International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

SLA-Priority Reinforcement Learning-Based Resource Allocation Framework for IoMT-Enabled Healthcare Fog Computing

Nagarjun E[1], Dharamendra Chouhan[2]

[1]Research Scholar, Department of CSE, UVCE, Bangalore University.
[2]Associate Professor, Department of Computer Science and Engineering, University of Visvesvaraya College of Engineering

Abstract
The rapid growth of the Internet of Medical Things (IoMT) has improved real-time health monitoring, but it has also created serious challenges in latency, energy consumption, bandwidth use, and adaptive resource management. Traditional cloud-only architectures require patient data to travel to distant data centers, which increases response time and can reduce the usefulness of urgent clinical alerts. Fog computing addresses this problem by placing computation close to medical sensors and hospital gateways. However, fog nodes have limited resources, and healthcare workloads change continuously. To address this issue, this study proposes a reinforcement learning based resource allocation framework for healthcare fog environments. The proposed model observes CPU utilization, memory utilization, network latency, patient load, and module placement, then selects actions such as scaling, migration, load balancing, or maintaining the current allocation. The model is evaluated against deep-learning predictor allocation, tabular Q-learning, and DQN-style DRL policies implemented in the accompanying reproducible experiment. The proposed SLA-priority RL method achieves the best overall performance among the compared strategies. It records the lowest average latency of 78.57 ms, compared with 79.75 ms for DL predictor allocation, 80.00 ms for tabular Q-learning, and 79.50 ms for DQN-style DRL. It also achieves the lowest energy index of 341.92 J and maintains 0.00% SLA violations, while the DL predictor, tabular Q-learning, and DQN-style DRL methods record SLA violations of 0.82%, 0.86%, and 0.81%, respectively. These results indicate that the proposed reward design improves responsiveness, energy efficiency, and SLA reliability for healthcare fog resource allocation.

Keywords: Fog computing, IoMT, Healthcare resource allocation, Reinforcement learning, Latency reduction.

1. Introduction

IoMT systems connect wearable devices, bedside sensors, smart hospital equipment, and mobile health applications to support continuous patient monitoring. These systems collect measurements such as heart rate, blood pressure, oxygen saturation, glucose level, respiratory rate, and activity state. In many healthcare situations, the usefulness of the data depends not only on accuracy but also on response time. A delayed emergency notification can reduce the value of real-time monitoring even when the measured signal is correct.

Cloud computing provides elastic storage and large-scale processing, but it is not always appropriate for time-sensitive healthcare events. Transmitting every patient tuple to a remote cloud can increase latency and bandwidth consumption [1], [2]. Fog computing extends cloud services toward the edge of the network, allowing local gateways and micro data centers to process data near the point of care [3], [4], [5]. For healthcare, this means that urgent patient events can be filtered, analyzed, and acted upon before long-term summaries are sent to the cloud [6].

The main challenge is that fog environments are distributed and resources constrained. A fog node may have enough capacity during normal monitoring but become overloaded during emergency bursts [7], [8]. Non-adaptive allocation is therefore inefficient because unchanged module placement cannot react to changes in patient load, network latency, or fog-node utilization. Reinforcement learning is suitable for this problem because an agent learns from interaction with the environment and improves its allocation policy based on rewards and penalties [14], [15], [16].

Contributions

The contributions of this research are as follows:

- A complete fog-based healthcare resource allocation architecture is proposed for IoMT monitoring.
- An end-to-end processing flow is defined from sensor data generation to fog processing, RL decision making, alert delivery, and cloud storage.
- A mathematical model is provided for latency, communication time, energy, utilization balance, and optimization constraints.
- A Q-learning based algorithm is described with state space, action space, reward design, and update rule.

