IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 2, February, 2026, pp. 371-383
Journal homepage: http://iieta.org/journals/jesa

Latency Optimization for UAV Networks via Latency-Aware Long-Term Deep Deterministic Policy Gradient in Slicing-Enabled MEC Systems

Hanan Thamer[1*], Suhad Faisal Behadili[2]

[1] Computer Science Department, College of Science, University of Baghdad, Baghdad 10001, Iraq
[2] Big Data Department, College of Artificial Intelligence, University of Baghdad, Baghdad 10001, Iraq

Corresponding Author Email: hanan.thamer1601b@sc.uobaghdad.edu.iq

https://doi.org/10.18280/jesa.590207

Received: 4 December 2025
Revised: 9 February 2026
Accepted: 24 February 2026
Available online: 28 February 2026

Keywords:
low-latency communication, mobile edge computing, network slicing, unmanned aerial vehicles

ABSTRACT

Smart delivery systems increasingly employ unmanned aerial vehicles (UAVs) to support time-critical operations. During flight, UAVs operate under highly dynamic conditions, including fluctuating network quality, heterogeneous task requirements, and limited onboard battery capacity. These factors significantly complicate task execution and resource management. To address these challenges, this study proposes a task-offloading framework that integrates mobile edge computing (MEC), network slicing (NS), and a latency-aware reinforcement learning model based on the long-term deep deterministic policy gradient (LLDDPG) algorithm. NS is explicitly incorporated into the baseline framework through slice-based task classification (S1, S2, and S3) and predefined allocation policies. The proposed LLDDPG model operates within the same slicing-enabled MEC environment and focuses on learning latency-aware task-offloading decisions, without performing slice-level optimization such as slice selection or inter-slice resource allocation. Offloading decisions are dynamically determined according to task size, deadline constraints, and instantaneous wireless channel conditions. Each task is assigned to the MEC node expected to provide the most efficient processing performance. Task-offloading is formulated as a sequential decision-making process, where actions are continuously updated as new tasks arrive. Experimental results demonstrate the effectiveness of the LLDDPG-based approach in learning latency-aware task-offloading decisions within a slicing-enabled MEC environment.

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) are used in wireless communication systems due to mobility and flexible deployment. UAVs can provide coverage in areas without fixed infrastructure. These properties support applications such as emergency response, infrastructure inspection, transportation, and delivery services [1]. Reliable low-latency communication remains difficult to achieve. UAV movement is continuous. Wireless channel conditions change frequently. This results in unstable links and variable transmission quality [2]. In many UAV architectures, task processing is performed by remote cloud servers. This design introduces additional transmission delay. It also increases backhaul bandwidth usage. These factors limit performance in latency-sensitive applications that require timely responses [3]. Recent studies investigate the integration of network slicing (NS) and mobile edge computing (MEC) in UAV networks. As shown in Figure 1, the high-level architecture of NS in a 5G system, including RAN, transport, core network, and service-specific slice instances (eMBB, mMTC, URLLC) [4]. MEC enables task execution at edge nodes located closer to the UAV. This reduces end-to-end latency [5]. NS allows multiple logical services to operate on shared infrastructure. Each service is assigned specific quality-of-service constraints [6].

In delivery operations, UAV-generated tasks differ in urgency, size, and timing requirements. The communication layer must therefore support variable task demands. Resource management mechanisms must adapt to changing conditions during operation [7]. The joint use of MEC and NS enables distributed allocation of communication and computation resources. This approach supports latency reduction, energy efficiency, scalability, and service reliability under dynamic mission conditions [8]. Despite these developments, many existing systems still employ static or heuristic offloading strategies. Such approaches are not suitable for dynamic delivery scenarios where communication variability, workload pressure, and energy constraints coexist [4].

The purpose of this study is to propose a framework for offloading tasks that is based on learning and that utilizes latency-aware long-term deep deterministic policy gradient (LLDDPG) in order to make intelligent, real-time judgments regarding the offloading of UAV delivery systems and therefore address these challenges. In this study, NS is used to model differentiated service classes within the baseline layer via slice-based task grouping and predefined allocation rules. The main contribution is introducing an LLDDPG-based decision layer that learns latency-aware task offloading actions under the same slicing-enabled MEC setting, rather than optimizing slice management itself. The model increases the

responsiveness and efficiency of time-sensitive scenarios by optimizing task distribution on the basis of context-aware characteristics, which include deadline compliance, MEC load, and link quality.

This paper is focused on the integration of MEC, NS, and intelligent resource orchestration within UAV networks to support next-generation UAV systems. The rest of this paper is organized as follows. Section two gives a Background of UAVs, MEC, NS, and LLDDPG algorithm. While section three related work, section four presents the utilized methodology. Section five presents the result and discussion. Additionally, it highlights the comparison among the outcomes of this study and previous ones. Finally, section six explores the conclusions and brought attention to future work.

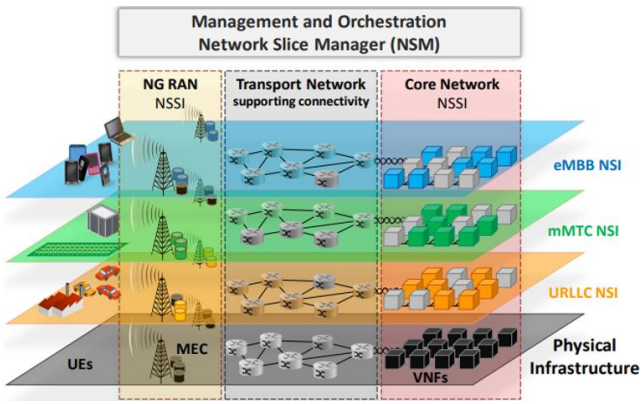

Figure 1. High-level network slicing (NS) architecture and overview [4]

2. BACKGROUND

2.1 Unmanned aerial vehicles communication challenges

UAVs have become more important in recent years across a wide range of wireless uses. This is because they are flexible, don't cost much to run, and are easy to set up. These traits make UAVs appealing, but adding them to communication networks causes a number of technical issues that have a direct effect on the stability and performance of the system [1]. Because a UAV is so mobile, especially because it can move easily in three dimensions, the network topology around it is almost never fixed. When the drone changes height or direction, the wireless channel changes too. This makes the signal power and link reliability go up and down. Such variations may lead to intermittent delays or temporary link failures, which pose serious limitations for latency-sensitive applications such as real-time monitoring, emergency response, and delivery missions [9]. As a result, UAV-based services require communication links that can sustain low-latency performance under rapidly changing conditions. In practical deployments, UAVs often operate in environments with sparse or heterogeneous infrastructure, where network conditions vary frequently and unpredictably. Traditional ground-based communication systems are generally optimized for static or slowly varying traffic patterns and are therefore ill-suited to support highly dynamic, on-demand UAV workloads [1]. Addressing these challenges requires scalable communication mechanisms and adaptive resource-management strategies capable of responding to continuous changes in UAV network topology and traffic demand [10, 11].

2.2 Mobile edge computing

Modern wireless networks are putting pressure on traditional cloud systems, and many of today's applications simply expect more than what distant cloud servers can deliver. Tasks such as smart surveillance, AR/VR, and various autonomous services produce large data streams and depend on very tight latency requirements. Because of that, the classical cloud model often becomes too slow to keep up. MEC, introduced by ETSI, was meant to close this gap by shifting computation and storage closer to the radio access network, which naturally results in faster and more responsive service [6].

Compared to mobile cloud computing (MCC), MEC keeps the processing near the point where the data is generated. This not only reduces end-to-end delay but also removes part of the load from the backhaul network. UAVs operate with limited onboard computational and energy resources. For this reason, offloading support is required [6]. MEC servers can be deployed at multiple locations. MEC servers can be deployed at different locations within the network. These locations include cellular base stations, roadside units, and mobile platforms such as UAVs and ground vehicles [2].

