frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 12 January 2026
DOI 10.3389/fcomp.2025.1723498

OPEN ACCESS

EDITED BY
Zhou Zhou,
Changsha University, China

REVIEWED BY
Sun Yanglong,
Jimei University, China
Seyed Omid Azarkasb,
K.N.Toosi University of Technology, Iran

*CORRESPONDENCE
Anju Babu
anjuedathadan@gmail.com

RECEIVED 12 October 2025
REVISED 03 December 2025
ACCEPTED 10 December 2025
PUBLISHED 12 January 2026

CITATION
Babu A and Bala GJ (2026) Latency and trust constrained fog node selection using deep reinforcement learning.
Front. Comput. Sci. 7:1723498.
doi: 10.3389/fcomp.2025.1723498

Latency and trust constrained fog node selection using deep reinforcement learning

Anju Babu[1,2]* and G. Josemin Bala[1]

[1]Karunya Institute of Technology and Sciences, Coimbatore, India, [2]Sahrdaya College of Engineering and Technology, Kodakara, India

Automated healthcare IoT systems demand secure, low-latency, and energy-efficient computation—capabilities well-supported by fog computing. Effective selection of fog nodes is critical for maximizing the performance of fog-based IoT platforms. This paper introduces a Secure Proximal Policy Optimization (Secure PPO) algorithm for trust-aware fog node selection, considering latency, energy consumption, processing power, and a trust flag for each node. Secure PPO enforces a trust constraint while optimizing latency and energy via PPO's clipped objective, ensuring stable and reliable learning. Simulation results demonstrate that Secure PPO achieves substantial improvements over A2C and Deep Q-Networks (DQN). Specifically, Secure PPO reduces inference latency by 24.36 and 37.57%, lowers convergence time by 55.56 and 66.67%, and decreases energy consumption by 11.90 and 20.04% compared to A2C and DQN, respectively. Additionally, Secure PPO improves accuracy by 9.42 and 18.88% over A2C and DQN. The framework maintains sub-millisecond inference time and ensures secure, reliable fog-based execution of automated healthcare tasks, substantially enhancing patient safety and operational efficiency within healthcare IoT environments.

KEYWORDS

deep reinforcement learning, energy efficiency, fog computing, latency reduction, secure PPO, trust-aware node selection

1 Introduction

Fog computing has emerged as a critical paradigm to address the increasing demand for low latency and localized service provisioning in modern computing environments, particularly with the proliferation of Internet of Things (IoT) devices (Alkhalaf and Hussain, 2023). This architecture extends cloud services closer to end-users, reducing communication delays that are often prohibitive for time-sensitive applications when relying solely on remote cloud servers (La et al., 2019). However, selecting optimal fog nodes for task offloading presents significant challenges due to the distributed nature of fog environments, the varying computational capabilities of nodes, and the imperative for secure and reliable operations (Roshan et al., 2020). To overcome these challenges, Deep Reinforcement Learning (DRL) algorithms, such as Proximal Policy Optimization (PPO), have shown promise in optimizing resource allocation and task offloading decisions, while the integration of trust models ensures that only reliable and secure fog nodes are chosen for computations (Goudarzi et al., 2023).

Deep Reinforcement Learning (DRL) algorithms are increasingly being applied in fog computing to address complex optimization problems, including task offloading and resource allocation (Allaoui et al., 2024). DRL models can learn optimal strategies by interacting with the dynamic fog environment, using neural networks to approximate

value functions or policies when the state-action space is large (Rahman et al., 2020). The Proximal Policy Optimization (PPO) algorithm is a particularly effective DRL technique known for its stable and efficient learning, making it well-suited for dynamically selecting fog nodes that minimize latency and optimize resource usage (Nagabushnam et al., 2025). PPO-based approaches can make intelligent decisions on where to process tasks—locally, at a nearby fog node, or in the cloud—considering various factors such as available resources, network conditions, and task deadlines (Rahman et al., 2020). This capability allows DRL to achieve significant reductions in overall delay and energy consumption.

In intelligent healthcare systems, decision accuracy and security are equally critical. While fog computing enables low-latency processing close to patients, it also introduces vulnerabilities due to heterogeneous and potentially untrusted nodes. A malicious or compromised fog node could cause unsafe control decisions, posing severe risks to patient safety. Hence, integrating a trust evaluation mechanism into the resource selection process is vital.

Traditional rule-based trust schemes are often static and lack the ability to adapt to the dynamic nature of fog environments. Conversely, Proximal Policy Optimization (PPO), a reinforcement learning algorithm known for its stability and sample efficiency, can autonomously learn optimal decision-making policies under varying network conditions. The integration of PPO with a multi-layered trust mechanism allows the system to simultaneously achieve security, energy efficiency, and real-time responsiveness, making it well-suited for safety-critical healthcare applications.

To ensure secure and reliable fog node selection, this paper employs a hybrid trust mechanism combining a static trusted flag with a dynamic reputation-based trust score. The trusted flag serves as an initial security filter to eliminate unverified or potentially malicious fog nodes. In parallel, the reputation metric continuously evaluates node behavior using parameters such as task success rate, response consistency, and reliability over time. The combined approach ensures that only verified and consistently reliable fog nodes are chosen by the PPO agent, maintaining both security integrity and operational performance. This layered trust assessment forms the foundation of the proposed Secure PPO framework.

This paper introduces an approach that leverages the PPO algorithm for intelligent fog node selection, specifically focusing on reducing latency by ensuring tasks are processed by the most suitable and trusted nodes within the fog computing infrastructure (Nagabushnam et al., 2025).

Beyond performance metrics like latency, the trustworthiness of fog nodes is a crucial consideration for reliable and secure fog computing environments (Alkhalaf and Hussain, 2023). Fog nodes, especially in cooperative settings like smart cities, must be evaluated for their reliability to prevent faulty nodes or malicious attackers from compromising communication and data processing. Trust models based on various factors, including service level agreement parameters and reputation values, are essential for identifying trusted nodes. Techniques such as fuzzy logic and blockchain have been proposed to establish and maintain trust in fog networks, ensuring data integrity, privacy, and secure task offloading. By integrating trust mechanisms with DRL-based node selection, the system can not only optimize for low latency and efficient resource allocation but also guarantee that tasks are handled by authenticated and dependable fog nodes, thereby enhancing the overall security and resilience of the fog computing infrastructure.

Key contributions of the paper are summarized as follows:

(1) We propose a Secure Proximal Policy Optimization (Secure-PPO) framework for intelligent fog node selection, integrating both performance optimization and security constraints.
(2) We design a hybrid trust mechanism that combines a static trusted-flag filter with a dynamic reputation-based trust score to ensure reliable and secure fog node participation.
(3) We incorporate the hybrid trust mechanism into the PPO learning workflow, enabling the agent to simultaneously optimize latency, reliability, and security during task offloading.
(4) We evaluate the proposed Secure-PPO model using a realistic fog computing environment and demonstrate that it significantly reduces latency while maintaining high trustworthiness and operational safety compared to baseline approaches.

Following sections of this research paper is arranged as follows. Section 2 gives an elaborated study on different methods, both heuristic and machine learning approaches in recent years. Section 3 defines the problem formulation and the system model. Section 4 describes the proposed methodology that is secure PPO approach. Section 5 discusses the results and its comparison with the timeline algorithms. Section 6 concludes the research work.