2. Related Works

Nagarjun et al. [9] proposed an agent-based resource provisioning algorithm was proposed for fog computing environments using the Contract Net Protocol (CNP). The approach dynamically assigns tasks to virtual machines according to available resources and task requirements. In a smart healthcare case study, the CNP-based method reduced network usage by 22.95% compared with SJF and 28.67% compared with FCFS. It also lowered energy consumption by 6% against both baselines and reduced loop delay by 19% compared with SJF and 38.48% compared with FCFS.

Tuli et al. [10] introduced HealthFog, a fog-enabled healthcare framework that applies ensemble deep learning on edge computing devices for automatic heart disease analysis. The system treats healthcare analytics as a fog service, where IoT-generated heart patient data is handled as user requests and processed closer to the data source. The authors used the FogBus framework to deploy and evaluate HealthFog, measuring performance through power consumption, bandwidth usage, latency, jitter, prediction accuracy, and execution time. A key strength of HealthFog is its configurable operating modes, which allow the system to prioritize either quality of service or diagnostic accuracy depending on the fog-computing scenario and user requirements.

Ming Tang et al.[11] proposed a model-free distributed deep reinforcement learning algorithm for task offloading in edge computing environments. The method enables each device to make independent offloading decisions without requiring prior knowledge of task models or the offloading choices of other devices. To improve long-term cost estimation, the algorithm combines LSTM, dueling DQN, and double-DQN techniques. Simulation results show improved utilization of edge-node processing capacity, along with lower task-dropping ratio and average delay compared with existing offloading algorithms.

Huang et al. [12] proposed the Deep Reinforcement learning-based Online Offloading (DROO) framework for wireless powered mobile-edge computing networks. The framework addresses binary task offloading, where each wireless device either processes a task locally or fully offloads it to an MEC server. DROO uses a deep neural network to learn offloading decisions from experience, avoiding repeated combinatorial optimization during rapidly changing channel conditions. An adaptive parameter-adjustment procedure is also included to reduce computational complexity during execution. Numerical results show that DROO achieves near-optimal offloading performance while reducing computation time by more than an order of magnitude compared with conventional optimization methods. In a 30-user network, the reported CPU execution latency is below 0.1 seconds, demonstrating suitability for real-time offloading in fast-fading wireless environments.

Zhou et al. [13] proposed an uplink and downlink rate-splitting transmission scheme for fog-enabled Internet of Medical Things systems. The scheme uses rate-splitting to manage interference and reduce delays caused by task offloading and feedback transmission. Offloading decisions, computing-resource allocation, uplink and downlink beamforming, and common-rate allocation are jointly optimized to minimize total time cost. Since the formulated optimization problem is non-convex, quadratic transform techniques and first-order Taylor approximation are applied to reformulate the objective and constraints into a tractable form. The reformulated problem is then solved through a two-stage alternating optimization approach. Simulation results show that the proposed scheme outperforms several benchmark methods.

Wu et al. [17] investigated latency-aware resource allocation in mobile edge generation and computing, where mobile edge computing is combined with generative artificial intelligence services. The study formulates a latency-minimization problem involving joint communication, computation, and AIGC resource allocation. Because the decision variables are strongly coupled, a deep reinforcement learning-based algorithm is used to solve the problem efficiently. Numerical results show that the proposed method achieves lower latency than several baseline algorithms, demonstrating the usefulness of DRL for complex edge-resource allocation scenarios.

Ranani et al. [18] proposed a Q-learning-based resource prediction and allocation method for a three-layer vehicular fog computing architecture. The method estimates the memory, bandwidth, and processor resources required by intelligent vehicle clients and uses reinforcement learning to support adaptive resource-management decisions. By learning from previous allocation experiences and updating the Q-learning agent over time, the system can respond to changing fog conditions using real-time information. Experimental results show that the approach improves resource allocation efficiency, reduces unnecessary resource consumption, and enhances average task processing time compared with the evaluated baseline methods.

3. System Model, Problem Statement and Objectives

3.1 System Model

The proposed system contains edge, fog, optimizer, and cloud components. Edge devices generate patient tuples. Fog nodes execute real-time patient data and analysis modules. The optimizer observes the fog environment and selects resource allocation actions. The cloud stores non-urgent summaries and supports offline analysis.

