Service Ready Caching and Offloading for Urgency Aware MEC IoMT

Kaushik Mishra[1], Ganesh Khekare[2]
[1]Lincoln University College (LUC), 47301, Petaling Jaya, Selangor Darul Ehsan, Malaysia
[2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, Tamil Nadu, India
E-mail ID: pdf.kaushik@lincoln.edu.my, khekare.123@gmail.com

Abstract: ECG-driven IoMT requires low-latency and service-ready edge execution for immediate clinical action. In the case of ECG-driven MEC IoMT, processing a task can occur at any edge node if the required ECG service/model/container is available. But nearest neighbor, least loaded neighbor, and popular caching schemes might not be able to process tasks in emergency conditions due to their inability to handle three factors together – service availability, resource constraints, and urgency. This research paper aims to address this issue by presenting Tiny MedDMARL, which is a lightweight decentralized learning framework for urgent service caching and offloading. The proposed framework makes use of centralized learning and decentralized actor-only execution by running only compressed actors on all MEC nodes. Two-timescale separation ensures that the task offloading occurs faster while service caching takes place slowly. Results of simulation through MIMIC IV ECG-driven tasks reveal 127.17 ms average latency, 79.60% cache hit rate, 17 ms cold-start delay, 6.1% SLA violation, 8.7% cloud offloading, and 96.8% critical task success.
Keywords: Internet of Medical Things, mobile edge computing, service caching, task offloading, ECG monitoring, decentralized learning, cold start delay.

Introduction

IoMT (Internet of Medical Things) enables real-time health monitoring in the form of interconnected medical sensors, wearable devices and edge infrastructure in hospitals. ECG (electrocardiogram) monitoring is one of the latency sensitive IoMT applications due to the necessity to make rapid inference and timely actions in the presence of abnormal heart diseases. Cloud only execution offers enough computing capacity but induces larger backhaul delay and communication dependencies. Mobile Edge Computing (MEC) alleviates such a problem by executing computation locally. However, MEC execution makes sense only when the selected MEC node has enough computing capacity and the desired ECG service installed.

It gives rise to the problem of service caching and task offloading optimization. When allocating a task to either the closest or least loaded MEC node without service check, the task would encounter cold start delay, neighbor forwarding, or even cloud fallback. Although existing MEC resource allocation and request scheduling approaches enhance computing efficiency, they fail to address service readiness for urgent healthcare workloads effectively [1], [2]. Service placement and request routing can improve urgent edge computation; yet it is hard to preserve rare clinical services under limited caching capability [3].

Deep Reinforcement Learning (DRL) based approaches offer better long-term decisions in MEC execution [4], [5]. Federated Deep Reinforcement Learning further supports distributed computing without centralized data [6]. However, full DRL or Multi-Agent Reinforcement Learning (MARL) training is computationally costly on MEC due to the need of critic network, replay buffer, optimizer, and gradient calculation. Such run-time overhead limits their applicability in healthcare edge environment. Also, popularity-driven cache eviction could lead to removal of the clinically important but less frequently accessed ECG services. Recent studies of collaborative caching and bursty traffic indicate that cache placement should stay stable and responsive under dynamic workload on MEC [7], [8].

To solve these problems, this paper designs a lightweight federated decentralized learning framework Tiny MedDMARL for urgency-aware service caching and task offloading in MEC-based IoMT. It conducts centralized training in the hospital edge controller or cloud controller and executes only compressed actor on MEC nodes. Fig. 1 illustrates the two-timescale strategy, where offloading is performed per ECG task and caching is updated periodically. Such an approach allows reducing the run-time overhead, ensuring service readiness and prioritizing critical ECG tasks before cloud fallback.

Contributions of this paper include but are not limited to the following. Firstly, an urgency-aware MEC IoMT framework is proposed for the execution of ECG tasks with limited cache and heterogeneous computing power. Secondly, a formulation of service readiness-aware optimization is proposed to minimize the total task latency, cache miss penalty, and CPU pressure. Thirdly, a lightweight Tiny MedDMARL scheme is presented using centralized training and decentralized actor-only execution. Finally, the two-timescale mechanism is proposed to distinguish fast task offloading and slow service cache placement.