2 Related works

Task placement is always an NP hard problem. To solve these many methods are adopted each having its own advantages and disadvantages. Recent studies and researchers have found that relying on machine learning techniques especially Reinforcement learning and deep reinforcement learning solves this problem in an efficient manner there by improving the quality of service of IoT applications.

2.1 Placement through heuristic approach

Building upon the foundational methods, the period from 2020 to the present has seen a distinct temporal shift toward more sophisticated, dynamic, and intelligent heuristic and metaheuristic algorithms for fog service placement. Comprehensive surveys published in 2020 cataloged the growing landscape of optimization techniques, including exact methods, heuristics, and hybrid approaches (Raghavendra et al., 2020), and provided a set of heuristics to address the Service Placement Problem (SPP; Salaht et al., 2020). A significant trend during this phase was the increasing focus on multi-objective optimization. Researchers proposed methods like an ant colony optimization-based solution to balance deployment cost and service latency (Huang et al., 2020) and an Improved Parallel Genetic Algorithm (IPGA) that

maintains a set of Pareto solutions to make compromises between latency, cost, resource utilization, and service time (Wu et al., 2022). Another approach utilized the cuckoo search algorithm to solve the multi-objective problem (Liu et al., 2022). The field also evolved to address the dynamic nature of fog environments, with a move toward autonomic systems. For example, one study proposed an autonomic service placement approach using the gray wolf optimization scheme to enhance system performance and consider execution costs (Salimian et al., 2021), while a 2023 paper introduced a dynamic strategy that performs multi-objective optimization on response time and available resources to select the fittest node in real-time (Trabelsi et al., 2023). The most recent advancements integrate machine learning and novel hybrid metaheuristics, with forthcoming 2025 papers proposing a combination of reinforcement learning and an improved gray wolf optimization method to place services based on user request volumes (Ashkani et al., 2025) and a Hybrid Prairie Dog and Dwarf Mongoose Optimization Algorithm (HPDDMOARS) designed to optimize energy, cost, and make span (Baskar et al., 2025). This chronological progression demonstrates a clear trajectory toward creating intelligent, adaptive, and context-aware strategies capable of managing the complexities of modern fog ecosystems.

2.2 Placement through RL techniques

Recent studies have demonstrated the effectiveness of RL and DRL in optimizing service placement by enabling adaptive, data-driven decision-making. Several algorithms are used which gives different types of performance enhancements. Double Deep Q-Networks with prioritized experience replay (DDQN-PER) have been used to minimize service latency and energy consumption by learning dynamic resource requirements and efficiently mapping services to fog nodes (Sharma and Thangaraj, 2024). Actor-critic-based distributed DRL techniques, such as those leveraging the IMPALA architecture, address the scalability and adaptability issues of centralized DRL by distributing learning across multiple fog nodes, significantly reducing execution costs and improving placement efficiency (Goudarzi et al., 2021).

The Asynchronous Advantage Actor-Critic (A3C) algorithm has also been applied to both service function chain (SFC) placement and general IoT service placement, achieving notable improvements in cost, latency, and resource utilization compared to traditional heuristics and other DRL methods (Zhang et al., 2022; Zare et al., 2022). Deep Q-Networks (DQN) have been explored for value-based service placement, focusing on maximizing utility and adapting to changing network conditions (Poltronieri et al., 2021). In (Trabelsi et al., 2023) Q-learning for adaptive IoT service deployment that jointly optimizes latency and energy demonstrates how classic tabular RL can still be effective with careful state/action discretization.

Use of Double DQN (Zare et al., 2022) reduces overestimation bias and dynamically deploy IoT services to fog nodes to minimize response time and resource violation. It shows clear improvements over heuristics. A DRL framework (Lera and Guerrero, 2024) tackling multi-objective placement (latency, energy, cost); shows how reward shaping and multi-objective RL formulations can be used. In (Lera et al., 2018) Not strictly DRL, but a method that combines graph/network insights with sequential allocation gives a good pointer to hybrid approaches that incorporate structure-aware partitioning before learning-based placement.

Table 1 shows a summary of the literature review covered in this research.

2.3 Research gap and motivation

Although heuristic, metaheuristic, and reinforcement learning–based approaches have significantly advanced the fog service placement landscape, several critical limitations remain unaddressed—particularly in safety-critical healthcare IoT systems. Existing heuristic and metaheuristic methods (e.g., ACO, IPGA,

TABLE 1 Summary of recent literature on placement methods.

Year	Method/Algorithm	Objective(s)	Strengths	Limitations/Gap
2020	SPP Heuristics (Salaht et al., 2020)	Latency, cost	Simple, fast	Static rules; no trust; low adaptiveness
2021	ACO-based Placement (Huang et al., 2020)	Cost–latency trade-off	Multi-objective path search	No security; limited scalability
2021	IPGA (Wu et al., 2022)	Latency, service time, utilization	Pareto-optimal solutions	High computational cost
2022	Cuckoo Search (Liu et al., 2022)	Multi-objective optimization	Good exploration	No real-time capability
2023	Dynamic Multi-objective Strategy (Trabelsi et al., 2023)	Response time, resources	Real-time adaptive	No trust evaluation
2024	DDQN/A3C/DQN RL Models (Sharma and Thangaraj, 2024; Goudarzi et al., 2021; Zhang et al., 2022; Zare et al., 2022; Poltronieri et al., 2021; Lera and Guerrero, 2024)	Latency, energy, cost	Self-learning, adaptive	No trust/security; unstable in large node sets
2025	RL + GWO Hybrid (Ashkani et al., 2025)	Load-based placement	Hybrid optimization	No healthcare focus; no trust
2025	HPDDMOARS (Baskar et al., 2025)	Energy, cost, makespan	Strong multi-objective optimization	Heavy computation; no secure placement

cuckoo search, GWO, and hybrid bio-inspired algorithms) excel at multi-objective optimization but rely on fixed rules or predefined cost functions. This limits their ability to adapt to rapid fluctuations in patient-driven workloads, dynamic fog topologies, and real-time operational constraints. Moreover, these approaches typically lack built-in mechanisms to evaluate fog node trustworthiness or detect unreliable nodes, making them unsuitable for environments where compromised decisions may endanger patient safety.

Similarly, while recent RL/DRL frameworks demonstrate strong adaptability to dynamic fog environments, most of them optimize only latency, energy, or cost, with no explicit trust or security integration. Existing DRL models overlook the possibility of untrusted, faulty, or malicious fog nodes influencing task execution—an unacceptable risk in healthcare IoT applications, where incorrect computation directly affects patient health. Furthermore, current RL solutions rarely combine static trust indicators with dynamic behavioral trust scoring, nor do they evaluate algorithm stability or scalability when fog networks scale to hundreds of nodes.

These gaps highlight the need for an intelligent, secure, and adaptive fog node selection framework capable of jointly optimizing latency, energy, reliability, and trust. Motivated by the stringent real-time and security requirements of automated healthcare IoT systems, this work proposes a Secure Proximal Policy Optimization (Secure PPO) model incorporating both a trusted flag and a dynamic reputation-based trust score into the RL reward structure. By combining PPO's stable policy updates with a hybrid trust mechanism, the proposed framework aims to achieve:

1. strong trust compliance,
2. low-latency and energy-aware placement, and
3. scalable performance under dynamic, high-density fog environments.