3.2 Architecture of the Model

Layer	Components	Function
Edge layer	Wearable sensors, bedside monitors, actuators	Generate patient data and receive urgent alerts
Fog layer	Hospital gateways and local fog nodes	Process filtered tuples and run analysis modules
Optimizer layer	State monitor, Q-learning agent, reward calculator	Select resource scaling, migration and load balancing actions
Cloud layer	Remote storage and analytics servers	Store historical records and perform non-real-time analytics

The end-to-end operation begins when a sensor generates a patient tuple. The nearest fog node receives the tuple and executes filtering. The analysis module then assigns priority to the tuple. The optimizer reads current CPU utilization, memory utilization, latency, patient load, and module placement. It chooses an action and applies it to the fog system. The resulting performance is measured and converted into a reward for the next learning step.

3.3 Problem Statement

Based on the related-work gaps, the problem addressed in this paper is the lack of an adaptive, healthcare-aware fog resource allocation method that can reduce latency and SLA violations while maintaining balanced fog-node utilization. Given a set of IoMT sensors, fog nodes, application modules, and dynamic patient workloads, the system must decide where to process data and how to allocate CPU and memory at each decision interval.

3.4 Objectives

- To design a healthcare fog architecture that processes urgent IoMT data close to patients.
- To formulate healthcare resource allocation as a reinforcement learning problem driven by observable fog-state metrics.
- To minimize response latency and SLA violations under normal and burst workload conditions while monitoring the energy trade-off.
- To improve CPU and memory utilization without overloading individual fog nodes.
- To compare the proposed adaptive method with DL predictor allocation, tabular Q-learning and DQN-style DRL baselines.

3.5 Mathematical Model

Let D_i represent data generated by the i^{th} medical sensor, C_f represent fog processing capacity, B_{if} represent bandwidth between the sensor and fog node, L_{if} represent link latency, Q_i represent queueing delay, and E_i represent energy consumed for processing. The tuple processing time is defined as:

$$T_{proc(i)} = D_i / C_f \quad (1)$$

The communication time is:

$$T_{comm(i)} = D_i / B_{if} + L_{if} \quad (2)$$

The total response time is:

$$T_{total(i)} = T_{proc(i)} + T_{comm(i)} + Q_i \quad (3)$$

The resource allocation objective is:

$$\min_J = w_1 * T_{total} + w_2 * E_{total} + w_3 * SLA_{viol} + w4 * U_{imbalance}$$

Optimization is subject to the following constraints:

- CPU and memory allocation must not exceed fog-node capacity.
- Critical tuple latency should remain below the SLA threshold.
- Bandwidth consumption must remain within available link capacity.
- Non-critical tasks may be delayed or migrated when urgent patient tuples arrive.

4. Proposed Method

The proposed model formulates resource allocation as a Markov decision process. The state contains CPU utilization, memory utilization, network latency, patient load, queue level, and module placement. The action set contains CPU scaling, memory scaling, module migration, load balancing, and no change. The reward function encourages low latency, controlled energy use, balanced utilization, and SLA satisfaction. Because patient monitoring is time-sensitive, the final policy prioritizes SLA-safe latency while still reporting energy as a secondary performance objective.

MDP element	Definition
State	CPU bin, memory bin, latency bin, queue bin, patient-load bin and placement state
Action	Scale CPU, scale memory, migrate module, balance load or keep current allocation
Reward	Positive reward for low latency and SLA satisfaction; penalty for overload and energy increase
Policy	Epsilon-greedy action selection over Q-values
Update rule	Q(s,a) <- Q(s,a) + alpha [r + gamma max Q(s',a') - Q(s,a)]

The proposed algorithm is executed as follows:

- Initialize fog topology, application modules, sensors, actuators and Q-table.
- Collect CPU, memory, latency, queue length, patient load and placement state at each interval.
- Discretize the observed values into the current RL state.
- Select an action using epsilon-greedy exploration and exploitation.
- Apply the action to the fog environment.
- Measure new latency, energy, utilization and SLA status.
- Calculate reward and update the Q-table.
- Repeat until the simulation horizon is completed.

