

Energy-Aware Dynamic Task Offloading Using Deep Reinforcement Learning in Cloud–Edge–IoT Systems

Prof. Georg Sakeri

Professor of Networked Systems, University of Greenwich, UK

Email: g.sakeri@greenwich.ac.uk

Article Received 19-04-2026, Revised 24-05-2026, Accepted 08-06-2026

Author(s) Retains the Copyrights of This Article

Abstract

The rapid expansion of Internet of Things applications has increased the demand for continuous processing of computation-intensive tasks generated by resource-constrained devices. Conventional cloud-centric processing can introduce substantial transmission delays and communication energy consumption, particularly when large volumes of data are generated close to the network edge. Cloud–edge–IoT architectures provide a distributed alternative by placing computational resources closer to end devices; however, deciding where and when individual tasks should be processed remains challenging because device workloads, wireless conditions, queue states, available computing resources, and battery levels continuously change. This research proposes an energy-aware dynamic task offloading framework based on deep reinforcement learning for cloud–edge–IoT environments. The proposed framework models task placement as a sequential decision problem in which an intelligent agent observes device energy status, task characteristics, communication conditions, queue information, and available edge and cloud resources before selecting an execution location. The possible decisions include local processing, edge execution, and cloud processing, enabling the system to adapt its behavior to current operating conditions. A composite reward mechanism balances energy consumption, task completion delay, deadline violations, and resource utilization. The framework also incorporates dynamic resource availability so that overloaded edge nodes can be avoided while underutilized resources can be exploited. Deep reinforcement learning enables the offloading policy to improve through repeated interaction with changing system states without requiring an explicit analytical representation of every network transition. The proposed approach is intended to reduce unnecessary energy expenditure while maintaining acceptable response times for delay-sensitive IoT applications. Its performance can be evaluated using heterogeneous workloads and varying network conditions against local execution, greedy offloading, conventional optimization, and established reinforcement-learning strategies. The resulting framework provides a flexible basis for intelligent task placement and resource management across interconnected IoT devices, edge servers, and cloud resources.

Keywords: Energy-Aware Computing, Dynamic Task Offloading, Deep Reinforcement Learning, Cloud–Edge–IoT, Mobile Edge Computing, Resource Allocation, Energy Optimization, Latency-Aware Computing, Task Scheduling, Edge Intelligence, IoT Resource Management, Adaptive Offloading.

1. Introduction and Related Work

The proliferation of IoT devices has created a computing environment in which large numbers of sensors, cameras, wearable devices, industrial controllers, vehicles, and intelligent gateways continuously generate data. Many of these devices have limited processing capability and battery capacity, while the applications running on them increasingly require computationally demanding operations such as image analysis, machine learning inference, environmental monitoring, anomaly detection, and real-time decision support. Processing

every task locally can therefore result in high energy consumption and increased execution time. Conversely, sending all tasks directly to remote cloud infrastructure can introduce communication latency and additional energy expenditure. Edge computing addresses this problem by positioning computational resources closer to data-generating devices [1]. The cloud–edge–IoT paradigm combines the extensive computational capacity of cloud infrastructure with the proximity and responsiveness of edge resources. This architecture allows a task to be processed locally, transferred to a nearby edge server,

or transmitted toward a remote cloud platform. The appropriate choice depends on several dynamic parameters, including task size, computational complexity, wireless bandwidth, channel conditions, device battery level, edge-server workload, cloud availability, and task deadline. Consequently, task offloading is not a fixed scheduling problem but a dynamic decision problem whose optimal solution can change as system conditions evolve [2].

Mao, Zhang, and Letaief investigated dynamic computation offloading for mobile-edge systems and demonstrated the importance of jointly considering computational decisions and energy conditions [3]. Their work illustrates that offloading decisions can substantially affect the energy behavior of resource-constrained devices. Mao, Zhang, and Letaief also investigated green mobile-edge computing through Lyapunov optimization, demonstrating the relevance of dynamic resource-management techniques for systems where workload and energy states vary over time [4]. Dinh et al. formulated offloading as a task-allocation and computational-resource optimization problem, emphasizing the interaction between task placement and available edge resources [5]. These studies established important analytical foundations for energy- and latency-aware offloading.

