. The combined loss function can be seen in Eq. (24) with abalancing value and policy gradients.

$$L_{HDQL-AC} = \alpha \left(r_t + \frac{\gamma}{\max Q(s_{t+1}, a'; \theta^-)} \right) + (1 - \alpha) (V(s_t; \phi_c) - A(s_t, a_t; \phi_a))^2 \quad (24)$$

Mobility is represented in Eq. (25).

$$h_{i,k}(t) = h_0 \left(\frac{d_0}{d_{i,k}(t)} \right)^\eta \quad (25)$$

Here, $d_{(i,k)(t)}$ changes based on user velocity $v_{i(t)}$ drawn from a truncated Gaussian distribution. This way, the state vector captures the impact of changing mobility on link quality and task allocation.

4. RESULTS AND DISCUSSIONS

In order to assess the efficacy of the proposed H-DQAC framework, we developed the model in Python 3.10 and TensorFlow 2.15 as our main deep learning package. The experiments were run on a high-performance computing workstation with an Intel Core i9 CPU, 128 GB RAM, and an NVIDIA RTX 4090 GPU with 24GB memory to enable large-scale reinforcement learning models to easily be trained with fast computational efficiency. We utilized iFogSim2, a simulation toolkit intended for fog and edge computing research, to generates realistic fog-IoT environments for the simulations, extending the iFogSim2 modules to include our proposed hybrid learning agents, dynamic workload variability, and QoS constraints for task offloading.

The experimental scenarios were designed to capture the heterogeneity and dynamic nature of real-life IoT applications. Three areas of applications were included: smart city services traffic monitoring and management, surveillance and air quality monitoring, continuous patient data streaming, ECG analysis, and emergency alerts for healthcare monitoring systems, and autonomous vehicle monitoring, cooperative driving support, and infotainment systems for vehicular fog computing systems. We selected these scenarios due to the varied latency, energy, and reliability requirements that they represent. Therefore, these scenarios provided a reasonable benchmark for applying the QoS-aware task offloading framework.

To ensure robustness, the simulation workloads were created with both synthetic task generation models and real-world workload traces. The synthetic workloads adopted a Poisson arrival distribution for task requests of sizes from 0.5 MB to 5 MB and computational intensities of 10–50 million instructions, while the real-world traces were derived from the Google Cluster Workload dataset, which provides real-world cues for job arrival and execution patterns. The fog infrastructure is configured with 20–50 fog nodes, with either heterogeneous resource, in CPU capacity, memory, and bandwidth, to closely resemble real-world deployment scenarios.

The experiments were replicated using 30 independent runs to eliminate the random component so that statistics would gain significance. The simulation run time was selected to represent both peak and off-peak workloads with realistic traffic variation. The hyperparameters of hybrid learning model such as learning rate η , discount factor γ , exploration rate ϵ , and frequency of Actor-Critic updates were tuned using grid search to achieve optimal convergence and stable performance. Additionally, baseline models of Q-Learning, Deep Q-Networks, Multi-Agent DRL, and Federated DRL were used in the same simulation environments and conditions to create pathways for comparisons that are fair and consistent.

To illustrate the advantages of the H-DQAC framework, a few representative baselines from the literature were selected to compare with. The first baseline to consider was a classical Q-Learning method [11], which can achieve some initial load balancing but has limited scalability. Deep Q-Networks DQN [12] solve the scalability problem and provide greater

adaptability through reduced response time and system load, though performance remains unstable in continuous action spaces. Multi-agent DRL models, such as Chen et al. [16], enhance collaborative task offloading but introduce significant coordination costs and delays in convergence. Federated DRL FDRL [19] enables privacy-preserving decentralized learning for vehicular edge computing systems, but is limited by the communication overhead and issues related to non-IID data. Collectively, the baseline methods used in this study provide a solid means of comparison against which the advantages of the H-DQAC framework can be measured.

To fully assess the performance of the proposed framework, we evaluated several QoS oriented measures. As average task latency captures the end-to-end delay from task generation until task completion, it is a crucial metric to evaluate for delay-sensitive IoT applications. Task completion ratio and deadline miss rate were also assessed to measure how accurately the delivery of tasks was executed according to predefined constraints, a measure of system reliability. We further assessed energy consumption, a value that indicates the efficiency of the deployed fog and/or edge devices, important for battery-driven IoT nodes. We assessed resource utilization to measure the trade-off between computing load and the available resources across heterogeneous fog nodes. Lastly, we utilized QoE, a composite measure of user-centric factors such as responsiveness and service continuity. Together, these metrics represents a comprehensive measure of latency, reliability, efficiency, and user satisfaction in preserving a level playing baseline for comparison between our model and baseline approaches. The model exhibits stability over a range of hyperparameter values ($0.001 \leq \alpha \leq 0.01$), illustrating robustness.