4.1 End-to-End Method Flow

The method begins with patient tuple generation at the edge layer. Each tuple is tagged as normal, warning, or critical based on its workload priority. Normal tuples can be aggregated or delayed briefly, while warning and critical tuples are sent to nearby fog nodes for immediate processing. The fog monitor then reports current CPU load, memory load, queue length, and latency to the optimizer. The optimizer converts these measurements into the RL state, selects an action, and applies that action by scaling resources, migrating a module, balancing load, or keeping the current placement. After the next interval, the system calculates reward from latency, energy index, utilization, and SLA status. This closed loop continues throughout the simulation.

4.2 Logic Method for Latency Reduction

The first problem is to reduce the total response time of patient tuples. In cloud-only or static fog systems,

queueing and transmission delay can increase during bursts. The proposed RL model reduces latency by moving computation toward available fog resources and giving priority to urgent tuples.

4.3 Method Logic for Adaptive Resource Utilization

The second problem is to maintain balanced utilization. Under-utilization wastes fog resources, while over-utilization increases queueing delay and SLA violations. The proposed reward function penalizes overload and encourages the agent to keep CPU and memory usage within a productive operating range.

5 Results and Discussion

5.1 Evaluation Matrix

Metric	Definition	Purpose
Latency	Average response time from sensor tuple generation to response	Measures real-time suitability
Energy index	Estimated energy consumed by fog processing and allocation behavior	Measures efficiency
CPU utilization	Average fog CPU use	Measures resource usage
Memory utilization	Average fog memory use	Measures allocation balance
SLA violation rate	Percentage of tuples exceeding the latency threshold	Measures reliability

5.2 Experimental Setup

Setting	Value
Simulation steps	10,000
Random seed	42
Training steps	6,000
SLA threshold	120 ms
Workload	Normal monitoring with periodic and emergency bursts
Compared methods	DL predictor allocation, tabular Q-learning, DQN-style DRL and proposed SLA-priority RL
Output metrics	Latency, energy index, CPU utilization, memory utilization and SLA violation rate

Topology / Task Parameter	Value Used in Experiment
IoMT workload streams	One aggregated patient-monitoring stream representing normalized ward-level demand
Logical fog nodes	Four allocation capacity levels representing low, balanced, scale-up and migrate-balance fog states
Application tasks/modules	Patient-data filtering, data-analysis/risk scoring and resource-optimizer control
Task priority classes	Normal, warning and critical tuples
Action set	fixed-low, balanced, scale-up and migrate-balance
Capacity levels	0.52, 0.68, 0.82 and 0.92 normalized processing capacity
Emergency bursts	Steps 2,500-3,999 and 6,800-7,599

Topology / Task Parameter	Value Used in Experiment
Periodic burst interval	First 180 steps of each 900-step cycle
State bins	Six workload-state bins derived from normalized patient load

Learning Parameter	DL Predictor	Tabular Q-learning	DQN-style DRL	Proposed SLA-priority RL
Training samples	First 6,000 steps	First 6,000 steps	First 6,000 steps	First 6,000 steps
Input/state	Six-step workload window	Six discretized load states	Load, state-bin value and high-load flag	Six discretized load states with SLA-priority guardrail
Model structure	MLP, 6 input units, 10 hidden units, 1 output	Q-table with 6 states x 4 actions	Neural Q approximator, 3 input units, 12 hidden units, 4 outputs	Reward-shaped Q-table with 6 states x 4 actions
Learning rate	0.018	0.14	0.010	0.22
Discount factor	Not applicable	0.82	0.86	0.92
Training iterations	650 epochs	30 episodes	55 epochs	80 episodes
Exploration	Not applicable	Epsilon-greedy, initial 0.30, minimum 0.04	Epsilon-greedy, initial 0.25, minimum 0.03	Epsilon-greedy, initial 0.30, minimum 0.04