This work therefore bridges the gap between conventional optimization strategies and trustworthy, deployment-ready RL models for real-time healthcare IoT.

3 System model and problem formulation

In healthcare IoT applications like patient monitoring systems, where a patient will be continuously monitored with the help of sensors , In emergency situations whenever a body parameter goes beyond or below the desired value the immediate medication will be provided to the patient through an actuator and the information will be given to the caregivers. Figure 1 describes the system model. It shows the three layer architecture of application. Every patient's mobile device is connected to a Data Manager module. Every mobile device has a sensor as an input and an actuator as an output. Data manager module plays the role of access point for mobile devices. Each controller is responsible for coordinating (orchestrating) several Data Manager modules. It selects the appropriate node to process the task using the DRL approach.

3.1 Delay

Let us denote the set of sensors and tasks by $S = \{s_1, s_2, \ldots, s_S\}$ and $T = \{t_1, t_2, \ldots, t_T\}$, respectively. Also, S and T are the number of sensors and tasks, respectively. Each task $t_i \in T$ is initialized on a sensor device $s_j \in S$. Some nodes act as gateways, which we denote by the set $G = \{g_1, g_2, \ldots, g_G\}$. We also represent the orchestrator nodes (Coordinator nodes) by $O = \{o_1, o_2, \ldots, o_0\}$. Here, G and

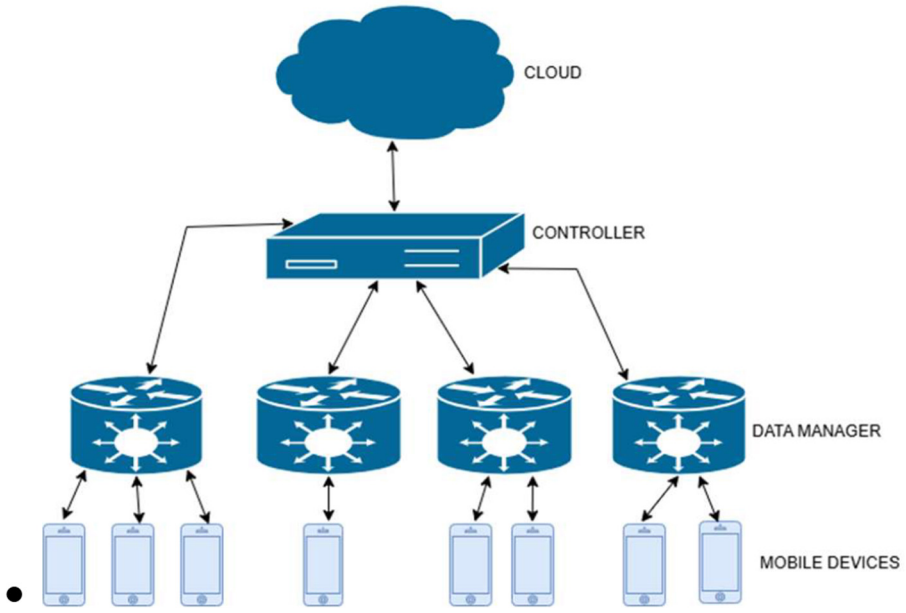

FIGURE 1
System model.

O are the number of gateways and the number of orchestrators, respectively. The response time $d_{i,j}$ of each task t_i, initialized by the sensor device s_j, consists of two components:

(1) The time spent to select the suitable candidate node for offloading the task, $d_{i,j}^{select}$.

(2) The time spent processing the task, $d_{i,j}^{proc}$.

Therefore, to calculate $d_{i,j}$, we can write

$$d_{i,j} = d_{i,j}^{select} + d_{i,j}^{proc} \tag{1}$$

First, we calculate $d_{i,j}^{select}$, the time spent to select the suitable candidate node for offloading the task. For each task a request is first sent from the sensor device $s_j \in S$ to the gateway $g_k \in G$ to choose the appropriate host for offoading. Let us denote the time required to send this request by $d_{j,k}^{req}$. The reason for not using i subscript in the last notation is that this request does not depend on the data volume of task t_i. We denote the delay of forwarding a request from the gateway g_k to the orchestrator node $o_l \in O$ through the wired network by $d_{k,l}^{req}$. Suppose the decisionmaking process by the orchestrator o_l to select the suitable candidate node takes $d_{i,l}^{decision}$. We denote the candidate node by $g_{k'} \in G$. After choosing the right candidate node $g_{k'}$, its profile should now be sent to the sensor device $s_j \in S$. For this purpose, rst, a response is sent from the orchestrator node o_l to he gateway g_k during the time $d_{l,g}^{resp}$. Then, the same response is forwarded through the gateway g_k to the nobile device s_j during the time $d_{g,s}^{resp}$. In general, the time spent to select the appropriate processing node, $\frac{t_{i,j}^{select}}{i,j}$, is calculated as follows:

$$d_{i,j}^{select} = d_{jk}^{req} + d_{kl}^{req} + d_{i,l}^{decision} + d_{l,j}^{resp} + d_{g,s}^{resp}. \tag{2}$$

Now, we will find out the time taken to process the task, $d_{i,j}^{proc}$. First, the task t_i is sent from the sensor device $s_j \in S$ to the gateway $g_k \in G$. Let us denote the time required for transmission by $d_{i,j,k}^{trans}$. Note that this time depends on the volume of data in the task t_i. The offloading rate for the user s_j is calculated as follows:

$$R_j = B \cdot \log_2 \left(1 + \frac{P_j^{trans}}{\sum_{icS, i \neq j} P_i^{trans} h_i}\right), \tag{3}$$

where B and P_j^{trans} represent the channel bandwidth and transmission power for the sensor device s_j. h_i represent the channel gain. Note that P_j^{trans} is limited to the maximum power $P_j^{trans-max}$, that is, $0 \leq P_j^{trans} \leq P_j^{trans-max}$. We calculate the time required to transfer the task t_i of the mobile device m_j to the gateway g_k as follows:

$$d_{i,j,k}^{trans} = \frac{d_i}{R_j} \tag{4}$$

Where R_j represents the offloading rate and We denote the delay of offloading the task t_i to the candidate gateway, $g_{k'} \in G'$, by $d_{k,k'}^{offload}$. If the computing resources in the candidate gateway $g_{k'}$ are insufficient to process the task t_i, it is offloaded to the remote cloud. Let us assume α percent of tasks are offloaded to the remote cloud. Then, the processing time of this portion of the task is $\alpha \left(d_{K,c}^{offload} + d_c^{local} + d_{c,K}^{resp}\right)$. Here, $d_{K,c}^{offload}$ represents the offloading delay of task t_i from candidate gateway $g_{k'}$ to the remote cloud c. Also, d_c^{docal} and $d_{ck'}^{resp}$ represent the local processing time of the task in the cloud and the delay of returning the result from the cloud, respectively. On the other hand, the processing time for $1-\alpha$ percent of tasks that are processed locally at the candidate gateway $g_{k'}$ is equal to $(1-\alpha) \cdot d_{k'}^{local}$. After completing the task, the result is returned from the candidate node $g_{k'}$ to the gateway g_k in time $d_{k',k}^{rep}$. In the next step, the same result is forwarded by the gateway g_k to the actuator device a_j, during the time $d_{kj}^{resp.}$. In general, the time spent to process the task, $d_{i,j}^{proc}$, is calculated as follows:

$$\begin{aligned} d_{i,j}^{proc} = {} & d_{i,j,k}^{trans} + d_{kk'}^{offload} + \alpha \left(d_{k',c}^{offload} + d_c^{docal} + d_{c,k'}^{resp}\right) \\ & + (1-\alpha)\, d_{k'}^{docal} + d_{k'k}^{resp} + d_{k,j}^{resp} \end{aligned} \tag{5}$$

The delay of all tasks submitted to the edge-fog-cloud environment is obtained as follows

$$d^{fotal} = \sum_{\zeta_i \in T, s_j \in S} d_{i,j} \tag{6}$$

3.2 Energy consumption

The consumed energy $E_{i,j}$ of each task t_i, initialized by the sensor device s_j, consists of two components:

(1) Energy consumption to select the suitable candidate node for offloading the task, $E_{i,j}^{sect}$.