Joint optimization of communication and computational resources has also received considerable attention. Sardellitti, Scutari, and Barbarossa investigated the joint optimization of radio and computational resources in multicell mobile-edge computing systems, showing that communication and computing decisions cannot always be optimized independently [6]. Such findings are particularly relevant to IoT systems because the energy required to transmit a task depends on communication conditions as well as task size and transmission power.

The emergence of deep reinforcement learning has provided an alternative to purely analytical optimization. Huang, Bi, and Zhang developed a deep reinforcement learning approach for online computation offloading in wireless-powered mobile-edge computing networks, demonstrating the applicability of DRL to dynamic offloading decisions [7]. Zhang et al. similarly investigated online computation offloading through deep reinforcement learning while considering energy and latency-related objectives [8]. These studies motivate the use of learning-based decision mechanisms for environments in which system states are continuously changing.

The main research challenge addressed in this work is therefore the development of a unified offloading mechanism capable of responding to dynamic task

requirements, device energy states, network conditions, and computing-resource availability. Instead of using a predetermined offloading threshold, the proposed framework formulates task placement as a sequential decision process. The learning agent observes the current environment, evaluates possible execution locations, receives a reward according to the resulting energy and latency behavior, and progressively improves its policy.

The major contribution of this study is a conceptual framework that integrates energy-aware task characterization, cloud-edge-IoT system modeling, DRL-based offloading, adaptive resource allocation, and energy-latency reward optimization within a common decision process. The framework is designed for heterogeneous IoT environments in which the computational and communication state may change between successive decisions.

2. System and Problem Formulation

The considered environment consists of a set of IoT devices, one or more edge servers, and cloud computing infrastructure. IoT devices generate computational tasks and can either execute them locally or transfer them to an edge or cloud resource. Edge servers provide intermediate computational capacity with lower communication distance than remote cloud resources, while the cloud provides large-scale processing capability but may involve greater transmission distance and communication delay.

A computational task is characterized by its input-data size, required number of CPU cycles, maximum tolerable delay, priority, and energy sensitivity. The task characteristics determine whether local processing is practical or whether remote execution should be considered. A computationally intensive task with a strict deadline may benefit from an edge server, whereas a small task with a low computational requirement may be completed locally without incurring transmission overhead.

The system state is represented by a collection of dynamic variables describing the current operating environment. These include the device battery level, processor utilization, task queue length, task size, computational requirement, deadline, wireless bandwidth, channel quality, transmission rate, edge-server workload, available CPU capacity, and cloud resource availability. The state representation is updated whenever a new task arrives or a relevant system condition changes.

The local execution time depends on the computational workload and device processing

capability. Remote execution additionally includes task transmission, queueing, edge or cloud processing, and result-return delays. Similarly, local energy consumption is primarily associated with CPU execution, whereas remote execution introduces communication energy in addition to the processing energy at the selected remote resource.

The offloading decision can be represented as a discrete action in which the agent selects local execution, edge execution, or cloud execution. The optimization objective is to determine an execution location that minimizes a weighted combination of energy consumption and task completion latency while satisfying resource and deadline constraints.

The overall objective can therefore be expressed conceptually as minimizing a combined cost:

$$J = \alpha E + \beta T + \gamma V + \delta R$$

where E represents energy consumption, T represents task completion time, V represents deadline or constraint violations, R represents resource inefficiency, and α , β , γ , and δ are weighting parameters.

The formulation is deliberately dynamic. The preferred execution location for one task may not remain appropriate for a later task because the device battery, wireless channel, edge queue, or cloud workload may have changed.