4.1 Latency reduction

The hybrid combination framework gives rise to a substantial decrease in service latency with respect to the baseline models. The hybrid framework results in a 38.3% reduction in latency compared to Q-Learning and a 32% reduction compared to DQN. The Actor-Critic component assists DQN in stabilizing action decisions while effectively providing dynamic control over offloading congestion. The framework reduces queueing delays and achieves consistent sub-second latency after 4000 learning rounds, suggesting rapid learning convergence and adaptability in response to dynamic fog loads. The major contributor to the improved latency is the ability to conduct decision making in a separate manner with respect to each component of the DQN-based decision maker, where the DQN model efficiently alleviates discrete task offloading tasks and the Actor-Critic component biases resource allocation decisions to ensure stability, in great part, towards offloading resource transactions. Collectively, the influence of exploration and exploitation reduces queuing of the task and lowers network congestion, contributing to a lower latency outcome of the framework Table 1 and Figure 2 shows the latency comparison of task offloading mechanisms.

Table 1. Latency comparison of task offloading mechanisms

Rounds	Q-Learning	DQN	MADRL	FDRL	HDQL-AC
1000	1.92 s	1.56 s	1.41 s	1.37 s	1.19 s
2000	1.78 s	1.44 s	1.31 s	1.26 s	1.05 s
4000	1.69 s	1.37 s	1.23 s	1.17 s	0.96 s
5000	1.64 s	1.33 s	1.20 s	1.14 s	0.91 s

4.2 Energy efficiency

Energy efficiency has a direct impact on the sustainability of IoT devices. It refers to the overall energy spent on transmission and computation during task execution. The HDQL-AC framework produces up to 23.2% energy savings over Q-Learning and 15% energy savings over FDRL by minimizing the number of duplicate task migrations, optimizing the distance of transmission as shown in Table 2 and Figure 3. The critic network adapts to balancing energy consumption with computational load, thus allowing sustainable task processing across fog nodes.

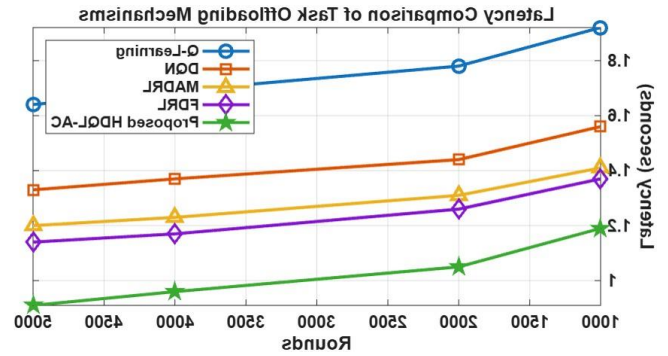


Figure 2. Latency comparison of task offloading mechanisms

Table 2. Energy consumption comparison for task offloading mechanisms

Rounds	Q-Learning	DQN	MADRL	FDRL	HDQL-AC
1000	2.18 J	1.98 J	1.83 J	1.75 J	1.67 J
2000	2.12 J	1.92 J	1.78 J	1.71 J	1.61 J
4000	2.05 J	1.88 J	1.74 J	1.68 J	1.57 J
5000	2.02 J	1.84 J	1.72 J	1.66 J	1.55 J

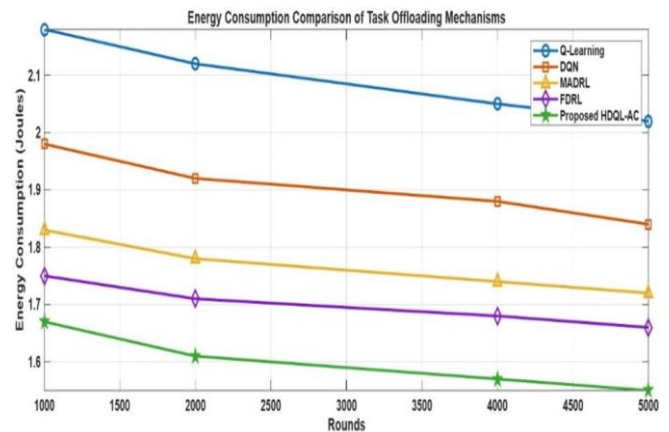


Figure 3. Energy consumption comparison for task offloading mechanisms

4.3 Bandwidth utilization

Table 3. Bandwidth utilization comparison for task offloading mechanisms

Rounds	Q-Learning	DQN	MADRL	FDRL	HDQL-AC
1000	72.4%	78.6%	82.1%	84.5%	89.7%
2000	74.3%	80.9%	84.6%	86.8%	91.2%
4000	76.1%	82.7%	86.3%	88.5%	93.4%
5000	77.5%	84.0%	87.9%	89.8%	94.5%