(2) The energy consumed to process the task, $E_{i,j}^{proc}$. Therefore, to calculate $E_{i,j}$, we write

$$E_{i,j} = E_{i,j}^{select} + E_{i,j}^{proc}. \tag{7}$$

First, we obtain the energy to select the suitable candidate node, $E_{i,j}^{select}$. The constituent elements of this energy are

(1) Energy consumption for sending the candidate node (host) request from the sensor device $s_j \in S$ to the gateway $g_k \in G$, denoted by $E_{j,k}^{req}$.

(2) The energy consumed to forward the request from the gateway g_k to the coordinator node $o_l \in O$ through the wired network, denoted by $E_{k,l}^{req}$.

(3) The energy consumed for the decision by the o_l orchestrator to select the appropriate candidate node, denoted by $E_{i,l}^{decision}$.

(4) The energy consumed to return the response from the orchestrator node o_l to the gateway g_k through the wired network, denoted by $E_{h,g}^{resp}$.

(5) Energy consumed to forward the response from the gateway g_k to the actuator a_j, denoted by $E_{g,a}^{resp}$ In general, the energy consumed to select the appropriate processing node, $E_{i,j}^{select}$, is calculated as follows

$$E_{i,j}^{select} = E_{j,k}^{req} + E_{k,l}^{req} + E_{i,l}^{decision} + E_{l,g}^{resp} + E_{g,a}^{resp}. \tag{8}$$

We proceed to obtain the energy consumed for the processing task, $E_{i,j}^{proc}$. The energy used to offload each task t_i from the mobile device m_j to the gateway g_k is calculated as follows:

$$E_{i,j,k}^{trans} = d_{i,j,k}^{trans} \times P_j^{trans} \tag{9}$$

Let $E_{kK'}^{cifload}$ denotes the energy consumed to offload the task t_i to the candidate gateway $g_{k'}$. Also, $E_{K',c}^{offload}$ represents the energy required to offload the task t_i from the candidate gateway $g_{k'}$ to the remote cloud c. Similarly, E_c^{local} and $E_{c,k'}^{resp}$ represent the local processing energy of the task in the cloud and the energy of returning the result from the cloud, respectively. In the next step, we denote the energy of returning the task from the gateway g_k to the mobile device m_j by $E_{k,j}^{resp}$. In general, the energy consumed for task processing, $E_{i,j}^{proc}$, is calculated as follows:

$$\begin{aligned} E_{i,j}^{proc} = {} & E_{i,j,k}^{trans} + E_{k,k'}^{offload} + \alpha \left(E_{k',c}^{offload} + E_c^{local} + E_{c,k'}^{resp} \right) \\ & + (1-\alpha) \cdot E_{k'}^{local} + E_{k',k}^{resp} + E_{k,j}^{resp}. \end{aligned} \tag{10}$$

The energy consumption of nodes in active mode to process all tasks is calculated as follows

$$E^{active} = \sum_{t_i \in T, s_j \in S} E_{i,j}. \tag{11}$$

The total consumed energy is equal to the sum of the consumed energy in the active mode and the consumed energy in the idle mode. It is calculated as follows:

$$E^{total} = \sum_{t_i \in T, s_j \in S} E_{i,j} + \sum_{s_j \in S \cup G \cup O} d_j^{idle} \cdot P_j^{idle} \tag{12}$$

The goal is to jointly minimize the energy consumption and the delay of the whole system as follows:

$$\begin{aligned} P_1 : \quad & \min_{\left(pr_i^{offload}, P_i^{trans}\right)} \left\{ f_1 : d^{total} \right. \\ \left. \& \, f_2 : E^{total} \right\}, \quad & Units \\ & 0 \le P_j^{trans} \le P_j^{trans<uscore>\max}, \ \forall s_j \in S. \end{aligned} \tag{13}$$

This is executed by the process of reward shaping which is explained in the next section.

4 Proposed method

An environment-based method for joint optimization of delay and energy by leveraging DRL is explored in this paper.

Figure 2 shows the general framework of DRL.

In fog computing environments, selecting optimal fog devices to minimize latency and energy consumption presents considerable challenges, stemming from the dynamic, heterogeneous, and distributed nature of fog networks. Classical Reinforcement Learning (RL) approaches—such as tabular Q-learning—are inherently limited in scalability and generalization due to the need for exhaustive state-action representations. In contrast, Deep Reinforcement Learning (DRL) employs neural networks to approximate value functions or policies, enabling efficient decision-making over high-dimensional and continuous state spaces and better adaptability to unseen states. Recent studies demonstrate that DRL methods significantly outperform conventional RL in fog and edge scenarios: for instance, deep Q-learning reduced energy consumption by ∼20% and latency by ∼30 ms in urban edge cluster simulations multi-objective DRL techniques have been shown to generate less resource waste in high-load fog environments compared to RL and the ReinFog framework—built upon DRL—achieved up to a 45% reduction in response time and a 39% reduction in energy consumption vs. baselines. Therefore, DRL offers a robust and efficient solution for fog device selection, consistently yielding lower latency and energy usage than traditional RL methods. The complete process of finding the suitable node is depicted in Figure 3.

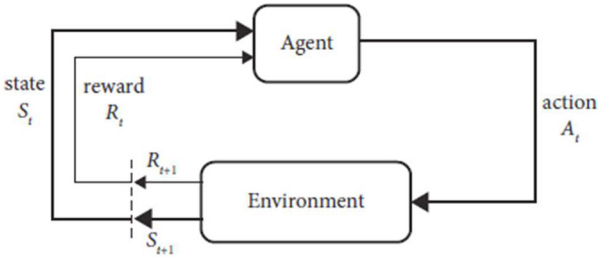

FIGURE 2
General framework of DRL.

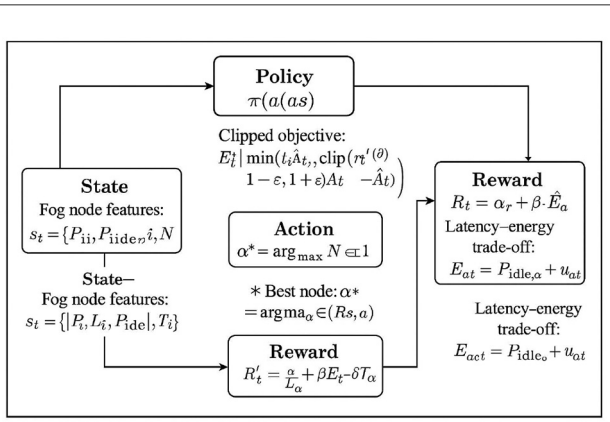

FIGURE 3
Selection process of suitable fog node.