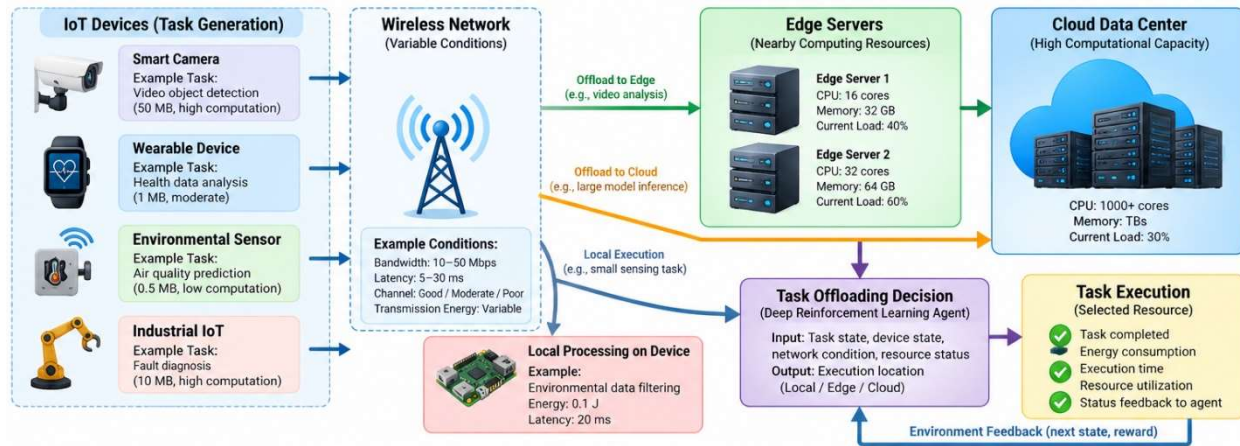


Figure 1. Cloud-Edge-IoT Task Offloading Architecture

The figure provides a visual representation of the relationship between IoT devices, task generation, wireless conditions, edge resources, cloud resources, the DRL decision agent, and adaptive task execution.

3. Proposed Deep Reinforcement Learning Framework

The proposed framework formulates task offloading as a Markov decision process. At each decision interval, the agent observes the current state of the cloud-edge-IoT environment and selects an action representing the execution location of the task. After the selected action is implemented, the environment returns information about execution time, energy consumption, resource utilization, and constraint satisfaction. This information is converted into a reward used to update the decision policy.

The state vector contains task-related, device-related, communication-related, and infrastructure-related information. Task-related information includes computational workload, input-data size, deadline, and priority. Device-related information includes battery level, CPU utilization, and available local

processing capacity. Communication information includes bandwidth, estimated transmission rate, and channel quality. Edge and cloud information includes available CPU resources, queue length, current utilization, and estimated service delay.

The action space contains three primary alternatives. The first is local execution, in which the task remains on the IoT device. The second is edge execution, in which the task is transferred to a selected nearby edge server. The third is cloud execution, in which the task is transmitted to cloud infrastructure.

A suitable DRL implementation can employ an actor-critic architecture or a value-based network depending on the action and resource-allocation formulation. For a discrete execution-location decision, a DQN-type architecture can be used. If CPU frequency, transmission power, bandwidth allocation, and execution location are jointly optimized as continuous or mixed actions, an actor-critic method such as DDPG or a related algorithm can be considered. This flexibility allows the framework to accommodate both simple task-placement decisions and more complex joint optimization.

The learning process begins with exploration of alternative decisions. As experience accumulates, the agent learns relationships between system states, actions, rewards, and future system conditions. The objective is not merely to select the execution location with the lowest immediate energy consumption. Instead, the agent learns policies that consider the

longer-term consequences of resource usage and workload distribution.

The proposed architecture is therefore intended to replace rigid offloading thresholds with an adaptive policy that can respond to changes in device energy, workload, communication conditions, and infrastructure congestion.