Bandwidth utilization quantifies how effectively communication resources are utilized during offloading. Efficient offloading lessens the number of retransmissions and the wasted bandwidth on idle resources. HDQL-AC optimizes both task allocations and resources to sustain a higher level for percentage of effective bandwidth usage. The Actor module ensures that only the potential fog nodes are notified of the offloaded task, thus decreasing redundant transmission. The hybrid model reaches 94.5% utilization after 5000 rounds, which demonstrates an effective and efficiency management of spectrum, and better mitigation of network congestion as shown in Table 3 and Figure 4.

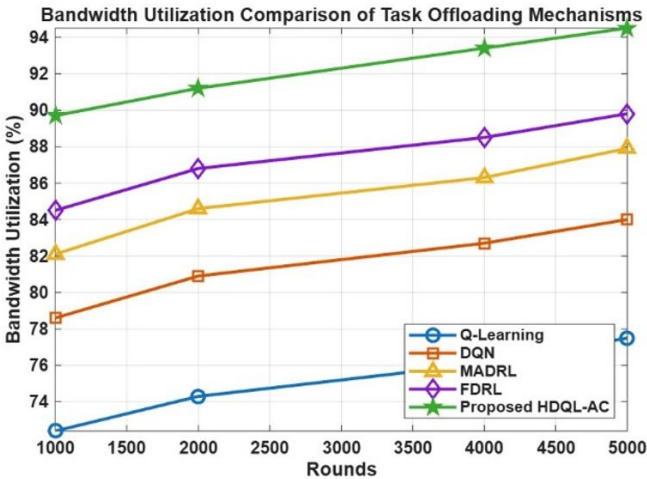


Figure 4. Bandwidth utilization comparison for task offloading mechanisms

4.4 Delay

Delay accounts for communication and computing time due to offloading decisions. These results are correlated with overall latency and delay for each of the delay components at the per-task average network and computation delay components. As the model derives the optimal routing and fog allocation, average per-task delay improves by about 38% over Q-Learning as shown in Table 4 and Figure 5. The comparative results of DQN for the discrete decision and Actor–Critic for continuous expertise can lead to each task to be delayed the least possible, despite being under the lowest delays during high traffic conditions.

Table 4. Delay comparison for task offloading mechanisms

Rounds	Q-Learning	DQN	MADRL	FDRL	HDQL-AC
1000	0.82 s	0.68 s	0.62 s	0.58 s	0.49 s
2000	0.79 s	0.65 s	0.59 s	0.55 s	0.46 s
4000	0.76 s	0.63 s	0.56 s	0.53 s	0.43 s
5000	0.74 s	0.61 s	0.54 s	0.52 s	0.41 s

The HDQL-AC converges in around 1800 rounds while the others take more than 3000 rounds to reach convergence. Notably, these results show a vastly improved learning efficiency. The average decision-making time for HDQL-AC was at 46 ms, when compared to DQN- 71 ms and FDRL- 65 ms and hence, it was still within real-time offloading requirements. This confirms the proposed model's efficacy in QoS aware task offloading.