5.3 Comparative Results

Strategy	Avg. latency (ms)	Energy index (J)	CPU util. (%)	Memory util. (%)	SLA violations (%)
DL predictor allocation [10]	79.75	343.82	82.61	81.07	0.82
Tabular Q-learning [20]	80.00	362.90	73.96	75.25	0.86
DQN-style DRL [11], [17]	79.50	355.36	77.29	77.50	0.81
Proposed SLA-priority RL	78.57	341.92	81.26	80.16	0.00

The comparative results demonstrate that the proposed SLA-priority RL method achieves the strongest overall performance among the evaluated strategies. The DL predictor allocation [10] records an average latency of 79.75 ms and an energy index of 343.82 J, but it still produces 0.82% SLA violations. This indicates that prediction-based allocation can reduce energy consumption, but it may not fully satisfy strict real-time healthcare requirements.

Tabular Q-learning [20] achieves an average latency of 80.00 ms with an energy index of 362.90 J and 0.86% SLA violations. This shows that basic Q-learning can support adaptive decision-making, but its performance is limited in terms of latency, energy efficiency, and SLA compliance. Similarly, the DQN-style DRL method [11], [17] improves latency to 79.50 ms compared with tabular Q-learning, but it still records 0.81% SLA violations and a relatively high energy index of 355.36 J.

In contrast, the proposed SLA-priority RL method achieves the lowest average latency of 78.57 ms, the lowest energy index of 341.92 J, and 0.00% SLA violations. The absence of SLA violations is particularly important in healthcare fog environments, where delayed responses can affect time-critical monitoring

and alert generation. These results show that the proposed method provides a better balance between latency reduction, energy efficiency, resource utilization, and service reliability than the compared DL, Q-learning, and DRL-based approaches.

5. Conclusion and Future Work

This paper proposed a reinforcement learning based fog computing architecture for healthcare resource allocation in IoMT systems. The model processes urgent patient data near the source, observes fog resource conditions, and uses Q-learning to select allocation actions. The mathematical model, evaluation matrix, problem formulation, and result analysis show how the system works end to end. In the reproducible experiment, the proposed SLA-priority RL method achieves the best overall performance among the compared strategies. It records the lowest average latency of 78.57 ms, compared with 79.75 ms for DL predictor allocation, 80.00 ms for tabular Q-learning, and 79.50 ms for DQN-style DRL. It also achieves the lowest energy index of 341.92 J and maintains 0.00% SLA violations, while the DL predictor, tabular Q-learning, and DQN-style DRL methods record SLA violations of 0.82%, 0.86%, and 0.81%, respectively. These results indicate that the proposed reward design improves responsiveness, energy efficiency, and SLA reliability for healthcare fog resource allocation. Future work will implement the same policy in a complete iFogSim2 Java environment, evaluate larger hospital topologies, and compare the Q-learning method with DQN and actor-critic algorithms.