A typical MEC architecture consists of several functional components, including virtualization layers, orchestration mechanisms, traffic steering, and local breakout functions. These components enable computational tasks to be executed at the network edge, closer to data sources and end users. Within the 5G service-based architecture (SBA), MEC operates in conjunction with NS. Multiple services share the same physical infrastructure while maintaining performance isolation between slices [8]. In addition, hybrid MEC (H-MEC) configurations combine fixed edge nodes with mobile computing resources. Such configurations are applied in environments where coverage conditions change frequently, including disaster-response scenarios, dense urban areas, and UAV-based delivery networks [2]. Recent MEC deployments also integrate complementary technologies such as non-orthogonal multiple access (NOMA), machine-learning-based resource optimization, and wireless power transfer. These technologies aim to improve resource utilization and energy efficiency at the edge [2]. Despite these capabilities, MEC introduces several challenges. The distributed nature of edge resources increases coordination complexity. Mobility may interrupt both computation and communication processes. Furthermore, decentralized architectures raise additional security and reliability concerns that must be carefully managed [6].

Still, MEC is not without challenges. Wide distribution of resources complicates coordination, mobility can interrupt both computation and communication, and the decentralized nature of the architecture raises additional security concerns that must be addressed.

2.3 Network slicing

NS is defined as a function of 5G networks that enables a physical infrastructure to be partitioned into multiple logical slices. Each slice is configured to meet specific service requirements [12]. This approach is applied in several domains, including IoT systems, automotive communication, and UAV networks.

The 3GPP standard specifies three main slice categories: massive Machine-Type Communication (mMTC), Ultra-

Reliable Low-Latency Communication (URLLC), and enhanced Mobile Broadband (eMBB) [4]. In UAV-related applications, URLLC is commonly associated with control and command signaling, mMTC with sensor data transmission, and eMBB with high-data-rate services such as image or video delivery [13].

NS enables differentiated allocation of network resources according to quality-of-service (QoS) requirements. Resource allocation is performed based on task characteristics and service constraints. Architectures such as AirSlice apply QoS-based allocation mechanisms for UAV traffic, assigning low-latency resources to control-related tasks and higher bandwidth to data-intensive transmissions [12]. Slice configuration and management are supported through software-defined networking (SDN) and network function virtualization (NFV). These technologies allow slice parameters to be adjusted during system operation without relying on fixed configurations.

The slicing workflow consists of several functional stages. These stages include slice admission control, resource allocation, traffic scheduling, and orchestration [14]. Each stage operates as part of the overall resource management process, as shown in Figure 2.

UAV missions exhibit heterogeneous requirements. Some missions require strict latency constraints. Others require sustained bandwidth or basic connectivity. When UAVs operate as temporary network nodes, slicing supports coverage extension and link stability in areas with limited signal quality [4].

Despite these advantages, NS introduces several unresolved challenges. In this work, NS is discussed to provide background context on its role in modern MEC-enabled networks. The proposed framework does not aim to resolve these open challenges; instead, slicing is assumed as an infrastructure-level capability that enables differentiated task handling in the baseline configuration.

Ensuring isolation and security among slices remains difficult, especially under high mobility conditions where UAVs frequently change positions, channels, and network associations [4]. As a result, efficient mobility management and slice continuity remain active research problems in UAV-enabled networks [13].

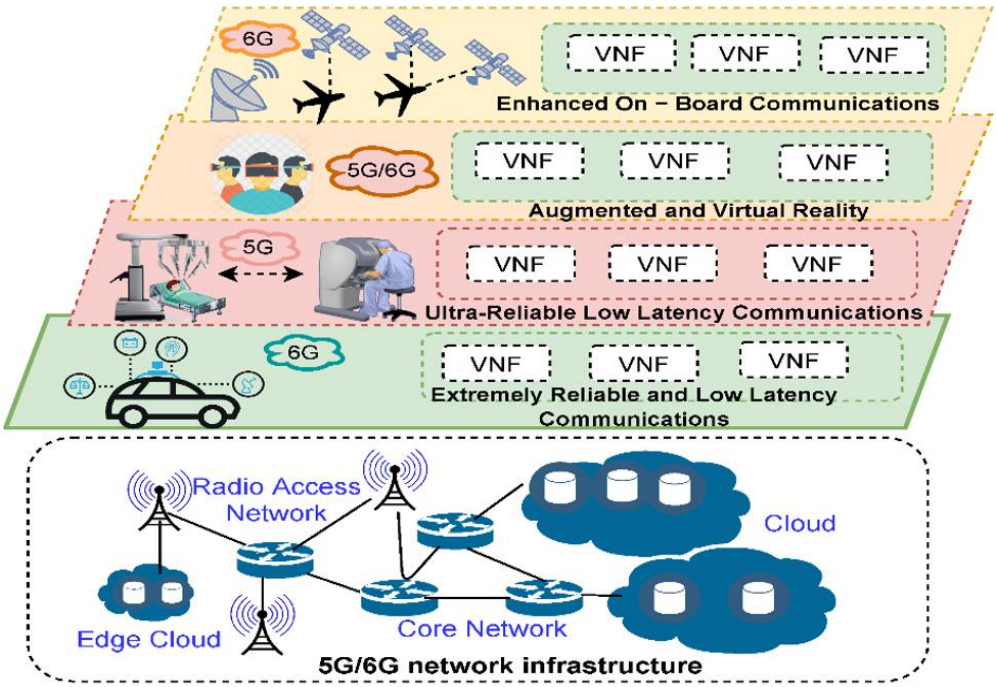

Figure 2. Phases of resource management in 5G/6G network slicing (NS) admission control, resource allocation, scheduling, and orchestration [14]

3. RELATED WORK

Recent studies in wireless networking have increasingly examined how UAVs can be integrated into next-generation systems, particularly when MEC and NS are part of the architecture [7]. UAVs are no longer viewed only as platforms for sensing or imaging; in many works, they also function as temporary base stations, relays, or lightweight edge-computing nodes capable of supporting 5G-class services such as eMBB, URLLC, and mMTC [15]. Introducing MEC changes the way tasks are processed, since computation can occur closer to the UAV rather than being forwarded to distant cloud servers. This naturally shortens the delay and reduces the load on the backhaul [16]. NS contributes in a similar way by dividing resources into virtual segments, allowing each service category to receive the level of performance it requires [17]. Previous studies have investigated UAV network performance under dynamic conditions [5]. Several works analyze the relationship between communication constraints and computational workload in UAV-based systems [17]. Other studies focus on communication aspects. These include line-of-sight (LoS) maintenance, Doppler effect handling, and spectrum management during three-dimensional UAV movement [18]. Existing solutions address individual components of the system. Joint consideration of task offloading, NS, and MEC orchestration is limited. Integrated frameworks operating under dynamic and realistic UAV conditions are not widely reported in current literature [19]. Many proposed methods rely on static or heuristic decision rules. These methods exhibit limited adaptability in UAV

delivery and surveillance scenarios where network conditions, workload, and energy constraints change during operation [20]. Learning-based approaches are therefore investigated to support adaptive offloading decisions in heterogeneous UAV–MEC systems [17].

As shown in Table 1, which summarizes selected prior studies that apply DDPG-based or latency-aware reinforcement-learning models in UAV-assisted MEC and NS contexts. The comparison focuses on system objectives, deployment assumptions, and reported performance metrics.

Table 1. Comparison of prior studies that applied DDPG-based or latency-aware reinforcement learning methods in UAV–MEC and NS environments, showing their scenarios, optimization goals, and performance improvements

Ref.	Reference (Authors, Year)	Scenario / Environment	RL Method	Objective	Main Reported Results
[21]	"Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach," Wireless Networks.	UAV-assisted MEC with UAV serving mobile users	DDPG (single agent)	Minimize maximum processing delay	Achieved significant delay reduction and fast convergence compared with DQN and heuristic baselines.
[22]	"Latency-aware task offloading in UAV-assisted edge computing," HCMCOU Journal of Science.	UAV + IoT devices with MEC	Latency-aware DDPG	Maximize task completion under latency constraints	DDPG achieved 100% success rate, exceeding DQN (95.93%) and the full-offloading baseline (82.73%).
[23]	"Lyapunov-based deep deterministic policy gradient for energy-efficient task offloading in UAV-assisted MEC," Drones.	UAV-assisted MEC with time-slotted operation	LyDDPG (DDPG + Lyapunov long-term optimization)	Reduce energy while maintaining stability and low delay	Reduced total energy consumption by ≥ 16% while preserving low latency and system stability.
[24]	"DDPG-based computation offloading strategy for maritime UAV," Electronics.	Maritime UAV–MEC network (MIoT)	DDPG	Minimize total task execution time	DDPG outperformed AC and DQN with lower execution time, faster convergence, and more stable performance.
[25]	"Deep reinforcement learning-based optimization for end-to-end network slicing with control-and user-plane separation," IEEE Transactions on Vehicular Technology.	5G Network slicing (E2E)	DDPG (long-term utility optimization)	Maximize long-term slice performance	Achieved superior slice utility, coverage, throughput, and delay compared with existing slicing strategies.
[26]	"Task offloading for multi-UAV edge computing with DRL," Cluster Computing.	Multi-UAV cooperative MEC	DDPG + DTLCM	Improve task distribution and reduce latency/energy	DDPG + DTLCM outperformed DQN and Q-Learning in stability, adaptiveness, and delay reduction.