System Architecture and Mathematical Formulation

The IoMT device, MEC nodes, hospital edge controller, and cloud server make up this system. The IoMT devices create ECG tasks. Every task will have its size of input, medical service requirement, CPU requirement, urgency, and deadline. While MEC nodes offer low latency task execution, they have little storage and processing power. As such, a task needs to be allocated to a node that can handle it computationally and functionally. Let $\mathcal{I}$, $\mathcal{E}$, $\mathcal{V}$, and $\mathcal{S}$ refer to the sets of IoMT tasks, MEC nodes, candidate execution nodes, and ECG services, respectively. Each task i at time slot t is represented as $\tau_i^t = \{b_i^t, c_i^t, s_i, \Delta_i^t, \omega_i^t\}$, where b_i^t is the input size, c_i^t is the CPU demand, s_i is the required service, Δ_i^t is the deadline, and ω_i^t is the urgency weight.

The task offloading variable is

$$x_{i,v}^t = \begin{cases} 1, & \text{if task } i \text{ is assigned to node } v, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The service caching variable is

$$y_{e,s}^t = \begin{cases} 1, & \text{if service } s \text{ is cached at MEC node } e, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The service readiness indicator is

$$H_{i,v}^t = \begin{cases} y_{v,s_i}^t, & v \in \mathcal{E}, \\ 1, & v \text{ is hospital edge or cloud.} \end{cases} \tag{3}$$

The delay of task i at node v is

$$D_{i,v}^{t} = d_{i,v}^{tx,t} + d_{v}^{q,t} + \frac{c_i^t}{f_{i,v}^t} + (1 - H_{i,v}^{t}) d_{s_i}^{cs,t} + d_{i,v}^{fb,t} \quad (4)$$

Here, $d_{i,v}^{tx,t}$, $d_{v}^{q,t}$, $c_i^t / f_{i,v}^t$, $d_{s_i}^{cs,t}$, and $d_{i,v}^{fb,t}$ denote transmission delay, queueing delay, computation delay, cold start delay, and fallback delay, respectively. The normalized CPU pressure is defined as:

$$\rho_v^t = \frac{\sum_{i \in \mathcal{J}} x_{i,v}^t f_{i,v}^t}{F_v} \quad (5)$$

The objective is to jointly optimize offloading, caching, and CPU allocation as

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{F}} \sum_{t=1}^{T} \sum_{i \in \mathcal{J}} \sum_{v \in \mathcal{V}} x_{i,v}^t \left[\alpha \omega_i^t D_{i,v}^t + \beta (1 - H_{i,v}^t) + \gamma \rho_v^t \right] \quad (6)$$

The first term minimizes urgency weighted latency. The second term penalizes cache misses. The third term discourages overloaded node selection.

Optimization is subject to the following constraints.

$$\sum_{v \in \mathcal{V}} x_{i,v}^t = 1, \forall i, t, \quad (6a)$$

$$x_{i,e}^t \leq y_{e,s_i}^t, \forall i, e, t, \quad (6b)$$

$$\sum_{s \in \mathcal{S}} y_{e,s}^t z_s \leq Z_e, \forall e, t, \quad (6c)$$

$$\sum_{i \in \mathcal{J}} x_{i,v}^t f_{i,v}^t \leq F_v, \forall v, t, \quad (6d)$$

$$D_i^t \leq \Delta_i^t, \forall i, t, \quad (6e)$$

$$0 \leq f_{i,v}^t \leq F_v, x_{i,v}^t, y_{e,s}^t \in \{0,1\}. \quad (6f)$$

These constraints ensure unique task assignment, service ready MEC execution, cache feasibility, CPU feasibility, deadline satisfaction, and binary decision validity.

Proposed Methodology

The small MedDMARL framework adopts centralized training and decentralized actor-only deployment. In the training phase, the hospital edge or cloud controller considers the following information: queue state, cache state, CPU state, service demand, task size, and task urgency. In the deployment phase, the trained policy is compressed and runs at the MEC node. In the runtime, each MEC node runs the compressed actor only. The critic, replay buffer, optimizer, and gradient are not run on MEC nodes. The proposed framework adopts the two-timescale approach. The fast timescale addresses every ECG task arrival and makes offloading choices from the local MEC, neighbor MEC, hospital edge, and cloud. The slow timescale updates the service caching every T_c slots by considering the demand, service size, and clinical criticality. This separation simplifies the action space and avoids unstable task triggered cache replacement.