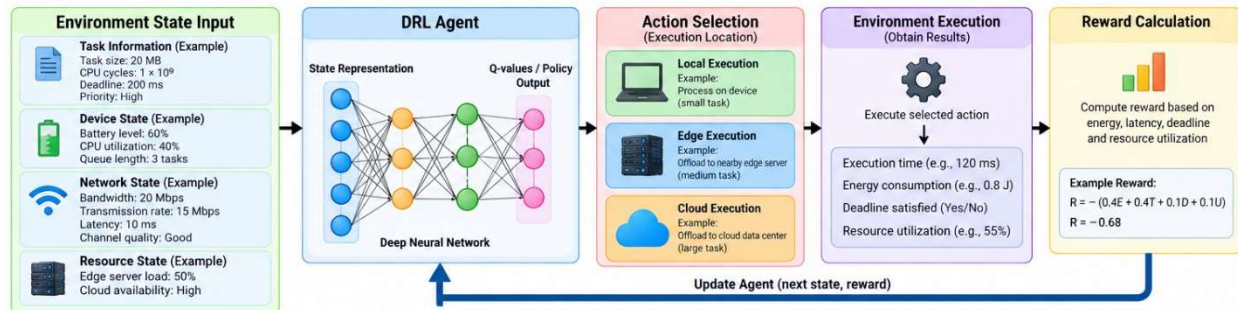


Figure 2. Deep Reinforcement Learning Offloading Decision Workflow

4. Adaptive Offloading and Resource Allocation

The adaptive offloading mechanism determines the execution location using the state information supplied to the DRL agent. When the device has sufficient computational capacity and the task is relatively small, local execution can avoid transmission overhead. When the task is computationally intensive and a nearby edge server has sufficient capacity, edge execution can provide a useful balance between communication delay and processing capability.

Cloud execution becomes relevant when edge resources are congested, when the computational requirement exceeds the available edge capacity, or when cloud infrastructure provides a more suitable execution environment. However, cloud execution can introduce additional communication delay and energy expenditure because task data must travel through the network.

The decision process therefore considers the current condition rather than assigning a permanent preference to one execution layer. This dynamic characteristic is particularly important in environments where many IoT devices simultaneously generate tasks. An edge server that is lightly loaded at one point may become congested after several devices offload tasks simultaneously.

Resource allocation is coupled with task placement. CPU resources at edge servers are allocated according to task requirements and current utilization. Communication resources can similarly be adjusted according to bandwidth availability and transmission requirements. The objective is to prevent a task-

placement policy from simply transferring congestion from IoT devices to edge infrastructure.

The proposed mechanism can additionally incorporate task priority. Delay-sensitive applications can be assigned higher latency penalties, whereas non-critical background tasks can tolerate longer execution times. This permits the DRL policy to adapt not only to system conditions but also to application requirements.

The framework can further support multiple edge servers by treating server selection as part of the action space. In such a configuration, the agent selects both an execution layer and an appropriate resource within that layer. This creates a more complex decision problem but allows the framework to exploit spatial diversity among edge resources.

5. Energy-Constrained Reward Optimization

The reward mechanism determines how the DRL agent evaluates the consequences of its decisions. A poorly designed reward may cause the agent to minimize one objective while degrading another. For example, optimizing exclusively for energy may lead to excessive local processing and unacceptable latency, whereas optimizing exclusively for latency may result in excessive remote execution and communication energy consumption.

The proposed reward therefore combines energy, latency, deadline compliance, and resource utilization. A conceptual reward function can be expressed as:

$$R_t = -(W_e E_t + W_l T_t + W_d D_t + W_u U_t)$$

where E_T is the energy consumed during decision interval t , T_i is task completion latency, D_i represents deadline violation, U_i represents resource inefficiency, and the w terms determine their relative importance.

A substantial penalty can be assigned when a task misses its deadline. This prevents the agent from achieving low energy consumption at the expense of application requirements. Similarly, a penalty can be introduced when an edge server becomes overloaded because of excessive offloading.

Battery-aware behavior can also be incorporated by increasing the energy penalty when the device battery approaches a predefined critical level. This allows the policy to become more conservative with respect to local computation when battery resources are scarce. The reward can also account for long-term resource effects. For example, sending every task to the same edge server may provide good immediate performance but cause future queue congestion. Reinforcement learning is suitable for this setting because the agent can learn policies based on accumulated experience rather than evaluating every decision independently.

6. Experimental Evaluation

The experimental evaluation should examine whether the proposed DRL framework can reduce energy consumption while maintaining acceptable task completion latency. The evaluation should include heterogeneous workloads and multiple operating

conditions so that the learned policy is tested under both stable and dynamic environments.

A suitable experimental environment should contain IoT devices with different computational capacities and battery levels, edge servers with different CPU capacities, and cloud resources with substantially larger processing capability. Network conditions should be varied by changing available bandwidth, transmission rate, and communication delay.

The benchmark scenarios should include local-only execution, cloud-only execution, a greedy offloading policy, a conventional optimization approach, and one or more reinforcement-learning baselines. The same task workloads and system conditions should be used across the competing methods to ensure that the comparison is meaningful.