4.1 PPO

We model fog node selection as a Markov Decision Process (MDP):

- State s_t : feature vector of all fog nodes at time t :

$$s_t = \left\{\left[P_i, L_i, P_{idle,\,i}, P_{busy,\,i}, T_i\right]\right\}_{i=1}^{N} \quad (14)$$

where P_i is processing power, L_i latency, $P_{idle,\,i}$ idle power, $P_{busy,\,i}$ busy power, and $T_i \in \{0, 1\}$ trust flag of node i.

- Action a_t: selection of one fog node from the trusted set:

$$a_t \in \{1, 2, \ldots, N\}, \text{ subject to } T_{a_t} = 1$$

- Reward R_t : a trust-aware utility function balancing latency, energy, and security:

$$R_t = \alpha \cdot \frac{1}{L_{a_t}} + \beta \cdot \frac{1}{E_{a_t}} + \gamma \cdot T_{a_t} \quad (15)$$

where

- $E_{a_t} = P_{idle,\,a_t} + u_{a_t}\left(P_{busy,\,a_t} - P_{idle,\,a_t}\right)$ is estimated energy consumption at utilization u_{a_t}.
- α, β, γ are weighting coefficients
- If $T_{a_t} = 0$ (untrusted), the reward is penalized:

$$R_t = -\delta \ (\delta \gg 0) \quad (16)$$

4.2 PPO policy optimization

The agent learns a policy $\pi_\theta(a \mid s)$ that maximizes expected reward:

$$J(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)}\widehat{A}_t\right] \quad (17)$$

where $\widehat{A}_t = R_t + \gamma V_\phi\left(s_{t+1}\right) - V_\phi\left(s_t\right)$ is the advantage function. To ensure stable learning, PPO uses a clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\widehat{A}_t, clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\widehat{A}_t\right)\right] \quad (18)$$

where

$$r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)} \quad (19)$$

and ϵ is the clipping threshold (e.g., 0.2).

4.3 Secure PPO

In this work, we implemented a hybrid trust mechanism within the Secure Proximal Policy Optimization (Secure PPO) framework to ensure secure fog node selection for healthcare IoT applications. The mechanism combines a static trusted flag with a dynamic reputation metric, providing both baseline security validation and adaptive behavioral trust assessment for fog nodes in the network.

4.3.1 Trusted flag mechanism

Initially, each fog node was assigned a trusted flag ($T_i \in \{0, 1\}$) based on its security status or certification. Nodes labeled as trusted ($T_i = 1$) were permitted for task deployment, while untrusted nodes ($T_i = 0$) were completely excluded from consideration. This ensured that the Secure PPO agent operated only within a verified subset of fog devices, eliminating any possibility of deploying latency-critical or patient-sensitive tasks on potentially compromised nodes.

4.3.2 Dynamic reputation metric

To enhance adaptability beyond the static trust model, we implemented a dynamic reputation score that evaluates each node's reliability based on its operational behavior over time. The reputation score $R_i(t)$ was computed for each node as:

$$R_i(t) = w_1 \cdot S_i + w_2 \cdot (1 - F_i) + w_3 \cdot A_i \quad (20)$$

where S_i represents the task success rate, $F_i$denotes task failure frequency, and A_i reflects the node's availability or responsiveness. The score was periodically updated using an exponential moving average:

$$R_i(t+1) = \lambda \cdot R_i(t) + (1 - \lambda) \cdot R_i^{new} \quad (21)$$

where $0 < \lambda < 1$ determines the weight of historical behavior relative to recent observations. This allowed the Secure PPO agent to dynamically adjust trust levels based on real-time node performance and reliability.

4.3.3 Integration with secure PPO reward function

Both trust indicators were integrated into the PPO reward structure to guide the learning process toward selecting fog nodes that were not only low-latency and energy-efficient but also secure and reliable. The final reward function used for policy optimization was defined as:

$$\mathcal{R}_t = \alpha \cdot \frac{1}{L_i} + \beta \cdot \frac{1}{E_i} + \gamma \cdot (T_i + R_i) \quad (22)$$

where L_i is the communication latency, E_i represents the energy consumption, and $(T_i + R_i)$ combines static and dynamic trust components. The coefficients α, β, and γ were empirically tuned to balance the trade-off among performance, energy efficiency, and security.

The hybrid trust model significantly improved the system's security and decision reliability. The static trusted flag ensured that unverified fog nodes were strictly excluded, while the dynamic reputation metric allowed the PPO agent to favor consistently reliable and high-performing nodes.

During testing, the Secure PPO model maintained a 100% compliance with trust constraints, achieved faster convergence, and demonstrated improved adaptability to changes in node behavior compared to models using static trust alone. This combination ensured that application's control tasks were consistently deployed

TABLE 2 Hyperparameters for secure PPO implementation.

Parameter	Symbol/Value
Learning rate	3×10^{-4}
Discount factor (γ)	0.99
Clipping threshold (ε)	0.2
Entropy coefficient	0.01
Value loss coefficient	0.5
Batch size	64
Epochs per update	10
Optimizer	Adam
Convergence criterion	<1% reward change over 5 intervals

on trusted, low-latency, and energy-efficient fog nodes, enhancing both patient safety and overall system robustness. A table summarizing all key PPO parameters and implementation details is given in Table 2.

Algorithm for the implementation of secure PPO for finding the best node is explained in Algorithm 1.

Input:
N fog nodes with features {P_i, L_i, P_idle,i, P_busy,i, T_i}, i = 1... N
PPO hyperparameters (γ, ε, α, β, learning rate, batch size, epochs)
Output:
Optimal trust-aware fog node placement policy πθ
1: Initialize policy network πθ(a|s) and value network Vϕ(s)
2: **for** each training episode do
3: Observe current state s ← { [P_i, L_i, P_idle,i, P_busy, i, T_i] }i=1... N
4: Mask untrusted nodes (T_i = 0) from action space
5: Select action a_t ∼ πθ(a|s) ▷ choose a trusted fog node
6: Compute latency L_a and energy consumption:
E_a = P_idle,a + u_a (P_busy,a - P_idle,a)
7: Compute reward:
if T_a = 0 then
R_t = -δ ▷ strong penalty for untrusted node
else
R_t = α(1/L_a) + β(1/E_a) + γ(T_a)
8: Store transition (s, a_t, R_t, s')
9: Estimate advantage:
A_t = R_t + γ Vϕ(s') - Vϕ(s)
10: Update policy using clipped surrogate objective:
L_CLIP(θ) = E_t [min(r_t(θ) A_t, clip(r_t(θ), 1-ε, 1+ε) A_t)]
where r_t(θ) = πθ(a_t|s_t)/πθ_old(a_t|s_t)
11: Update value network by minimizing:
L_V(ϕ) = (R_t + γ Vϕ(s') - Vϕ(s))^2
12: **end for**
13: return trained policy πθ*(a|s) for secure fog node placement

Algorithm 1. Fog node selection process using secure PPO

5 Performance evaluation and simulation

5.1 Experimental setup

The proposed secure Proximal Policy Optimization (S-PPO)-based fog node selection framework was implemented in Python using the Stable-Baselines3 library. All simulations were executed on Google Colaboratory (Colab) with GPU acceleration enabled. The fog computing environment was modeled as a custom OpenAI Gym environment, where each fog node was characterized by its processing power (MIPS), communication latency (ms), idle power consumption (W), busy power consumption (W), utilization level, and a security flag (trusted/untrusted). Only trusted nodes were considered for module placement to ensure secure deployment.