In the fast-offloading stage, the service availability, queue length, CPU state, task size, and task urgency will be considered. When the local MEC node has the required service and enough CPU availability, the task will be executed locally. Otherwise, the policy will consider the service ready neighbor MEC nodes. Finally, if no edge node can handle the task, the task will be offloaded to the hospital edge or

cloud. In case of critical ECG tasks, an emergency rule will first look for a deadline-feasible service ready edge path or cloud. The slow cache update uses a service priority score,

$$\Pi_{e,s}^{t} = \lambda_1 P_{e,s}^{t} + \lambda_2 C_s^{clin} - \lambda_3 z_s \quad (7)$$

where $P_{e,s}^{t}$ is service demand, C_s^{clin} is clinical criticality, z_s is service size, and $\lambda_1, \lambda_2, \lambda_3$ are weighting factors. Services with higher priority are cached subject to storage capacity. This allows rare but critical ECG services to remain available even when their request frequency is lower than common services. Figure 1 shows the overall workflow of the proposed method.

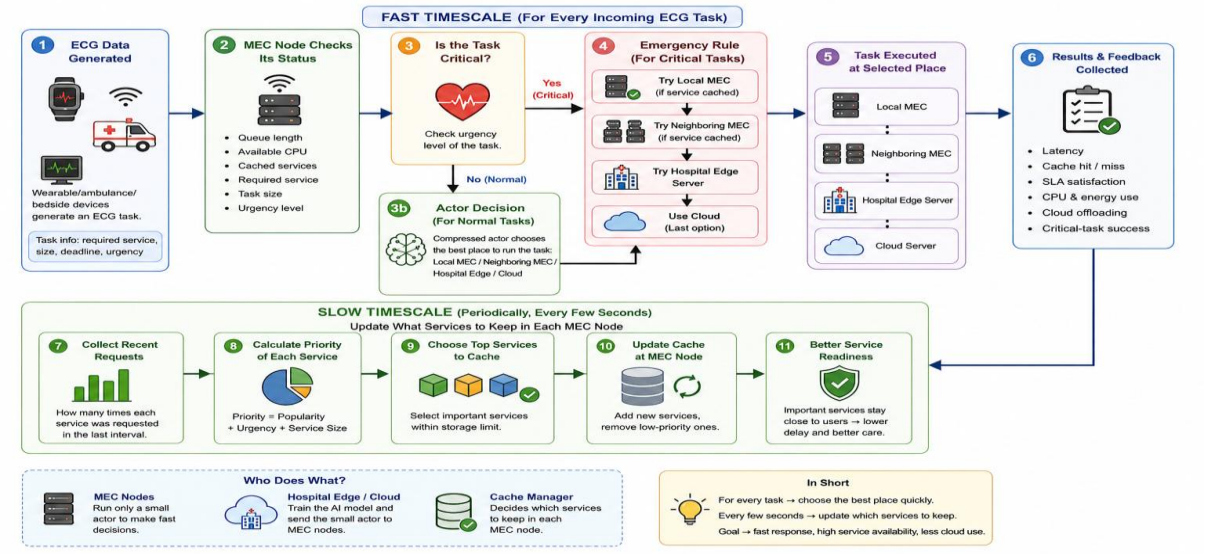


Fig. 1. Overall workflow of Tiny MedDMARL.