The principal evaluation metrics should include average energy consumption per completed task, average task completion latency, deadline satisfaction rate, resource utilization, task completion rate, and DRL convergence behavior. Energy and latency should be reported separately because a single combined objective can conceal important differences. The experimental evaluation should also consider different workload intensities. Under light workload conditions, local processing may be practical for a substantial proportion of tasks. As workload increases, edge and cloud resources become increasingly important. The experiment should therefore include low-, medium-, and high-load scenarios.

Table 1. Recommended Experimental Results Table

Method	Average Energy Consumption	Average Latency	Deadline Satisfaction	Task Completion Rate	Resource Utilization
Local Execution	Actual result	Actual result	Actual result	Actual result	Actual result
Greedy Offloading	Actual result	Actual result	Actual result	Actual result	Actual result
Conventional Optimization	Actual result	Actual result	Actual result	Actual result	Actual result
Conventional RL	Actual result	Actual result	Actual result	Actual result	Actual result
Proposed DRL Framework	Actual result	Actual result	Actual result	Actual result	Actual result

Important: These cells should be replaced with values obtained from the actual implementation. Numerical values should not be invented merely to make the paper appear experimentally complete.

The evaluation should report mean performance over multiple independent runs where stochastic learning is involved. Standard deviation or confidence intervals should also be reported where appropriate. This is

particularly important for DRL because initialization, exploration, and training order can influence individual runs.

7. Energy-Latency Trade-off, Ablation and Scalability Analysis

Energy and latency are inherently conflicting objectives in many cloud-edge-IoT systems. Local

processing avoids transmission energy but can consume substantial device energy and require longer computation times. Edge processing can reduce latency because computational resources are closer to the device, although transmission energy and edge congestion must be considered. Cloud execution can provide extensive processing capacity but may introduce greater communication delay.

The proposed framework should therefore be evaluated using different reward-weight configurations. Increasing the energy weight should encourage the policy to conserve device energy, while increasing the latency weight should encourage faster task completion. The resulting operating points can be used to examine the energy-latency relationship.

An ablation analysis should separately remove major components of the proposed framework. For example, one experiment can remove battery information from the state vector, another can remove edge-resource information, and another can use a reward function without deadline penalties. The resulting changes in performance can demonstrate the contribution of each component.

Scalability should be investigated by increasing the number of IoT devices, generated tasks, edge servers, and simultaneous decision requests. The computational cost of state processing and policy inference should be measured alongside task-level performance. A practical system should maintain acceptable decision latency even as the number of participating devices increases.

The framework should also be tested under changing network conditions. Bandwidth reduction, temporary edge-server congestion, device battery depletion, and sudden workload increases can be introduced during evaluation. These experiments can determine whether the learned policy can adapt to environmental changes rather than only performing well under stationary conditions.

8. Limitations, Future Directions and Conclusion

8.1 Limitations

The proposed framework has several limitations. First, DRL requires repeated interaction with an environment during training, and obtaining representative training conditions can be difficult when real-world IoT systems are highly heterogeneous. A policy trained under one workload distribution may not immediately generalize to a substantially different deployment environment.

Second, the quality of the learned policy depends on the state representation. Missing information about battery conditions, edge queues, network bandwidth,

or task deadlines can cause the agent to make decisions using incomplete information. Reliable telemetry collection is therefore an important requirement.

Third, DRL training can involve significant computational overhead. Although policy inference may become relatively lightweight after training, model development and retraining can require substantial resources. This issue becomes more important when the target IoT environment itself has limited computing capability.

Fourth, reward-weight selection remains an important design issue. Different applications have different priorities. A healthcare monitoring application may prioritize deadline satisfaction, while an environmental sensor network may prioritize battery lifetime. Consequently, a single fixed reward configuration may not be optimal for every deployment.

8.2 Future Directions

Future research can investigate multi-agent deep reinforcement learning in which individual edge nodes or device groups operate as cooperating agents. Such a model could potentially distribute decision-making across large geographically distributed infrastructures. Federated reinforcement learning is another relevant direction because IoT and edge operators may not be able to transfer raw operational data to a centralized training environment. Federated approaches could allow participating nodes to contribute model updates while keeping local telemetry within their respective environments.