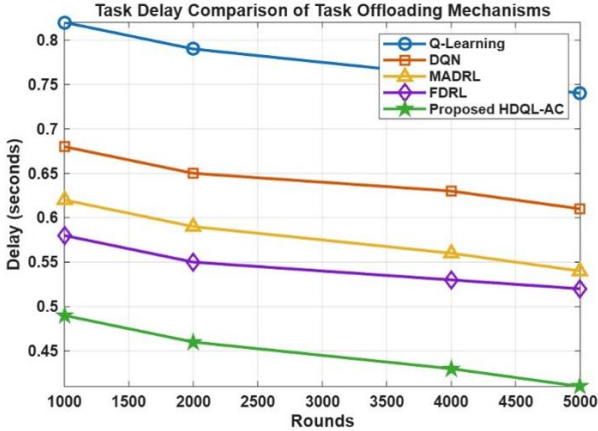


Figure 5. Delay comparison for task offloading mechanisms

5. CONCLUSION

In this paper, we introduce a new H-DQAC framework for QoS-aware task offloading in Fog-IoT settings. The framework integrates DQN for discrete task offloading choices with Actor–Critic to provide a stable and efficient resource allocation mechanism further enhanced with a pre-filter heuristic component for an efficient reduction of the action space. Our main objective was to balance and optimize each component of the frame design for the joint objectives of latency, energy consumption, task success ratio, throughput, and overall QoS utility. The proposed framework directly addresses the drawbacks or limitations of existing DRL based approaches that are all single-agent, multi-agent, or hybrid, while creating the potential to address combinations of each type of approach. We performed extensive simulations under dynamic IoT workloads and heterogeneous fog resources. Results showed that the proposed framework outperformed the respective state-of-the-art baselines; it provided a 38.3% reduction in latency, 23.2% reduction in energy usage, 22.7% improvement to the task success ratio, 30.3% improvement to the overall throughput, as well as a 40.3% increase to the QoS utility when compared to classical Q-Learning. These results are an indication or demonstration of how the framework was able to balance the need to adaptively explore versus exploit as means of efficient resource allocation while also providing reliable task offloading in latency and energy sensitive environments. The future work will consider the integration of federated learning and security-aware offloading, as well as transitioning the framework into real-time smart city or vehicular IoT contexts to further broaden the applicability of the present framework.

REFERENCES

- [1] Gupta, S., Singh, N. (2022). Fog-GMFA-DRL: Enhanced deep reinforcement learning with hybrid grey wolf and modified moth flame optimization to enhance the load balancing in the fog-IoT environment. *Advances in Engineering Software*, 174: 103295. <https://doi.org/10.1016/j.advengsoft.2022.103295>
- [2] Joshi, N., Srivastava, S. (2023). QoS-aware task allocation and scheduling in three-tier cloud-fog-IoT architecture using double auction. In *Proceedings of the 13th International Conference on Cloud Computing and*