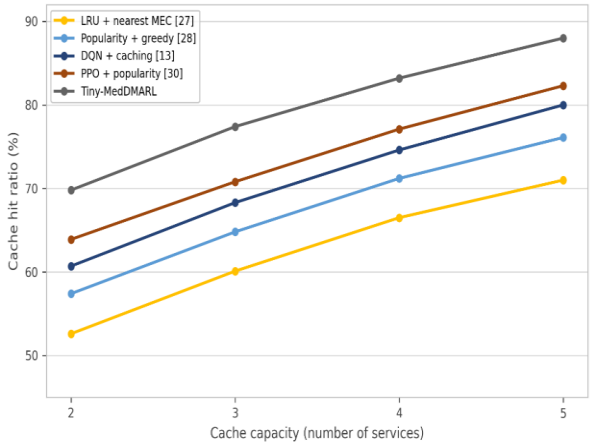
Experimental Setup

The MIMIC IV ECG dataset is employed to build the workload for the ECG driven IoMT workload. An ECG entry corresponds to one IoMT task. Herein, 12 lead ECG signals form the input data, the diagnostic category defines the needed service type, and abnormal ECG forms the urgent or critical IoMT tasks. The dataset is used for generating the workload and distributing urgency levels; however, it is not utilized to train the clinical classifier. The simulation is built in Python 3.10 and PyTorch for the MEC based IoMT workload environment with discrete time. The number of IoMT devices lies between 100 and 500 while the number of MEC nodes is within 5 and 20. MEC CPU capacity changes within 2 GHz to 10 GHz. The number of cached services per MEC node is within 2 and 4. The task arrivals occur according to Poisson process with emergency bursts, while the services' popularity follows Zipf distribution. The results are reported in terms of mean plus standard deviation. The proposed approach is compared to cloud only, nearest MEC, least loaded MEC, LRU plus nearest MEC, popularity plus greedy, DQN offloading, DQN plus caching and PPO plus popularity approaches.

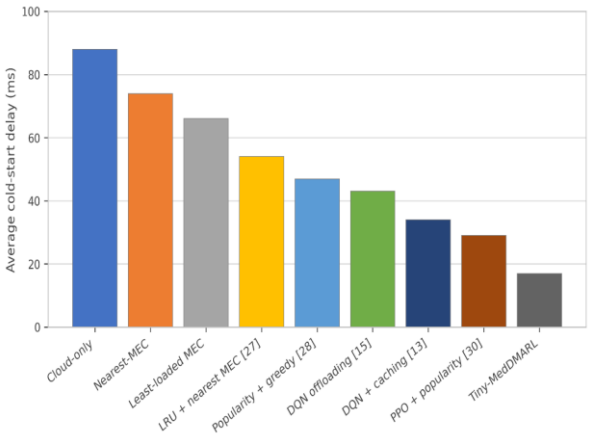
Results and Discussion

The cache hit ratio of Tiny MedDMARL at various cache capacities is presented in Fig. 2a. Tiny MedDMARL has achieved an average cache hit ratio of 79.60%. It enhances cache availability by 8.26% compared to PPO plus popularity and by 12.27% compared to DQN plus caching. The reason behind it is the

effectiveness of urgency aware caching that helps in keeping clinically critical ECG services away from displacement only based on demand frequency. Fig. 2b shows the cold start delay. Tiny MedDMARL has reduced the cold start delay to 17 ms, which is 41.38% less than PPO plus popularity and 50.00% less than DQN plus caching. This result verifies that edge service readiness minimizes the startup overhead, forwarding delay, and cloud fallback. Fig. 2c shows the impact of cache update interval T_c. For T_c=5, the latency is 143 ms and cache hit ratio is 76.1%. For T_c =20, the latency gets decreased to 131 ms and cache hit ratio gets increased to 83.2%. For T_c =50, the latency gets increased to 145 ms and cache hit ratio gets reduced to 75.3%. This proves that too frequent updates lead to instability and too late updates lead to stale placement of cache. Moderate intervals work the best. In Fig. 2d, average latency of Tiny MedDMARL is shown for increasing task arrival rate. Tiny MedDMARL maintains the lowest latency while the number of tasks ranges from 20 to 120 per slot. Tiny MedDMARL attains the average latency of 127.17 ms, where there is a 10.86% decrease from PPO plus popularity and 15.32% from DQN plus caching. Such gains are due to avoiding service unavailable MEC nodes. Fig. 2e shows the SLA violation rate. Tiny MedDMARL has obtained the lowest SLA violation rate of 6.1%. The deadline misses have been reduced by 31.46% in Tiny MedDMARL compared to PPO plus popularity and by 40.20% compared to DQN plus caching. This is important for the ECG monitoring because any delay in emergency response would be less clinically useful. The energy consumption and cloud offloading of Tiny MedDMARL are depicted in Fig. 2f. The energy consumption of Tiny MedDMARL is reduced to 5.6 J per task, leading to a 9.68% reduction in comparison to PPO plus popularity and a 52.94% reduction in comparison to cloud only execution. Cloud offloading has been minimized in Tiny MedDMARL by 39.58% in comparison to PPO plus popularity and by 49.12% in comparison to DQN plus caching. Such results verify that edge service readiness improves remote transmission and execution. Fig. 2g shows the case study of the emergency IoMT. The critical task success in Tiny MedDMARL is 96.8%, which is 3.64% higher than PPO plus popularity and 5.45% higher than DQN plus caching. While the gain over learning baselines is not so high, it is meaningful, since each additional successful completion of urgent ECG task leads to better emergency service provision.