Transfer learning can also be explored to reduce retraining requirements when a trained policy is moved between different network configurations. This is particularly relevant when the number of devices, edge-server capacity, or workload characteristics change.

Further work should also consider security-aware offloading. An execution location may be computationally efficient but unsuitable if the destination resource does not satisfy application security or privacy requirements. Future models can therefore incorporate trust, data sensitivity, privacy constraints, and security risk into the reward formulation.

Real-world testbed validation represents another important direction. Simulation provides controlled experimentation, but deployment on actual IoT devices and edge servers would allow practical issues such as wireless interference, processor variability, queue behavior, and energy-measurement uncertainty to be examined directly.

8.3 Conclusion

This research presented an energy-aware dynamic task-offloading framework for cloud-edge-IoT systems using deep reinforcement learning. The proposed approach treats task placement as a sequential decision problem in which the agent observes task characteristics, device energy conditions, communication status, edge-server resources, and cloud availability before selecting an execution location.

The framework combines local, edge, and cloud execution options with adaptive resource allocation and an energy-latency-aware reward mechanism. The formulation recognizes that the most appropriate execution location can change as workload, network conditions, battery levels, and computing-resource availability evolve.

The literature demonstrates that edge computing can reduce the distance between data generation and computation, while prior offloading studies have established the importance of jointly considering communication, energy, and computational resources [1]–[6]. DRL-based approaches have subsequently provided mechanisms for learning online offloading policies under dynamic conditions [7]–[14]. The proposed framework extends this direction conceptually by integrating task placement, energy awareness, dynamic resource availability, and adaptive decision-making into a unified cloud-edge-IoT architecture.

Experimental validation should compare the proposed method with local execution, greedy offloading, conventional optimization, and established reinforcement-learning methods using identical workloads and system conditions. Particular attention should be given to energy consumption, latency, deadline satisfaction, resource utilization, scalability, and adaptation to changing environments. Actual experimental measurements should replace the result placeholders in Table 1 before publication.

References

- [1]. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. This foundational work discusses bringing computation and services closer to end devices to address latency and resource limitations.
- [2]. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [3]. Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3604, 2016.
- [4]. Y. Mao, J. Zhang, and K. B. Letaief, "A Lyapunov Optimization Approach for Green Mobile-Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7529–7543, 2016.
- [5]. T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Resource Optimization," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.
- [6]. S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.
- [7]. L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.
- [8]. Y. Zhang, T. Liu, Y. Zhu, and Y. Yang, "A Deep Reinforcement Learning Approach for Online Computation Offloading in Mobile Edge Computing," *Proceedings of the IEEE International Conference on Communications*, 2020. The work investigates online offloading and jointly considers task latency and energy consumption.
- [9]. R. Zhao, X. Wang, J. Xia, and L. Fan, "Deep Reinforcement Learning Based Mobile Edge Computing for Intelligent Internet of Things," *Physical Communication*, vol. 43, Art. 101184, 2020. The study uses DQN-based decision-making for task offloading and considers latency and energy-related system costs.
- [10]. M. Chen, T. Wang, S. Zhang, and A. Liu, "Deep Reinforcement Learning for Computation Offloading in Mobile Edge