6. References

1. Nagarjun, E., Dharamendra Chouhan, and S. M. Dilip Kumar. "Cost-Efficient Hybrid Fog Resource Provisioning Using MKFCM and Flower Pollination Algorithm." In *International Conference on Frontiers in Computing and Systems*, pp. 345-356. Singapore: Springer Nature Singapore, 2024.
2. OpenFog, C. "Openfog reference architecture for fog computing." *URL: https://www. openfogconsortium. org/ra* (2017).
3. Nagarjun E, Dharamendra Chouhan and Dilip Kumar S M. “Multi-Objective Osprey Optimization Algorithm-Based Resource Allocation in Fog-IoT”. International Journal of Advanced Computer Science and Applications (IJACSA) 16.2(2025). http://dx.doi.org/10.14569/IJACSA.2025.01602122
4. N. E, D. Chouan and S. Dilip Kumar, "Reputation Based Resource Allocation for IoT in Fog Computing Environments," *2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS)*, Raichur, India, 2025, pp. 1-6, doi: 10.1109/ICICACS65178.2025.10967829.
5. Iorga, Michaela, Larry Feldman, Robert Barton, Michael J. Martin, Nedim S. Goren, and Charif Mahmoudi. "Fog computing conceptual model." (2018).
6. Dubey, Harishchandra, Admir Monteiro, Nicholas Constant, Mohammadreza Abtahi, Debanjan Borthakur, Leslie Mahler, Yan Sun, Qing Yang, Umer Akbar, and Kunal Mankodiya. "Fog computing in medical internet-of-things: architecture, implementation, and applications." In *Handbook of large-scale distributed computing in smart healthcare*, pp. 281-321. Cham: Springer International Publishing, 2017.
7. Andrew, Barto, and Sutton Richard S. "Reinforcement learning: an introduction." (2018).
8. Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8, no. 3 (1992): 279-292.

9. E. Nagarjun, D. Chouhan, I. Zabiulla and S. M. D. Kumar, "Efficient Resource Provisioning in Fog Computing Using Agent-Based Contract Net Protocol: A Smart Healthcare Case Study," *2024 First International Conference on Software, Systems and Information Technology (SSITCON)*, Tumkur, India, 2024, pp. 1-6, doi: 10.1109/SSITCON62437.2024.10796063.
10. Tuli, Shreshth, Nipam Basumatary, Sukhpal Singh Gill, Mohsen Kahani, Rajesh Chand Arya, Gurpreet Singh Wander, and Rajkumar Buyya. "HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments." *Future Generation Computer Systems* 104 (2020): 187-200.
11. Tang, Ming, and Vincent WS Wong. "Deep reinforcement learning for task offloading in mobile edge computing systems." *IEEE transactions on mobile computing* 21, no. 6 (2020): 1985-1997.
12. Huang, Liang, Suzhi Bi, and Ying-Jun Angela Zhang. "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks." *IEEE Transactions on Mobile Computing* 19, no. 11 (2019): 2581-2593.
13. J. Zhou, Y. Chen, C. Zhou, and Y. Sun, "Zhou, Jiasi, Yan Chen, Cong Zhou, Yanjing Sun, and Chintha Tellambura. "Joint uplink and downlink rate splitting for fog-computing-enabled internet of medical things." *IEEE Internet of Things Journal* 11, no. 23 (2024): 37872-37884.
14. Jayanetti, Amanda, Saman Halgamuge, and Rajkumar Buyya. "Reinforcement learning based workflow scheduling in cloud and edge computing environments: A taxonomy, review and future directions." *arXiv preprint arXiv:2408.02938* (2024).
15. Xu, Jingyu, Weixiang Wan, Linying Pan, Wenjian Sun, and Yuxiang Liu. "The fusion of deep reinforcement learning and edge computing for real-time monitoring and control optimization in IoT environments." In *2024 3rd International Conference on Energy and Power Engineering, Control Engineering (EPECE)*, pp. 193-196. IEEE, 2024.
16. Cai, Huaiguang, Zhi Zhou, and Qianyi Huang. "Online resource allocation for edge intelligence with colocated model retraining and inference." In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pp. 1900-1909. IEEE, 2024.
17. Wu, Yinyu, Xuhui Zhang, Jinke Ren, Huijun Xing, Yanyan Shen, and Shuguang Cui. "Latency-aware resource allocation for mobile edge generation and computing via deep reinforcement learning." *IEEE networking letters* 6, no. 4 (2024): 237-241.
18. B. M. Ranani, M. Ahmadi, and S. Ahmadian, "A Q-learning approach for dynamic resource management in three-tier vehicular fog computing," *arXiv:2602.14390*, 2026.