(a)



(b)

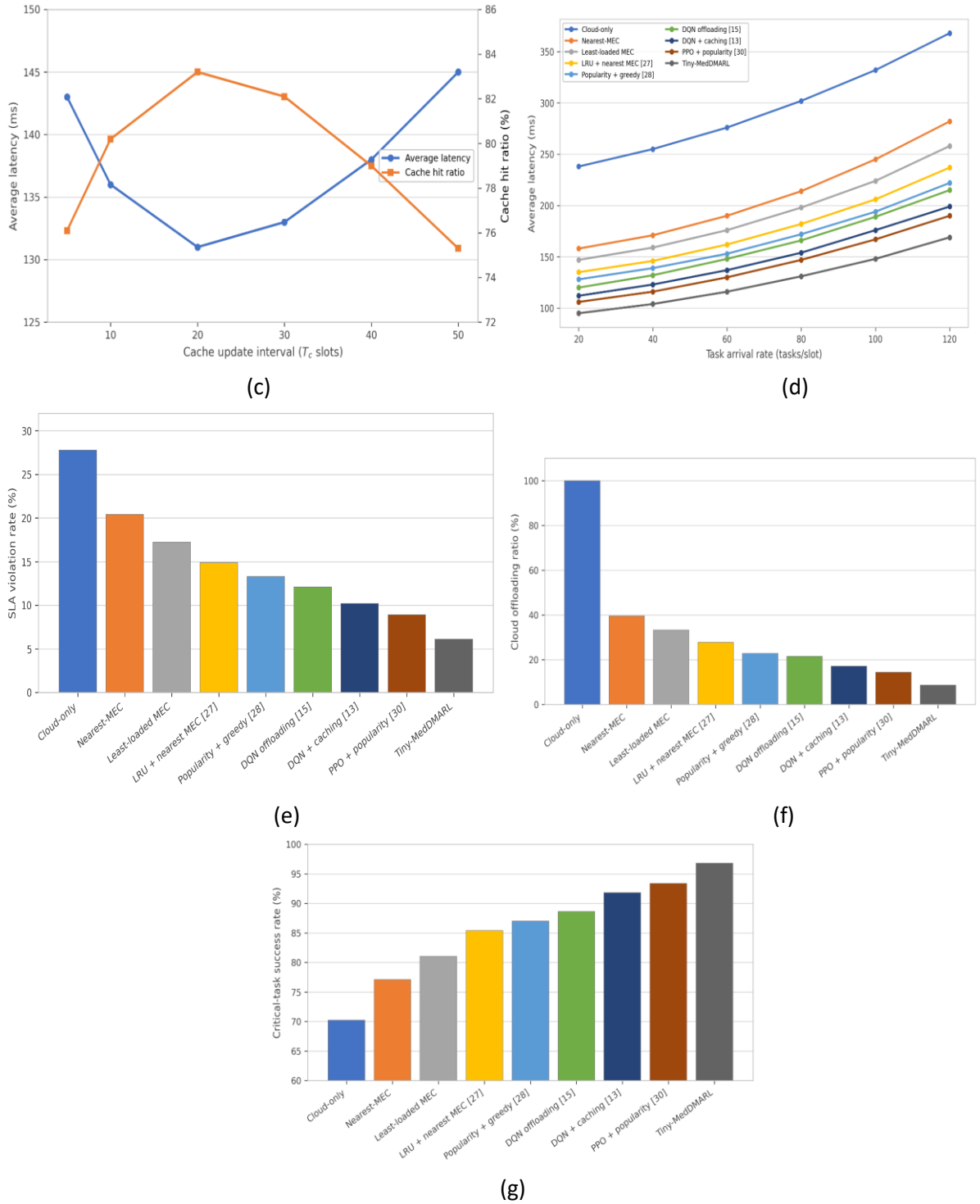


Fig. 2. (a) Cache hit ratio under different MEC cache capacities, (b) Cold start delay comparison under different service caching strategies, (c) Impact of cache update interval on latency and cache hit ratio, (d) Average latency under increasing task arrival rate, (e) SLA violation rate across compared methods, (f) Resource efficiency and cloud dependency, and (g) Emergency IoMT case study using critical task success rate.