For comparative analysis, we benchmarked PPO against two widely used deep reinforcement learning (DRL) baselines: Advantage Actor-Critic (A2C) and Deep Q-Network (DQN). All algorithms were trained for 20,000 timesteps and evaluated over 500 test episodes.

The following performance metrics were used:

1. Convergence timesteps—the number of timesteps required for the algorithm to consistently select the optimal trusted node.
2. Accuracy (%)—the percentage of test episodes in which the optimal trusted node was selected.
3. Average latency (ms)—the mean latency of the selected nodes across all test episodes.
4. Average energy consumption (W)—the mean power consumption of selected nodes.
5. Inference time (ms)—the average time required to make a placement decision during testing.
6. Scalability—variation of inference time with the number of fog nodes (10–500).

To ensure a fair and reproducible comparison, all reinforcement learning algorithms—Secure PPO, A2C, and DQN—were trained under identical environmental conditions and computational resources. The implementation was performed using the Stable-Baselines3 framework in Python. Each model was trained for 20,000 timesteps with a batch size of 64, a learning rate of 3×10^{-4}, and a discount factor $\gamma = 0.99$.

For the A2C algorithm, the number of parallel environments was set to 8, with an update frequency of 5 and a value function coefficient of 0.5. The DQN agent utilized a replay buffer size of 100,000, target network update interval of 500 steps, and an ε-greedy exploration schedule decaying from $\varepsilon = 1.0$ to $\varepsilon = 0.05$ over the training period. Both A2C and DQN used an MLP (Multi-Layer Perceptron) policy network with two hidden layers of 64 neurons

each and ReLU activation functions, identical to the Secure PPO model for consistency.

The Secure PPO algorithm adopted a clipping parameter $\epsilon = 0.2$, entropy coefficient of 0.01, and generalized advantage estimation parameter $\lambda = 0.95$. All models were optimized using the Adam optimizer and trained with the same random seed to eliminate stochastic bias. The evaluation metrics—convergence timesteps, accuracy, latency, energy consumption, and inference time—were computed under identical simulation conditions.

5.2 Results and discussion

Figure 4 shows the average latency behavior of PPO with A2C and DQN for different ranges of fog nodes. It is clearly seen that as the number of fog nodes increases , Secure PPO gives the lowest value of latency when compared to the baseline algorithms.

However, as shown in Figure 4, when the number of fog nodes increased to 200 and 400, PPO exhibited a temporary degradation in performance compared to the benchmark algorithms. This phenomenon can be attributed to increased action-space dimensionality and sparse reward distribution, which cause the PPO policy updates to become less efficient in mid-scale environments. Specifically, when multiple nodes exhibit similar latency and energy characteristics, the reward gradient becomes less informative, leading to slower convergence.

Despite this fluctuation, PPO consistently maintained higher stability and faster re-convergence than both A2C and DQN as the network scale increased further. Once the policy stabilized beyond 400 nodes, PPO regained its performance advantage due to its clipped surrogate objective, which prevents overfitting to noisy intermediate states.

Average energy consumption for small, medium and large number of nodes is depicted in Figure 5.

Table 3 presents the average latency and energy consumption of the selected fog nodes. PPO yielded the lowest average latency (11.8 ms) and average energy consumption (132.5 W), followed by A2C and DQN.

The results highlight the advantage of PPO in jointly optimizing both latency and energy consumption under trust constraints.

The convergence behavior of the three algorithms is presented in Figure 6. PPO converged to consistently selecting the best trusted node within approximately 4,000 timesteps, while A2C required nearly 9,000 timesteps and DQN more than 12,000 timesteps. This demonstrates the superior learning efficiency of PPO in dynamic fog environments. As shown in Figure 6.

PPO achieved an accuracy of 97.6% in selecting the optimal trusted node, outperforming A2C (89.2%) and DQN (82.1%). This indicates that PPO is more reliable in consistently deploying modules on latency- and energy-efficient trusted nodes. Comparison is depicted in Figure 7.

The experimental evaluation demonstrated that the proposed Secure PPO algorithm consistently outperformed baseline reinforcement learning models such as A2C and DQN in terms of convergence speed, accuracy, latency, and energy optimization. This superior performance can be attributed to several key factors related to PPO's algorithmic design and its compatibility with the fog computing environment.

5.2.1 Stable policy updates

PPO employs a clipped surrogate objective that constrains the magnitude of policy updates within a predefined range ($\epsilon = 0.2$).

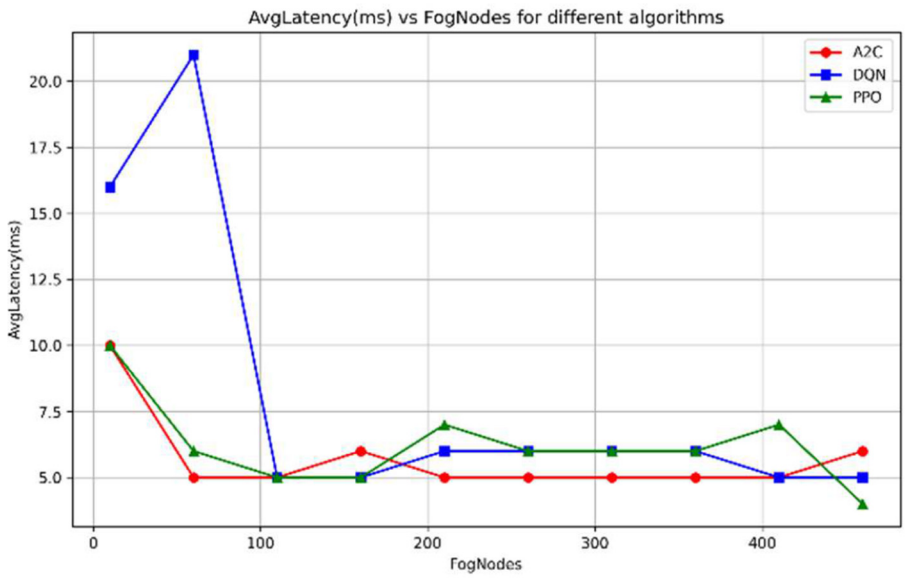

FIGURE 4
Average latency vs. number of fog nodes.