Note: DDPG = deep deterministic policy gradient; UAV–MEC = unmanned aerial vehicle–mobile edge computing; NS = network slicing

4. METHODOLOGY

4.1 Task modeling and dataset preparations

An initial configuration was evaluated using size-based task assignment. In this configuration, tasks were assigned according to file size only. Slice S3 recorded the highest processing delay under this setting. A load-balancing mechanism was then applied. Task distribution across MEC nodes changed after this step. Overall processing delay decreased compared to the initial configuration.

However, the role of NS is applied in the baseline configuration by grouping tasks into slices (S1, S2, S3) with predefined processing characteristics. These slice-based rules define the baseline behavior used for evaluation. The proposed LLDDPG does not perform slice selection or inter-slice resource optimization; instead, it learns task offloading decisions under the same slicing-enabled MEC environment.

The LLDDPG-based offloading policy was evaluated next. Under this configuration, total processing time decreased from 10,479 s to 3,795 s. Slice-level processing times were recorded as follows: 425 s for S1, 320 s for S2, and 3,050 s for S3. This configuration was selected for subsequent experiments. A second experimental phase was conducted using a dataset of more than 28,000 UAV images. The dataset was obtained from previous studies [27, 28]. Each image was treated as an independent task. Additional task parameters were assigned. These parameters included deadline constraints, priority labels, and estimated transmission energy values.

Task metadata was extracted during preprocessing. The extracted information included file path, image resolution, and file size. This metadata was used for dataset organization and simulation input preparation. The retrieved information was organized and stored for later stages of processing.

The information that was extracted was put together using pandas into an organized CSV file. There are now more factors for tasks, like randomly chosen deadlines (100-1000 ms), priority levels based on directory labels, and energy estimates based on data size and communication cost. The Latency-aware LLDDPG model was trained and tested in a simulated MEC setting using this task-level dataset.

By switching from static size-based categorization to dynamic task modeling, the better framework made offloading choices that were more flexible and aware of the situation.

This made LLDDPG even better at working in real UAV–MEC situations. Table 2 presents sample entries from the synthesized task dataset used in the offloading experiments.

The task dataset described in Section 4.1 serves as the input to the system architecture presented in the following section. Each task extracted from the dataset is treated as an independent offloading unit characterized by its size, deadline, and priority. These task attributes are directly mapped to the architectural components of the proposed UAV–MEC system, where the LLDDPG agent observes task-level and system-level states to make offloading decisions.

Table 2. Synthesized an unmanned aerial vehicle (UAV) task dataset used for offloading simulation

Image Name	Size (MB)	Resolution	Deadline (ms)	Priority
230322_144646_476.jpg	24.49	9504 × 6336	1200	High
230322_144649_296.jpg	23.92	9504 × 6336	1500	Medium
230322_144650_606.jpg	23.17	9504 × 6336	800	High
IoU-0.23_p77_.jpg	0.03	32 × 40	600	Low
IoU-0.25_p25_.jpg	0.03	38 × 31	900	Medium

4.2 The proposed system architecture

The proposed architecture enables intelligent task offloading in UAV-based delivery systems by integrating MEC, NS, and a Latency-aware LLDDPG agent. It consists of four layers:

(1) UAVs that generate and offload tasks,

(2) A communication layer using 5G with variable latency,

(3) A distributed MEC Layer with heterogeneous edge nodes,

(4) A decision layer powered by RL for latency-aware.

Figure 3 shows the interaction among components for dynamic, real-time offloading.

The proposed model integrates several modular components to enable intelligent task offloading in UAV-based smart delivery networks. These components include:

- UAV Agent: Captures tasks from delivery missions and initiates the offloading process based on task characteristics and UAV status.
- Task Generator: Transforms real-world UAV image data into structured tasks containing file size, deadline, and energy estimates.
- MEC Node Simulator: Emulates heterogeneous edge servers with dynamic load levels and response times.
- LLDDPG Controller: This is the reinforcement-learning agent that watches what's going on in the system task details, the UAV's status, and which MEC nodes are available. Based on this information, it chooses where to send each task to reduce delay and save energy.
- All parts of the system keep exchanging information in a continuous loop, which helps the UAV make fast, context-aware decisions as conditions change.