Conclusion and Future Work

- In this paper, we proposed Tiny MedDMARL, which is a light-weighted decentralized learning paradigm for urgency-aware service caching and task offloading in MEC empowered IoMT.
- This paradigm mitigates the challenge faced by edge healthcare systems, wherein the task execution is possible in an MEC node only in case of availability of necessary ECG service. Using service readiness awareness, urgency awareness in cache, centralized training, actor-based decentralization, and twotimescale operations, Tiny MedDMARL enhances the performance of emergencies in IoMT under limited edge resources.
- Simulation results reveal that Tiny MedDMARL attains an average latency of 127.17 ms, 79.60 percent hit ratio, 17 ms cold start delay, 6.1 percent SLA violation ratio, 8.7 percent offload ratio to the cloud, 5.6 joules per task energy consumption, and 96.8 percent successful critical task execution.
- Future work will encompass validation of Tiny MedDMARL using actual MEC IoMT testbeds, profiling of cold starts using containers, inter-MEC secure collaboration, urgency validation using medical criteria, and collaborative learning among multiple hospitals.

References

[1] N. Li, L. Zhai, Z. Ma, X. Zhu, and Y. Li, "Lyapunov guided deep reinforcement learning for service caching and task offloading in mobile edge computing," *Computer Networks*, 2024.

[2] J. Liu, C. Li, and Y. Luo, "Efficient resource allocation for IoT applications in mobile edge computing via dynamic request scheduling optimization," *Expert Systems with Applications*, vol. 255, p. 124716, 2024.

[3] M. K. Somesula, S. K. Mothku, and S. C. Annadanam, "Cooperative service placement and request routing in mobile edge networks for latency sensitive applications," *IEEE Systems Journal*, vol. 17, no. 3, pp. 4050–4061, 2023.

[4] M. K. Somesula, S. K. Mothku, and A. Kotte, "Deep reinforcement learning mechanism for deadline aware cache placement in device to device mobile edge networks," *Wireless Networks*, vol. 29, no. 2, pp. 569–588, 2023.

[5] M. Xie, J. Ye, G. Zhang, and X. Ni, "Deep reinforcement learning based computation offloading and distributed edge service caching for mobile edge computing," *Computer Networks*, vol. 250, p. 110564, 2024.

[6] X. Zhao, Y. Wu, T. Zhao, F. Wang, and M. Li, "Federated deep reinforcement learning for task offloading and resource allocation in mobile edge computing assisted vehicular networks," *Journal of Network and Computer Applications*, vol. 229, p. 103941, 2024.

[7] G. S. Chhabra, S. K. Satti, G. N. V. Rajareddy, et al., "Time and traffic aware collaborative task offloading with service caching replacement in cloud assisted mobile edge computing," *Cluster Computing*, vol. 28, p. 900, 2025.

[8] W. Ren, Z. Xu, W. Liang, H. Dai, O. F. Rana, P. Zhou, and G. Wu, "Learning driven service caching in MEC networks with bursty data traffic and uncertain delays," *Computer Networks*, vol. 250, p. 110575, 2024.

[9] B. Gow, T. Pollard, L. A. Nathanson, A. Johnson, B. Moody, C. Fernandes, N. Greenbaum, J. W. Waks, P. Eslami, T. Carbonati, A. Chaudhari, E. Herbst, D. Moukheiber, S. Berkowitz, R. Mark, and S. Horng, "MIMIC IV ECG: Diagnostic electrocardiogram matched subset," *PhysioNet*, 2023.

[10] C. Shang, Y. Huang, Y. Sun, and M. Guizani, "Joint computation offloading and service caching in mobile edge cloud computing via deep reinforcement learning," *IEEE Internet of Things Journal*, 2024.