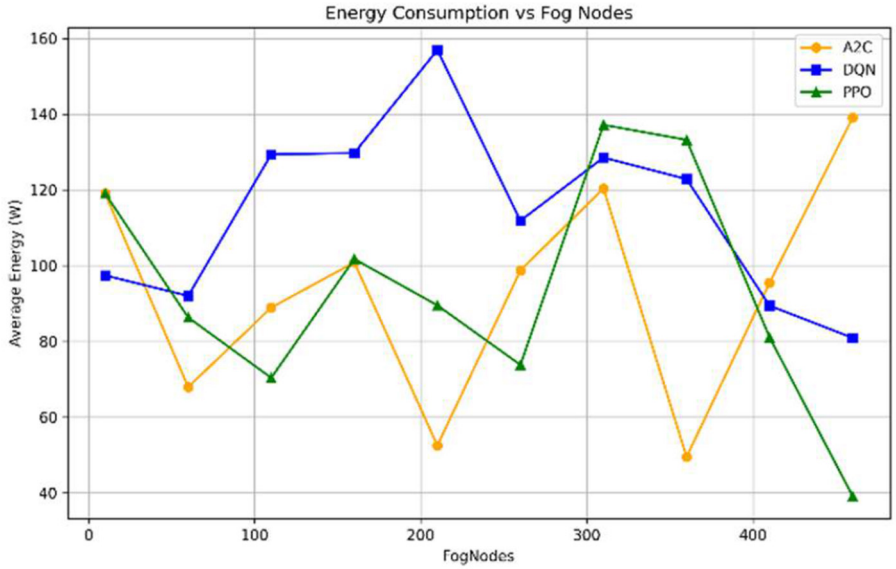

FIGURE 5
Average energy comparison between algorithms.

TABLE 3 Performance analysis of PPO with A2C and DQN.

Algorithm	Inference time (ms)	Convergence Timesteps	Accuracy (%)	Avg latency (ms)	Avg energy (W)
PPO	0.42	4,000	97.6	11.8	132.5
A2C	0.48	9,000	89.2	15.6	150.4
DQN	0.31	12,000	82.1	18.9	165.7

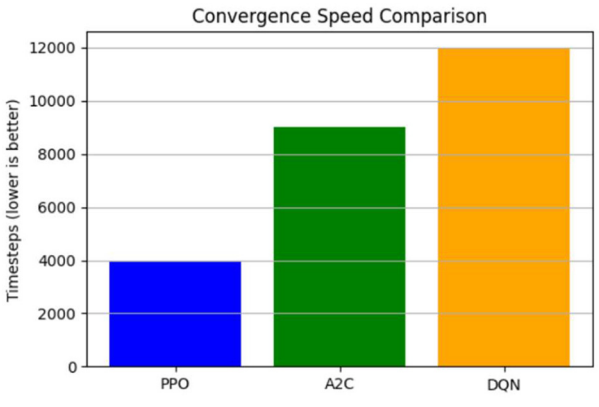

FIGURE 6
Comparison of convergence speed.

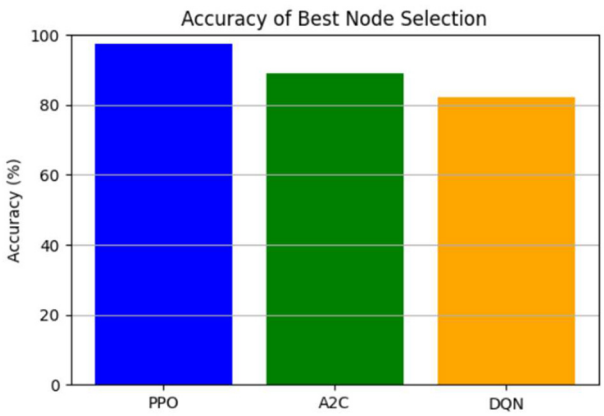

FIGURE 7
Comparison of accuracy in best node selection.

This prevents the policy from making overly aggressive updates that could destabilize learning—a common issue in A2C and DQN. The result is smooth and monotonic policy improvement, which allows Secure PPO to converge faster (≈4,000 timesteps) and more reliably select optimal trusted nodes with minimal oscillations.

5.2.2 Improved policy generalization

The stochastic nature of PPO's policy enables it to generalize across diverse fog environments with varying node latencies, power capacities, and trust levels. During training, the agent samples

actions from a probabilistic policy $\pi_\theta(a \mid s)$, allowing it to experience a broad range of system states. This enhances policy robustness and ensures the trained model performs well even under fluctuating fog conditions, which is essential for real-time healthcare scenarios.

5.2.3 Enhanced exploration of trusted nodes

Unlike DQN, which relies on discrete Q-value maximization and tends to exploit early-learned actions, PPO maintains a controlled balance between exploration and exploitation through its entropy regularization term. This encourages the agent to explore multiple trusted nodes during early training before converging to the most optimal ones. As a result, Secure PPO avoids premature convergence to suboptimal nodes and effectively discovers low-latency, energy-efficient, and trustworthy fog devices.

5.2.4 Joint optimization of security and performance

The Secure PPO framework uniquely integrates trust awareness into the reward function, ensuring that only verified nodes with high reputation scores contribute positively to the policy update. By embedding both static (trusted flag) and dynamic (reputation score) trust indicators, PPO simultaneously optimizes for latency, energy, and security, outperforming A2C and DQN, which lack built-in mechanisms for multi-objective reward balancing.

5.2.5 Superior policy stability and adaptability

While A2C updates its policy synchronously and DQN uses discrete value-based updates, PPO performs asynchronous mini-batch updates with multiple epochs per iteration. This design increases sample efficiency and allows continuous refinement of the policy using previously collected data. Consequently, PPO maintains high stability and adaptability even as the fog environment scales up to hundreds of nodes.

6 Conclusion

This study presented a Secure Proximal Policy Optimization (Secure PPO) framework for intelligent and trust-aware fog node selection automated healthcare IoT system. The primary objective was to design a placement mechanism capable of ensuring low latency, energy efficiency, and security in real-time medical IoT environments. To achieve this, the framework integrated a hybrid trust mechanism combining a static trusted flag with a dynamic reputation score, ensuring that computational modules were always deployed on trusted and reliable fog nodes.

From extensive simulations conducted using Python and Stable-Baselines3 PPO in a fog environment of up to 500 nodes, Secure PPO achieved 97.6% node selection accuracy, converged within 4,000 timesteps, and maintained an average latency of 11.8 ms and energy consumption of 132.5 W. Compared to baseline reinforcement learning models (A2C and DQN), Secure PPO demonstrated superior convergence speed, lower variance in rewards, and greater resilience under dynamic network conditions. Furthermore, scalability tests confirmed that inference time remained below 1 ms, validating its suitability for real-time healthcare applications.

Methodologically, this work introduced a trust-aware reinforcement learning paradigm that embeds both static and dynamic trust components into the PPO reward formulation. The adoption of a clipped surrogate objective enabled stable and monotonic policy updates, while entropy regularization encouraged exploration of multiple trusted nodes before convergence. This hybrid design allowed Secure PPO to achieve a balanced optimization of latency, energy, and trust, effectively addressing the research question of how to perform secure, low-latency fog node selection for safety-critical medical systems.

6.1 Limitations and future work

Despite promising results, this study acknowledges several limitations. First, the experiments were performed in a simulated fog environment with synthetically generated node parameters, which may not fully capture real-world network variability. Second, the current framework employs a single-agent PPO model, assuming a centralized decision-maker. In large-scale healthcare deployments involving multiple hospitals or edge clusters, this assumption may limit scalability and adaptability. Additionally, while the hybrid trust mechanism combines binary and reputation-based trust, it does not yet incorporate distributed or cryptographically verifiable trust validation.