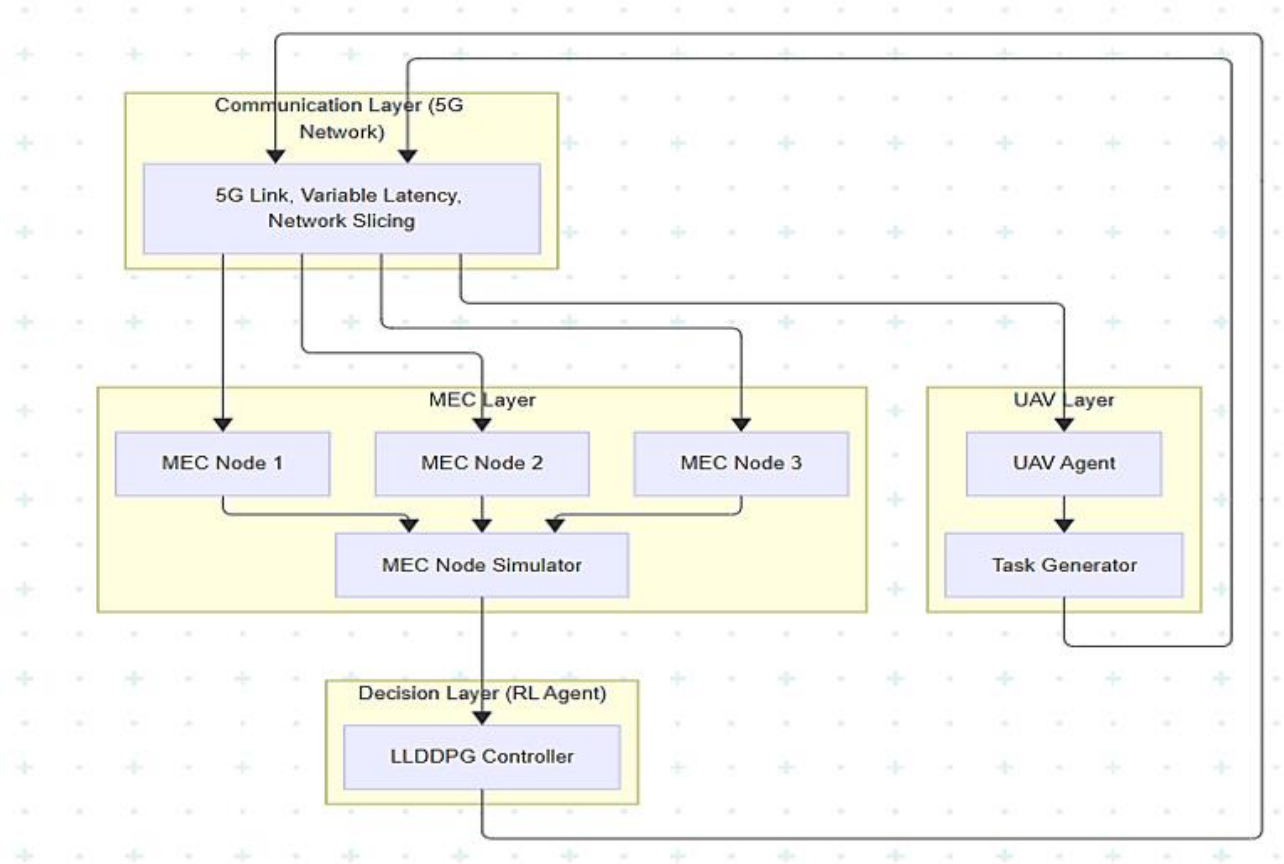

Figure 3. The proposed system architecture

4.2.1 The proposed model components

The LLDDPG is an extension of the standard deep deterministic policy gradient (DDPG) algorithm, designed for task offloading in latency-sensitive UAV–MEC environments. Unlike conventional DDPG, which optimizes a generic long-term reward, LLDDPG is tailored to account for task-level latency characteristics and sequential decision-making under dynamic network and workload conditions. The model applies a reinforcement-learning approach to task offloading in UAV delivery scenarios. The operating environment changes during execution. Offloading decisions are generated using a LLDDPG algorithm. Static decision rules are not used. Task assignment is updated at each decision step. The update depends on the current network state and available computation resources. Communication conditions and MEC server availability vary over time. Offloading decisions are recalculated accordingly. The choice changes depending on whatever is happening at that moment: how close a task is to missing its deadline, how busy the MEC servers look, how much battery the UAV still has, and even how good or weak the wireless link is at that instant.

The operating environment changes during execution. Offloading decisions are generated using a LLDDPG algorithm. Static decision rules are not used. Task assignment is updated at each decision step. The update depends on the current network state and available computation resources. Communication conditions and MEC server availability vary over time. Offloading decisions are recalculated accordingly. The choice changes depending on whatever is happening at that moment: how close a task is to missing its deadline, how busy the MEC servers look, how much battery the UAV still has, and even how good or weak the wireless link is at that instant.

• Model overview

Each UAV creates its own set of tasks while it moves through its delivery route. These tasks don't all look the same some are larger, some have tighter deadlines, and some need to be handled more urgently than others.

When a new task appears, the UAV has to make a quick decision:

- Either try to process it on its own with its limited onboard resources;
- Or send it to one of the MEC servers available, where each server has a different amount of computing power.

This decision is not made heuristically, but instead through an RL-based controller trained to optimize task distribution across the network.

• LLDDPG-based offloading

The decision-making process is modeled as a continuous-state, continuous-action reinforcement learning problem. The LLDDPG agent observes a state vector that includes:

- UAV battery level
- Task file size
- Deadline in milliseconds
- Link quality between UAV and MEC nodes
- Current processing load on each MEC server

Based on this input, the agent selects an action from a continuous action space representing offloading probabilities to each MEC node (or staying local). The policy is optimized using an actor–critic architecture.

- The actor proposes an action (offloading decision);
- The critic evaluates its expected cumulative reward.

The reward function is designed to minimize end-to-end latency, avoid missed deadlines, and reduce energy consumption, encouraging the agent to learn efficient, real-time decisions under uncertainty. Figure 4 shows the LLDDPG-based offloading workflow. The agent observes UAV and task-related parameters and learns to make optimal offloading decisions using a reinforcement learning loop.

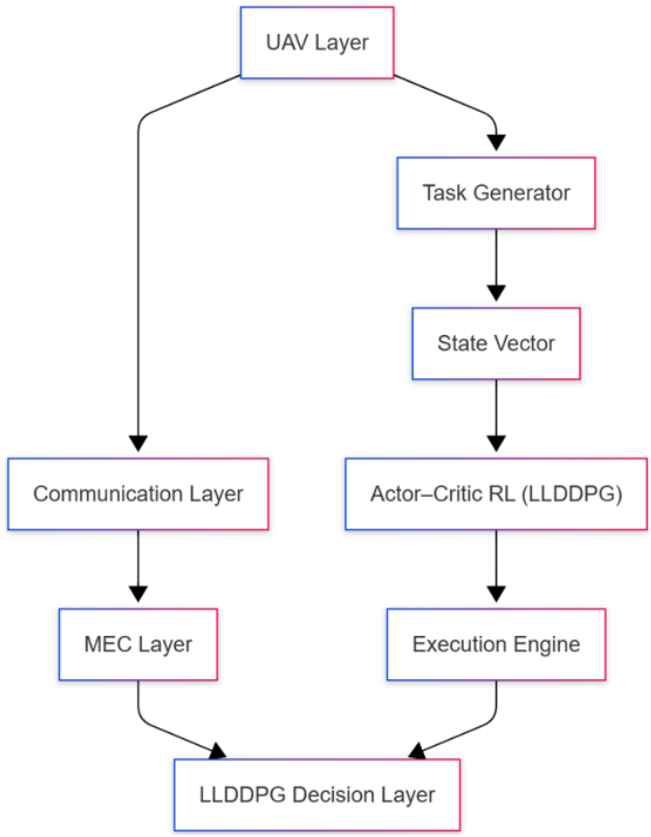

Figure 4. LLDDPG-based offloading workflow

Note: LLDDPG: latency-aware long-term deep deterministic policy gradient.

To explicitly incorporate latency awareness into the learning process, the reward function is designed to penalize end-to-end task delay and deadline violations. The immediate reward at decision step t is defined as:

$$r_t = -(\alpha \cdot L_t + \beta \cdot V_t) \quad (1)$$

where, L_t denotes the end-to-end latency experienced by task t, V_t represents the deadline violation indicator (equal to 1 if the task misses its deadline and 0 otherwise), and α and β are weighting coefficients controlling the relative importance of latency minimization and deadline satisfaction.

The long-term optimization aspect of LLDDPG is achieved by maximizing the expected cumulative discounted reward over sequential task arrivals. Instead of optimizing each offloading decision independently, the agent learns policies that account for the long-term impact of current actions on future system states, such as MEC load evolution, resource availability, and task congestion. This enables more stable and efficient offloading behavior under continuously changing UAV–MEC conditions.

Compared to the standard DDPG algorithm, which optimizes a generic long-term reward without domain-specific awareness, LLDDPG integrates latency-sensitive objectives and task-level constraints into the learning process. This allows the agent to prioritize time-critical tasks and adapt its

offloading decisions based on dynamic MEC load and wireless link conditions. As a result, LLDDPG is better suited for latency-sensitive UAV–MEC scenarios than conventional DDPG. The overall workflow of the proposed LLDDPG-based task offloading process is summarized in Algorithm 1.

Algorithm 1. LLDDPG-Based Task Offloading Procedure

Initialize the LLDDPG agent and system parameters
Initialize replay memory
for each training episode do
  Observe the current system state (task features, UAV status, MEC load)
  for each arriving task do
    Select an offloading action using the current LLDDPG policy
    Execute the selected action (local processing or MEC offloading)
    Observe the resulting latency and task completion outcome
    Compute the reward based on latency and deadline satisfaction
    Store the experience (state, action, reward, next state) in memory
    Update the LLDDPG model parameters using stored experiences
  end for
end for
Return the trained LLDDPG offloading policy

• Training procedure

The LLDDPG model was trained over multiple episodes using the task dataset described in Section 4.1.

We split the synthesized UAV task dataset into 20% for training and 80% for testing (seed = 42). To preserve distributional balance, we apply an approximate stratified split by file size, deadline, and priority. During training, the LLDDPG agent interacts with the environment over multiple episodes using mini-batches drawn exclusively from the 20% training subset. At each step, the agent selects offloading actions (local vs. MEC nodes) conditioned on the state vector (UAV energy, task size, deadline, MEC load, and link quality), receives rewards shaped to minimize end-to-end latency and deadline violations, and updates actor–critic parameters with target networks and prioritized experience replay. All model selection and hyperparameter tuning are performed on the training subset only; the 80% test subset remains strictly unseen until final evaluation and figure generation.

• Integration with MEC and NS

The MEC infrastructure in the system includes heterogeneous servers with support for NS. Each server hosts multiple virtual slices configured with different latency and resource guarantees. The LLDDPG agent implicitly learns to assign tasks to the most appropriate slice by associating observed performance feedback with the action taken, without explicit slice labeling.