- Computing Environment,” *Computer Communications*, vol. 175, pp. 1–12, 2021. The study models offloading as an MDP and employs DDPG for distributed resource provisioning.
- [11]. Z. Chen and X. Wang, “Decentralized Computation Offloading for Multi-User Mobile Edge Computing: A Deep Reinforcement Learning Approach,” *EURASIP Journal on Wireless Communications and Networking*, 2020, Art. 188.
 - [12]. T. Alfakih, M. Hassan, A. Gumaei, et al., “Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA,” *IEEE Access*, vol. 8, pp. 54074–54084, 2020, doi: 10.1109/ACCESS.2020.2981434.
 - [13]. L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, “Delay-Aware and Energy-Efficient Computation Offloading in Mobile Edge Computing Using Deep Reinforcement Learning,” *IEEE Internet of Things Journal*, 2021. The work considers dynamic edge environments, multiple edge servers, task deadlines, and energy consumption.
 - [14]. L. Liao, Y. Lai, F. Yang, and W. Zeng, “Online Computation Offloading with Double Reinforcement Learning Algorithm in Mobile Edge Computing,” *Journal of Parallel and Distributed Computing*, vol. 171, pp. 28–39, 2023. The proposed DRLCO framework jointly addresses offloading, CPU frequency, and transmission-power decisions.
 - [15]. S. K. Nalluri, V. T. Bathini and V. K. B. Parasaram, "AI-Augmented Manufacturing Execution Systems for Precision Healthcare: A Scalable Framework for Smart Genomic and Proteomic Production," *2026 IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC)*, Houston, TX, USA, 2026, pp. 1-6, doi: 10.1109/SATC69565.2026.11542384.
 - [16]. P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, and K. Li, “A Survey on Computation Offloading in Edge Systems: From the Perspective of Deep Reinforcement Learning Approaches,” *Computer Science Review*, vol. 53, Art. 100656, 2024.
 - [17]. A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, “A Survey on the Computation Offloading Approaches in Mobile Edge Computing: A Machine Learning-Based Perspective,” *Computer Networks*, vol. 182, Art. 107496, 2020.
 - [18]. A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, “A Review on the Computation Offloading Approaches in Mobile Edge Computing: A Game-Theoretic Perspective,” *Software: Practice and Experience*, vol. 50, pp. 1719–1759, 2020.
 - [19]. “A Survey on Computation Offloading in Mobile Edge Computing Networks,” *Journal of Network and Computer Applications*, 2022. The survey examines delay minimization, energy minimization, resource allocation, server selection, and RL-based approaches.
 - [20]. “A Survey on Computation Offloading Modeling for Edge Computing,” *Journal of Network and Computer Applications*, vol. 169, Art. 102781, 2020. The work reviews offloading decisions, task partitioning, resource allocation, MDPs, optimization, and learning-based models.
 - [21]. “Energy-Efficient Computation Offloading and Resource Allocation for Delay-Sensitive Mobile Edge Computing,” *International Journal of Distributed Sensor Networks*, 2018. The work jointly considers completion time, energy consumption, CPU frequency, transmission power, and offloading decisions.
 - [22]. Koneni, C., S. Lokula, and L. Panjiyar. "A Cloud-Native Approach to Statistical Anomaly Detection and Automated Data Quality Validation." *International Journal of Data Science and IoT Management System* 4.4 (2025): 533-543
 - [23]. “Resource Allocation and Computation Offloading with Data Security for Mobile Edge Computing,” *Future Generation Computer Systems*, 2019. The study formulates joint resource allocation and offloading while considering energy, time, and security requirements.
 - [24]. B. Huang et al., “Security and Cost-Aware Computation Offloading via Deep Reinforcement Learning in Mobile Edge Computing,” *Wireless Communications and Mobile Computing*, 2019, Art. 3816237. The work formulates secure offloading as an MDP while considering energy, delay, and task-loss-related costs.

- [25]. "Deep Reinforcement Learning-Based Computation Offloading and Distributed Edge Service Caching for Mobile Edge Computing," *Computer Networks*, vol. 250, Art. 110564, 2024. The study jointly considers latency, energy, resource allocation, task offloading, and edge service caching using LSTM-DDPG.
- [26]. C. Koneeni and S. Lokula, "Evidence-Quality Telemetry for Cloud Incident Response: Detecting Gaps, Drift, and Integrity Failures in Large-Scale Observability Pipelines," *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*, Boston, MA, USA, 2026, pp. 1-6, doi: 10.1109/ISDFS69419.2026.11459097.
- [27]. "Energy-Aware Dynamic Task Offloading Using Deep Reinforcement Transfer Learning in SDN-Enabled Edge Nodes," *Internet of Things*, vol. 25, Art. 101118, 2024. The study combines dynamic partial offloading, transfer learning, trust evaluation, and energy-aware decision-making.
- [28]. A. K. Chakravarthy, T. Rama Reddy, and N. Ramakrishnaiah, "EdgeAutoOffloadAI: A Deep Reinforcement Learning Framework for Adaptive Task Offloading and Resource Management in Mobile Edge Computing," *Discover Computing*, vol. 29, Art. 136, 2026. The recent work combines DRL with Transformer-based state representation and considers latency, energy, resource utilization, and fairness in dynamic edge environments.