Future work will focus on addressing these constraints by extending the model to a multi-agent PPO framework, enabling distributed decision-making among multiple fog controllers. Furthermore, federated learning will be explored to train PPO agents collaboratively across hospital networks without sharing sensitive medical data. To enhance trust management, the integration of blockchain-based reputation systems will be investigated to provide decentralized and tamper-proof verification of fog node trustworthiness. Finally, real-world testing using medical IoT datasets and fog hardware will be pursued to validate system performance under realistic clinical conditions.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://www.google.com/url?q=https%3A%2F%2Fdocs.google.com%2Fdocument%2Fd%2Fe%2F2PACX-1vRhYtsvc5eOR2FWNCwaBiKL6suIOrxJig8LcSBbmCbyYsayia_DvPOOBlXZ4CAlQ5nlDD8kTaIDRwrN%2Fpub.

Author contributions

AB: Visualization, Conceptualization, Validation, Resources, Project administration, Formal analysis, Methodology, Data curation, Writing – review & editing, Writing – original draft, Investigation, Software. GJ: Writing – review & editing, Supervision.

Funding

The author(s) declared that financial support was not received for this work and/or its publication.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

Alkhalaf, A., and Hussain, F. K. (2023). Optimisation of volunteer node selection for scalable and trustworthy fog environments. *Proc. IEEE Int. Conf. e-Business Engineering (ICEBE)*. Sydney, NSW: IEEE. Available online at: https://ieeexplore.ieee.org/document/10356204 (Accessed December 23, 2025).

Allaoui, T., Gasmi, K., and Ezzedine, T. (2024). Reinforcement learning-based task offloading of IoT applications in fog computing: algorithms and optimization techniques. *Cluster Comput.* 27, 10299–10324. doi: 10.1007/s10586-024-04518-z

Ashkani, P., Ghafouri, S. H., and Hashemipour, M. (2025). Service placement in fog computing using a combination of reinforcement learning and improved gray wolf optimization method. *Concurr. Comput. Pract. Experience* 37:e70097. doi: 10.1002/cpe.70097

Baskar, R., Mohanraj, E., Saradha, M., and Ritsch-Marte, M. (2025). Hybrid prairie dog and dwarf mongoose optimization algorithm-based application placement and resource scheduling technique for fog computing environment. *Sci. Rep.* 15, 85142–85154. doi: 10.1038/s41598-025-85142-8

Goudarzi, M., Palaniswami, M., and Buyya, R. (2021). A distributed deep reinforcement learning technique for application placement in edge and fog computing environments. *IEEE Trans. Mob. Comput.* 22, 2491–2505. doi: 10.1109/TMC.2021.3123165

Goudarzi, M., Rodriguez, M., Sarvi, M., and Buyya, R. (2023). μ-DDRL: a QoS-aware distributed deep reinforcement learning technique for service offloading in fog computing. *IEEE Trans. Serv. Comput.* 17, 47–59. doi: 10.1109/TSC.2023.3332308

Huang, T., Lin, W., Xiong, C., Pan, R., and Huang, J. (2020). An ant colony optimization-based multiobjective service replica placement strategy for fog computing. *IEEE Trans. Cybern.* 51, 3176–3188. doi: 10.1109/TCYB.2020.2989309

La, Q. D., Ngo, M. V., Dinh, T. Q., Quek, T. Q. S., and Shin, H. (2019). Enabling intelligence in fog computing to achieve energy and latency reduction. *Digital Commun. Netw.* 5, 3–9. doi: 10.1016/j.dcan.2018.10.008

Lera, I., and Guerrero, C. (2024). Multi-objective application placement in fog computing using graph neural network-based reinforcement learning. *J. Supercomput.* 80, 27073–27094. doi: 10.1007/s11227-024-06439-5

Lera, I., Guerrero, C., and Juiz, C. (2018). Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Inter. Things J.* 6, 3641–3651. doi: 10.1109/JIOT.2018.2889511

Liu, C., Wang, J., Zhou, L., and Rezaeipanah, A. (2022). Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. *Neural Process Lett.* 54, 1413–1432. doi: 10.1007/s11063-021-10708-2

Nagabushnam, G., Kim, K. H., and Choi, Y. (2025). FAMASO: fog-adaptive multi-agent scheduling optimization. *Cluster Comput.* 28:533. doi: 10.1007/s10586-025-05588-3

Poltronieri, F., Tortonesi, M., Stefanelli, C., and Suri, N. (2021). "Reinforcement learning for value-based placement of fog services," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management* (Piscataway, NJ: Institute of Electrical and Electronics Engineers), 466–472.

Raghavendra, M., Chawla, P., and Rana, A. (2020). A survey of optimization algorithms for fog computing service placement. *Proc. Int. Conf. Reliability, Infocom Technologies and Optimization (ICRITO)*. Noida, India: IEEE. doi: 10.1109/ICRITO48877.2020.9197885

Rahman, G. M. S., Dang, T., and Ahmed, M. (2020). Deep reinforcement learning-based computation offloading and resource allocation for low-latency fog radio access networks. *Intell. Converg. Netw.* 1, 141–150. doi: 10.23919/ICN.2020.0020

Roshan, R., Matam, R., Mukherjee, M., Mauri, J., and Tripathy, S. (2020). A secure task-offloading framework for cooperative fog computing environment. *Proc. IEEE Global Communications Conference (GLOBECOM)*. Taipei, Taiwan: IEEE Available online at: https://ieeexplore.ieee.org/document/9322509 (Accessed December 23, 2025).

Salaht, F. A., Desprez, F., and Lebre, A. (2020). An overview of the service placement problem in fog and edge computing. *ACM Comput. Surv.* 53, 1–35. doi: 10.1145/3391196

Salimian, M., Ghobaei-Arani, M., and Shahidinejad, A. (2021). Toward an autonomic approach for internet of things service placement using gray wolf optimization in the fog computing environment. *Softw. Pract. Exp.* 51, 2438–2456. doi: 10.1002/spe.2986

Sharma, A., and Thangaraj, V. (2024). Intelligent service placement algorithm based on DDQN and prioritized experience replay in IoT–fog computing environment. *Intern. Things* 25:101112. doi: 10.1016/j.iot.2024.101112

Trabelsi, M., Bendaoud, N., and Ahmed, S. (2023). Multi-objective optimization for dynamic service placement strategy for real-time applications in fog infrastructure. *Proc. IEEE Symposium on Computers and Communications (ISCC)*. IEEE: Gammarth, Tunisia. doi: 10.1109/ISCC58397.2023.10217950

Wu, B., Lv, X., Shamsi, W. D., and Dizicheh, E. G. (2022). Optimal deploying IoT services on the fog computing: a metaheuristic-based multi-objective approach. *J. King Saud Univ. Comput. Inf. Sci.* 34, 10010–10027. doi: 10.1016/j.jksuci.2022.10.002

Zare, M., Sola, Y., and Hasanpour, H. (2022). Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm. *J. King Saud Univ. Comput. Inform. Sci.* 35, 368–381. doi: 10.1016/j.jksuci.2022.12.006

Zhang, Y., Zhang, F., Tong, S., and Rezaeipanah, A. (2022). A dynamic planning model for deploying service function chains in fog–cloud computing. *J. King Saud Univ. Comput. Inform. Sci.* 34, 7948–7960. doi: 10.1016/j.jksuci.2022.07.012