This model is illustrated in Figure 5, which shows the interaction between UAVs, MEC nodes, and the LLDDPG controller operating in a closed-loop feedback system.

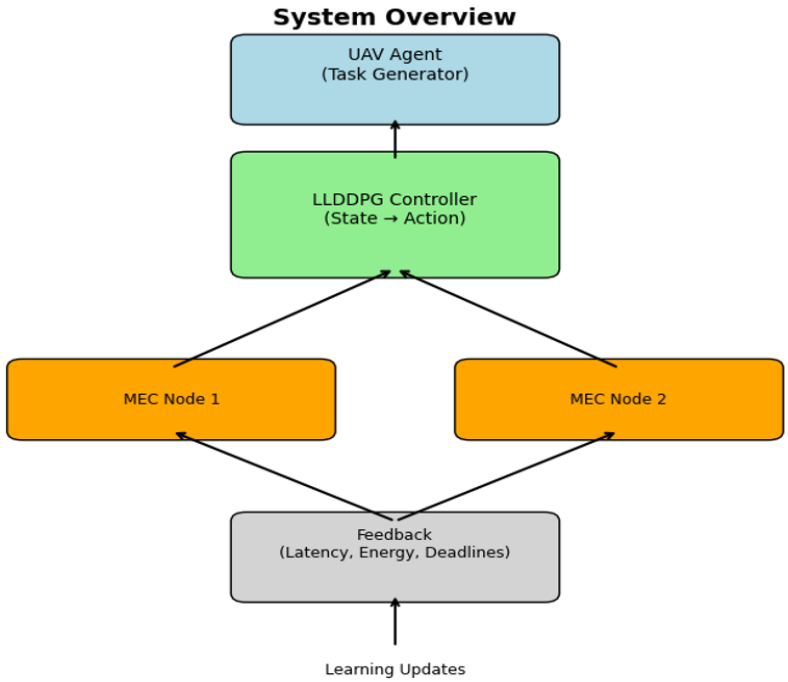

Figure 5. The interaction between UAVs, MEC nodes, and the LLDDPG

Note: UAVs: unmanned aerial vehicles; MEC: mobile edge computing; LLDDPG: latency-aware long-term deep deterministic policy gradient.

4.3 Simulation strategy

4.3.1 Simulation environment and parameter settings

The simulation environment is implemented in Python using task-level UAV image data. Each image is treated as an independent task with attributes including file size (MB), deadline, priority, and an estimated energy cost. Tasks are categorized into three slices based on size: S1 (≤ 1 MB), S2 (1-3 MB), and S3 (> 3 MB). MEC nodes are modeled as heterogeneous servers with predefined processing rates assigned per slice. Specifically, MEC processing rates are set to 5 MB/s (S1), 3 MB/s (S2), and 1 MB/s (S3). Local processing is modeled as slower execution with a rate of 0.5 MB/s. For each task, the processing delay is computed as size divided by the selected processing rate (local or MEC).

Deadline constraints are generated within the range 100-

1000 ms using a fixed random seed (42). Priority labels are generated as {High, Medium, Low} with probabilities {0.2, 0.5, 0.3}. The task energy estimate is modeled as proportional to task size using an energy factor of 0.5 (energy_est = 0.5 × size_MB). The learning agent observes a state vector that includes task size, normalized deadline, normalized energy estimate, UAV battery level, a normalized link-quality coefficient, and MEC load indicators. For reproducibility, the battery level is set to 0.7 and link quality is represented by a fixed normalized coefficient (link_q = 0.8) during simulation runs. MEC load indicators are initialized as (0.3, 0.5, 0.2).

The dataset split follows a fixed 20/80 train–test ratio with seed = 42. During training, a subset of up to 8000 tasks is sampled with random_state = 7, and the model is trained for 100 epochs. Performance is evaluated using end-to-end latency and deadline satisfaction rate under the same parameter settings. Table 3 presents the main simulation parameters used in the evaluation, including task slicing settings, MEC processing capabilities, wireless link assumptions, and training configuration. These parameters define the experimental environment and ensure the reproducibility of the simulation results.

Table 3. Simulation parameters and experimental settings used in the UAV–MEC offloading evaluation

Parameter	Description	Value
Task type	UAV image-based computation tasks	Image files
Number of tasks	Total tasks used in evaluation	22,925
Task slicing	Size-based task classification	S1 ≤ 1 MB, S2: 1-3 MB, S3 > 3 MB
Number of slices	Defined service slices	3
MEC nodes	Number of available MEC servers	3
MEC processing rate (S1)	Processing speed for Slice 1	5 MB/s
MEC processing rate (S2)	Processing speed for Slice 2	3 MB/s
MEC processing rate (S3)	Processing speed for Slice 3	1 MB/s
Local processing rate	UAV onboard computation speed	0.5 MB/s
Deadline range	Task deadline interval	100-1000 ms
Deadline generation seed	Random seed for deadline assignment	42
Task priority levels	Priority classes	High, Medium, Low
Priority distribution	Probability of priority levels	{0.2, 0.5, 0.3}
Energy model	Energy cost per task	0.5 × task size (MB)
UAV battery level	Initial normalized battery level	0.7
Link quality coefficient	Normalized wireless link quality	0.8 (fixed)
MEC load indicators	Initial normalized MEC loads	{0.3, 0.5, 0.2}
Train/test split	Dataset partitioning ratio	20% / 80%
Training sample size	Tasks used for training	8,000
Training random seed	Random state for training subset	7
Number of training epochs	RL training iterations	100
Simulation type	Evaluation method	Simulation-based

Note: MEC: mobile edge computing; UAV: unmanned aerial vehicle.