- Services Science (CLOSER 2023), Prague, Czech Republic, pp. 253-260. <https://doi.org/10.5220/0011967400003488>
- [3] Chalapathi, G.S.S., Chamola, V., Vaish, A., Buyya, R. (2021). Industrial internet of things (iiot) applications of edge and fog computing: A review and future directions. *Fog/Edge Computing for Security, Privacy, and Applications*, 83: 293-325. https://doi.org/10.1007/978-3-030-57328-7_12
 - [4] Peng, P., Lin, W., Wu, W., Zhang, H., Peng, S., Wu, Q., Li, K. (2024). A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches. *Computer Science Review*, 53: 100656. <https://doi.org/10.1016/j.cosrev.2024.100656>
 - [5] Pakmehr, A. (2024). Task offloading in fog computing with deep reinforcement learning: Future research directions based on security and efficiency enhancements. *arXiv preprint arXiv:2407.19121*. <https://doi.org/10.48550/arXiv.2407.19121>
 - [6] Allaoui, T., Gasmi, K., Ezzedine, T. (2024). Reinforcement learning based task offloading of IoT applications in fog computing: Algorithms and optimization techniques. *Cluster Computing*, 27(8): 10299-10324. <https://doi.org/10.1007/s10586-024-04518-z>
 - [7] Mashal, H., Rezvani, M.H. (2024). Multiobjective offloading optimization in fog computing using deep reinforcement learning. *Journal of Computer Networks and Communications*, 2024(1): 6255511. <https://doi.org/10.1155/2024/6255511>
 - [8] Dayong, W., Bakar, K.B.A., Isyaku, B., Eisa, T.A.E., Abdelmaboud, A. (2024). A comprehensive review on internet of things task offloading in multi-access edge computing. *Heliyon*, 10(9): e29916. <https://doi.org/10.1016/j.heliyon.2024.e29916>
 - [9] Abdelghany, H.M. (2025). Dynamic resource management and task offloading framework for fog computing. *Journal of Grid Computing*, 23(2): 19. <https://doi.org/10.1007/s10723-025-09804-7>
 - [10] Tomar, A., Sharma, M., Agarwal, A., Jha, A.N., Jaiswal, J. (2025). Task offloading of IOT device in fog-enabled architecture using deep reinforcement learning approach. *Pervasive and Mobile Computing*, 112: 102067. <https://doi.org/10.1016/j.pmcj.2025.102067>
 - [11] Razaq, M.M., Rahim, S., Tak, B., Peng, L. (2022). Fragmented task scheduling for load-balanced fog computing based on Q-learning. *Wireless Communications and Mobile Computing*, 2022(1): 4218696. <https://doi.org/10.1155/2022/4218696>
 - [12] Wang, Z., Goudarzi, M., Gong, M., Buyya, R. (2024). Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. *Future Generation Computer Systems*, 152: 55-69. <https://doi.org/10.1016/j.future.2023.10.012>
 - [13] Duan, W., Li, X., Huang, Y., Cao, H., Zhang, X. (2024). Multi-agent-deep-reinforcement-learning-enabled offloading scheme for energy minimization in vehicle-to-everything communication systems. *Electronics*, 13(3): 663. <https://doi.org/10.3390/electronics13030663>
 - [14] Chen, M., Guo, A., Song, C. (2023). Multi-agent deep reinforcement learning for collaborative task offloading in mobile edge computing networks. *Digital Signal Processing*, 140: 104127. <https://doi.org/10.1016/j.dsp.2023.104127>
 - [15] He, H., Yang, X., Mi, X., Shen, H., Liao, X. (2024). Multi-agent deep reinforcement learning based dynamic task offloading in a device-to-device mobile-edge computing network to minimize average task delay with deadline constraints. *Sensors*, 24(16): 5141. <https://doi.org/10.3390/s24165141>
 - [16] Abbas, N., Sharafeddine, S., Mourad, A., Abou-Rjeily, C., Fawaz, W. (2022). Joint computing, communication and cost-aware task offloading in D2D-enabled Het-MEC. *Computer Networks*, 209: 108900. <https://doi.org/10.1016/j.comnet.2022.108900>
 - [17] Dai, X., Luo, Z., Zhang, W. (2024). Computation offloading with privacy-preserving in multi-access edge computing: A multi-agent deep reinforcement learning approach. *Electronics*, 13(13): 2655. <https://doi.org/10.3390/electronics13132655>
 - [18] Nieto, G., De la Iglesia, I., Lopez-Novoa, U., Perfecto, C. (2024). Deep reinforcement learning techniques for dynamic task offloading in the 5G edge-cloud continuum. *Journal of Cloud Computing*, 13(1): 94. <https://doi.org/10.1186/s13677-024-00658-0>
 - [19] Zhao, H., Li, Y., Pang, Z., Ma, Z. (2025). Federated multi-agent DRL for task offloading in vehicular edge computing. *Electronics*, 14(17): 3501. <https://doi.org/10.3390/electronics14173501>
 - [20] Toopchinezhad, M.P., Ahmadi, M. (2025). Deep reinforcement learning for delay-optimized task offloading in vehicular fog computing. In 2025 29th International Computer Conference, Computer Society of Iran (CSICC), Iran, Islamic Republic of, pp. 1-6. <https://doi.org/10.1109/CSICC65765.2025.10967415>
 - [21] Park, J., Chung, K. (2023). Distributed DRL-based computation offloading scheme for improving QOE in edge computing environments. *Sensors*, 23(8): 4166. <https://doi.org/10.3390/s23084166>
 - [22] Huang, C., Chen, G., Xiao, P., Xiao, Y., Han, Z., Chambers, J.A. (2024). Joint offloading and resource allocation for hybrid cloud and edge computing in SAGINs: A decision assisted hybrid action space deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 42(5): 1029-1043. <https://doi.org/10.1109/JSAC.2024.3365899>
 - [23] Choppara, P., Mangalampalli, S. (2025). An efficient deep reinforcement learning based task scheduler in cloud-fog environment. *Cluster Computing*, 28(1): 67. <https://doi.org/10.1007/s10586-024-04712-z>
 - [24] Raju, M.R., Mothku, S.K., Somesula, M.K. (2024). SMITS: Social and mobility aware intelligent task scheduling in vehicular fog computing—A federated DRL approach. *Computer Communications*, 222: 13-25. <https://doi.org/10.1016/j.comcom.2024.04.023>
 - [25] Thanedar, M.A., Panda, S.K. (2024). Energy and priority-aware scheduling algorithm for handling delay-sensitive tasks in fog-enabled vehicular networks. *Journal of Supercomputing*, 80(10): 14346-14368. <https://doi.org/10.1007/s11227-024-06004-0>
 - [26] Hussain, M., Nabi, S., Hussain, M. (2024). RAPTS: Resource aware prioritized task scheduling technique in heterogeneous fog computing environment. *Cluster Computing*, 27(9): 13353-13377. <https://doi.org/10.1007/s10586-024-04612-2>