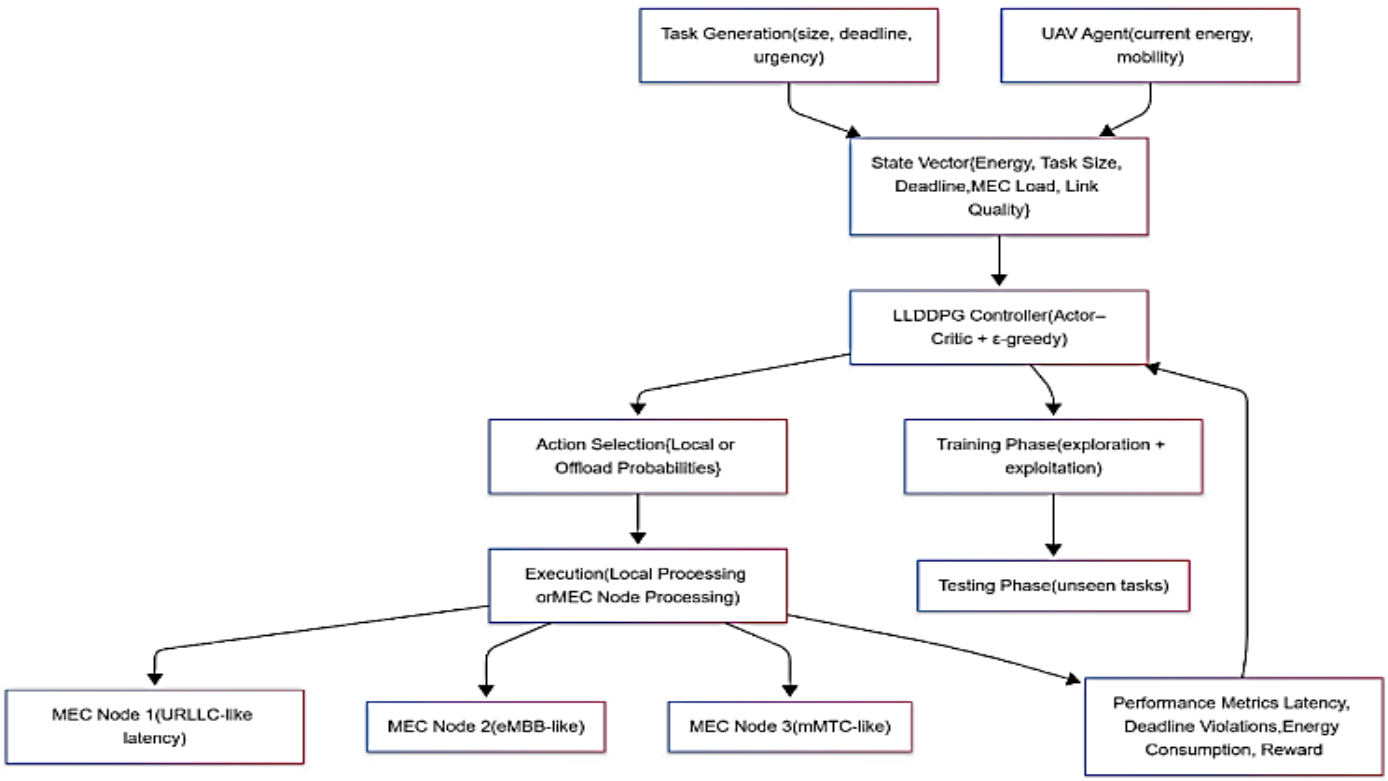

Figure 6. Simulation strategy overview

4.3.2 Simulation procedure

The performance evaluation was conducted using a Python-based simulation environment. The environment represents UAV-assisted MEC systems. UAV-generated tasks are modeled as independent computational units. Each task is described by file size, deadline, priority level, and estimated energy cost. MEC servers are configured with heterogeneous processing capacities. Wireless link conditions vary during simulation runtime. MEC nodes operate with non-uniform processing speeds. This configuration produces different processing delays and server load levels. Explicit NS is not implemented at the protocol level. At each decision step, the UAV observes the current system state. An offloading decision is selected using the trained LLDDPG policy. Available actions include local task execution and offloading to one MEC node. The system state is defined using a set of parameters. These parameters include remaining UAV energy, task size, task deadline, MEC server load, and wireless channel quality. During the evaluation phase, the learned policy is kept fixed. Task arrivals, execution outcomes, and system feedback are recorded throughout the simulation. Performance evaluation is based on quantitative metrics. These metrics include end-to-end latency, deadline satisfaction rate, energy consumption, and cumulative reward values.

- End-to-end latency
- Deadline violation rate
- Cumulative energy consumption
- Reward progression across episodes

The simulation is divided into two stages. These stages are training and testing. During training, the LLDDPG agent interacts with the environment. Policy parameters are updated during this stage. An ε-greedy strategy is applied. This strategy controls action selection during learning. After training, the learned policy is fixed. No further updates are applied. The fixed policy is evaluated using a separate set of unseen tasks. Evaluation is performed without modifying policy parameters. Performance is measured under changing operating conditions.

Figure 6 outlines the overall process. The aim of this setup is to observe how the LLDDPG agent responds when tasks appear at irregular times, when MEC nodes vary in load, and when QoS requirements shift. In essence, the simulation tests whether the system can continue functioning effectively even when the environment becomes inconsistent a common situation in practical UAV–MEC scenarios.

5. RESULTS AND DISCUSSION

5.1 Latency performance analysis

A total of 22,925 new UAV tasks were used to test how well the suggested LLDDPG-based offloading framework worked with the new 20/80 train–test split. The latency distribution for each job is shown in Figure 7. The 500 ms deadline is shown as a dashed line for reference. Our tests showed that most jobs were completed well below this time limit, with an average latency of 155.46 ms. This shows how quick the framework is and how it can keep up low-latency performance even when the training fraction is lowered.

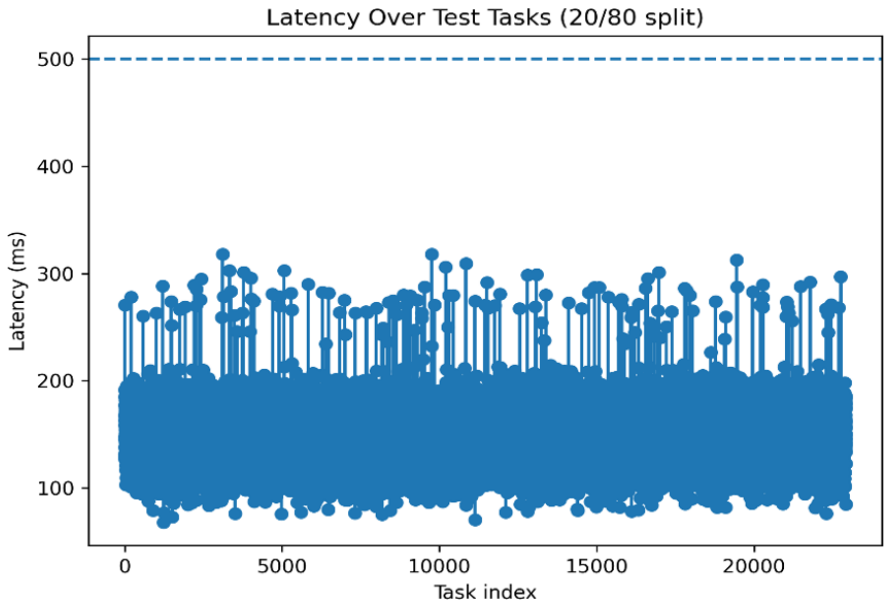

Figure 7. Latency over test tasks

Table 4. Latency-aware long-term deep deterministic policy gradient (LLDDPG) evaluation metrics over tasks

Metric	Value
Average reward	-169.03
Average latency	155.46 ms
Success rate	93.3%
Total tasks	22,925
Successful tasks	21,400
Failed tasks	1,525

The overall success rate for meeting deadlines was 93.3%, which showed that the agent was able to change to tasks it had not seen before while still being able to do so in real time. The success rate is slightly lower than it was in the beginning (100% success with smaller test sets), which is to be expected since the review is more complicated and the tests cover more ground. That being said, keeping delay well below the 500 ms limit shows how strong the model is. Table 4 shows a summary of the performance measures.

5.2 Reward and success analysis

The LLDDPG model was evaluated using 22,925 previously unseen offloading tasks. The recorded average reward value was -169.03. The mean end-to-end latency across all evaluated tasks was 155.46 ms. This value remained below the defined latency threshold of 500 ms.

Task completion statistics were recorded during the testing phase. A total of 21,400 tasks were completed within their deadline constraints. A total of 1,525 tasks exceeded the deadline. The deadline success rate was measured as 93.3%. Most recorded latency values remained below the threshold during evaluation.

Figure 8 shows the distribution of reward values and latency measurements for the test tasks. The reward plot reflects task-level outcomes under the learned LLDDPG policy, whereas the latency plot presents individual task delays relative to the 500 ms reference line. The concentration of latency values below this threshold indicates that most tasks satisfied the real-time processing requirement during evaluation.

The observed reward distribution reflects the trade-off learned by the LLDDPG agent between minimizing latency and avoiding deadline violations. Lower reward values are mainly associated with tasks belonging to larger slices (S2 and S3), which naturally incur higher processing delays. The variance observed in the reward curve indicates adaptive policy exploration during testing, rather than unstable learning, as the corresponding latency values remain consistently below the defined threshold for most tasks. This behavior suggests that the agent prioritizes urgent tasks while dynamically adjusting offloading decisions based on MEC load conditions.

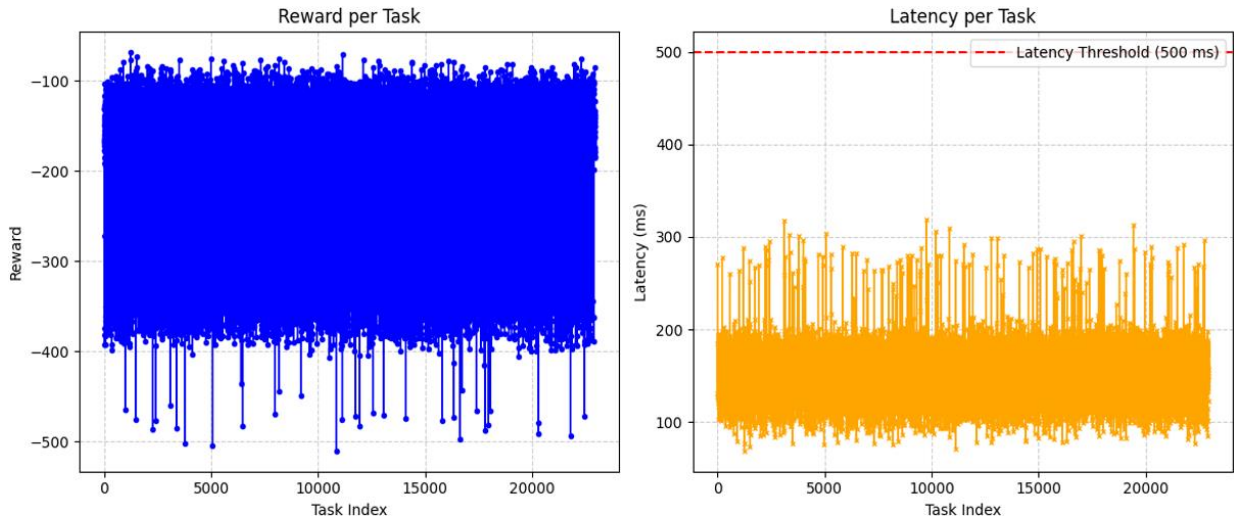

Figure 8. Per-task reward (left) and latency (right) for 100 unmanned aerial vehicle (UAV) delivery tasks
Note: The red dashed line denotes the 500 ms latency threshold

5.3 Comparative evaluation

A comparison was conducted using a static rule-based offloading strategy. The baseline uses predefined decision rules, and adaptation to network or server state changes is not applied. Under the same evaluation settings, the LLDDPG-based framework produced an average latency of 155.46 ms, with a recorded deadline success rate of 93.3%. The static baseline produced higher latency values, with a larger number of tasks exceeding the 500 ms latency threshold.

The performance improvement achieved by the LLDDPG-based framework is attributed to its ability to adapt offloading decisions to instantaneous system states. Unlike static rule-based strategies, the learned policy responds to variations in task urgency and MEC load, which leads to reduced congestion in heavily loaded slices, particularly Slice S3. By redistributing tasks dynamically, the agent avoids persistent overload conditions and achieves more balanced processing across available resources.

The objective of this comparison is to validate the effectiveness of adaptive learning-based offloading against commonly adopted static and heuristic baseline strategies in UAV–MEC systems. These baselines are used as reference configurations to highlight the limitations of fixed-rule decision policies under dynamic task arrivals and varying resource conditions. A direct experimental comparison with advanced reinforcement learning approaches, such as standard DDPG, PPO, or Lyapunov-based optimization, is beyond the scope of the current study and is identified as an important direction for future work.

Figures 9 and 10 show how the workload looked before and after using multi-MEC load balancing. The proposed framework spread the load more evenly across the slices. Slice S3, which was heavily loaded at first, saw a clear drop-in processing time after the tasks were redistributed.

Figure 11 compares average latency and success rates between the static offloading method and the proposed LLDDPG model. The proposed framework reduced average latency by more than 60% while increasing task success rates by over 20%.

Figure 12 shows a bar chart of the processing times for slices S1, S2, and S3 before and after load balancing. S1 and S2 increased a little because the tasks were redistributed, but Slice S3 which had the heaviest load dropped a lot. This confirms that even simple balancing helps.

Figure 13 shows the effect of LLDDPG more clearly. The processing times went down across all slices, and Slice 3 fell from more than 3000 seconds to under 1000 seconds. This shows that the model can significantly reduce task delays and keep the service running more smoothly.

The experimental evaluation in this study is conducted under a simplified simulation setting involving a single UAV and a limited number of MEC nodes. UAV mobility and wireless channel variations are abstracted to focus on task-

level offloading behavior. While this setup allows controlled analysis of latency-aware decision-making, it does not capture the full complexity of large-scale multi-UAV scenarios or highly dynamic network environments. Nevertheless, the observed performance gains indicate the potential effectiveness of the proposed LLDDPG framework under dynamic task arrivals, motivating further investigation in more complex deployments.

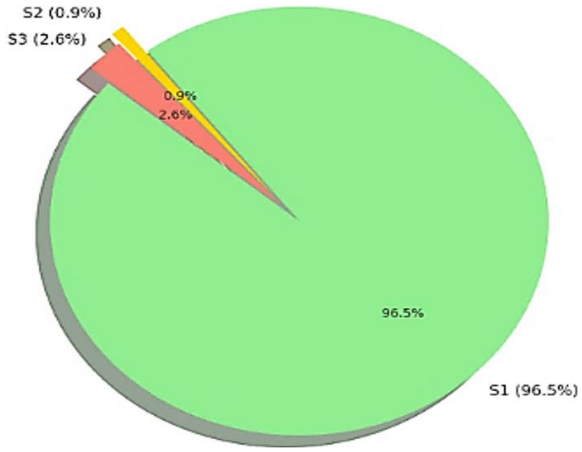

Figure 9. Workload distribution before applying multi-multi-mobile edge computing (MEC)load balancing

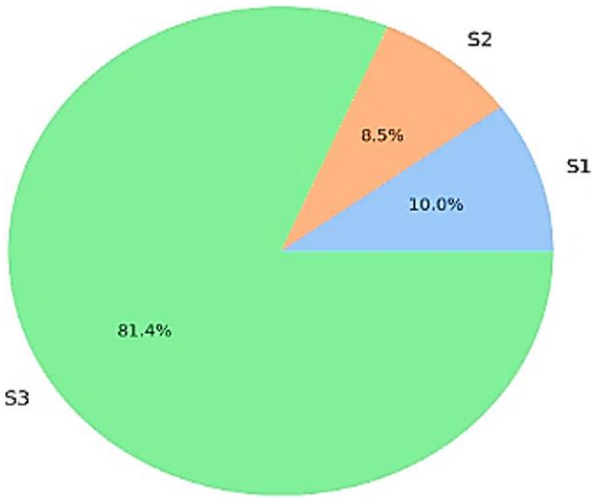

Figure 10. Workload distribution after applying multi-mobile edge computing (MEC) load balancing

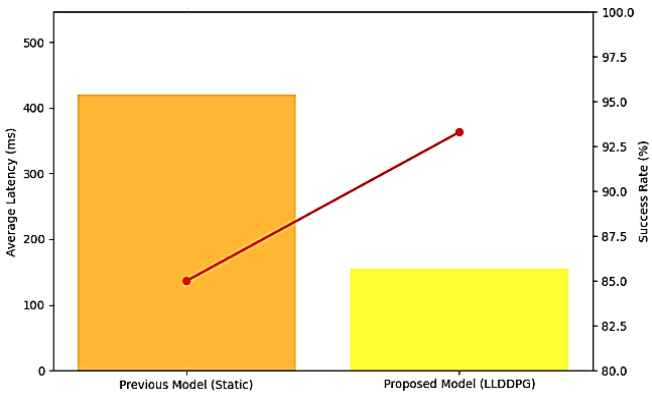

Figure 11. Compares average latency and success rates between the static method and long-term deep deterministic policy gradient (LLDDPG) model

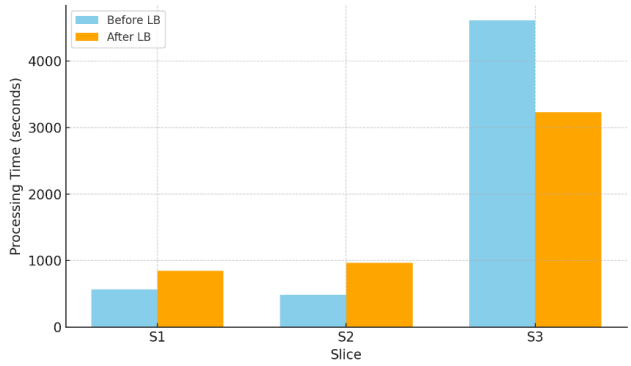

Figure 12. Processing times for slices before and after load balancing

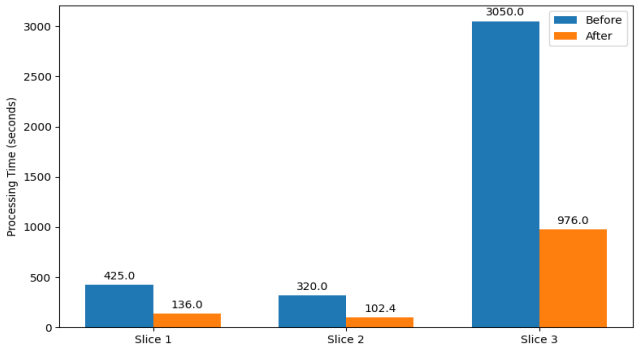

Figure 13. Processing time before and after optimization

5.4 Limitations and future enhancements

The evaluation was conducted in a simulation environment with discrete time steps. UAV mobility and wireless channel conditions are modeled using simplified assumptions, without explicit continuous trajectory variation or fast channel fluctuations. The experimental setup considers a single UAV and a limited number of MEC nodes with predefined processing capacities. Large-scale deployments, dense edge infrastructures, and multi-UAV coordination overhead are not addressed in this study. Evaluation beyond simulation, including hardware-based testing and real UAV deployment, is also not considered.

Future work will extend the current framework to multi-UAV scenarios with dense MEC deployments, higher mobility patterns, and explicitly modeled wireless channel dynamics to evaluate scalability and robustness under more complex operating conditions. In addition, comprehensive comparisons with state-of-the-art reinforcement learning and optimization-based offloading methods will be conducted to further assess the relative performance of the proposed LLDDPG framework.

6. CONCLUSION

A task-offloading framework for UAV-assisted networks was evaluated under a slicing-enabled MEC environment. NS is used to model heterogeneous task requirements at the baseline level, while a latency-aware LLDDPG-based decision policy is employed for adaptive task offloading. The evaluation used a test set of 22,925 tasks. The measured average end-to-end latency was approximately 155 ms. The deadline success rate was recorded as 93%. Performance was

compared against a static rule-based offloading strategy. Lower latency values were observed under the LLDDPG configuration. Slice-level processing times were recorded. Slice 1 processing time decreased from 425 s to 136 s. Slice 2 decreased from 320 s to 102 s. Slice 3 decreased from above 3000 s to below 1000 s. The study is limited to simulation-based evaluation. Results are reported under controlled conditions. Further validation in larger and more complex deployment scenarios is not included.

REFERENCES

[1] Zeng, Y., Wu, Q., Zhang, R. (2019). Accessing from the sky: A tutorial on UAV communications for 5G and beyond. Proceedings of the IEEE, 107(12): 2327-2375. https://doi.org/10.1109/JPROC.2019.2952892

[2] Jiang, F., Wang, K., Dong, L., Pan, C., Xu, W., Yang, K. (2020). AI driven heterogeneous MEC system with UAV assistance for dynamic environment: Challenges and solutions. IEEE Network, 35(1): 400-408. https://doi.org/10.1109/MNET.011.2000440

[3] Xia, X., Fattah, S.M.M., Babar, M.A. (2023). A survey on UAV-enabled edge computing: Resource management perspective. ACM Computing Surveys, 56(3): 1-36. https://doi.org/10.1145/3626566

[4] Bouzid, T., Chaib, N., Bensaad, M.L., Oubbati, O.S. (2023). 5G network slicing with unmanned aerial vehicles: Taxonomy, survey, and future directions. Transactions on Emerging Telecommunications Technologies, 34(3): e4721. https://doi.org/10.1002/ett.4721

[5] Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B. (2017). A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 19(4): 2322-2358. https://doi.org/10.1109/COMST.2017.2745201

[6] Pham, Q.V., Fang, F., Ha, V.N., Piran, M.J., et al. (2020). A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. IEEE Access, 8: 116974-117017. https://doi.org/10.1109/ACCESS.2020.3001277

[7] Rost, P., Mannweiler, C., Michalopoulos, D.S., Sartori, C., et al. (2017). Network slicing to enable scalability and flexibility in 5G mobile networks. IEEE Communications Magazine, 55(5): 72-79. https://doi.org/10.1109/MCOM.2017.1600933

[8] Skondras, E., Michailidis, E.T., Michalas, A., Vergados, D.J., Miridakis, N.I., Vergados, D.D. (2021). A network slicing framework for UAV-aided vehicular networks. Drones, 5(3): 70. https://doi.org/10.3390/drones5030070

[9] Colajanni, G., Sciacca, D. (2023). Multi-layer 5G network slicing with UAVs: An optimization model. Networks and Spatial Economics, 23(3): 755-769. https://doi.org/10.1007/s11067-023-09595-y

[10] Wadod, M.M., Mohammed, F.G. (2023). Tree crown detection using UAV-captured high-resolution aerial images for Baghdad university campus. Journal Port Science Research, 6(special): 67-80.

[11] Difar, H.F., Abed, F.M. (2022). Automatic extraction of unmanned aerial vehicles (UAV)-based cadastral map: Case study in AL-Shatrah District-Iraq. Iraqi Journal of Science, 63(2): 877-896.

[12] Wijethilaka, S., Liyanage, M. (2021). Survey on network slicing for Internet of Things realization in 5G networks. IEEE Communications Surveys & Tutorials, 23(2): 957-994. https://doi.org/10.1109/COMST.2021.3067807

[13] Yuan, Z., Muntean, G.M. (2020). AirSlice: A network slicing framework for UAV communications. IEEE Communications Magazine, 58(11): 62-68. https://doi.org/10.1109/MCOM.001.1900325

[14] Hurtado Sanchez, J.A., Casilimas, K., Caicedo Rendon, O.M. (2022). Deep reinforcement learning for resource management on network slicing: A survey. Sensors, 22(8): 3031. https://doi.org/10.3390/s22083031

[15] Huda, S.A., Moh, S. (2022). Survey on computation offloading in UAV-enabled mobile edge computing. Journal of Network and Computer Applications, 201: 103341. https://doi.org/10.1016/j.jnca.2022.103341

[16] Abdalla, A.S., Marojevic, V. (2021). Communications standards for unmanned aircraft systems: The 3GPP perspective and research drivers. IEEE Communications Standards Magazine, 5(1): 70-77. https://doi.org/10.1109/MCOMSTD.001.2000032

[17] Wang, Z., Zhao, W., Hu, P., Zhang, X., Liu, L., Fang, C., Sun, Y. (2024). UAV-assisted mobile edge computing: Dynamic trajectory design and resource allocation. Sensors, 24(12): 3948. https://doi.org/10.3390/s24123948

[18] Zeng, Y., Zhang, R., Lim, T.J. (2016). Wireless communications with unmanned aerial vehicles: Opportunities and challenges. IEEE Communications Magazine, 54(5): 36-42. https://doi.org/10.1109/MCOM.2016.7470933

[19] Abdulhameed, A.S. (2025). Enhance mobility management of FANETs based on hybrid deterministic and stochastic models (Doctoral dissertation, University of Baghdad). https://doi.org/10.13140/RG.2.2.29102.01605

[20] Al-Taie, A.I., Doos, Q.M. (2023). Material selection for unmanned aerial vehicles (UAVs) wings using Ashby indices integrated with grey relation analysis approach based on weighted entropy for ranking. Journal of Engineering, 29(7): 189-200. https://doi.org/10.31026/j.eng.2023.07.12

[21] Wang, Y., Fang, W., Ding, Y., Xiong, N. (2021). Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach. Wireless Networks, 27(4): 2991-3006. https://doi.org/10.1007/s11276-021-02632-z

[22] Hai, N.T. (2023). Latency-aware task offloading in UAV-assisted edge computing: Leveraging deep reinforcement learning. Ho Chi Minh City Open University Journal of Science-Engineering and Technology, 13(2): 69-79. https://doi.org/10.46223/HCMCOUJS.tech.en.13.2.2910.2023

[23] Liu, J., Zhang, X., Zhou, H., Lei, X., Li, H., Wang, X. (2025). Lyapunov-based deep deterministic policy gradient for energy-efficient task offloading in UAV-Assisted MEC. Drones, 9(9): 653. https://doi.org/10.3390/drones9090653

[24] Zhao, Z., Xu, Y., Yu, Q. (2025). DDPG-based computation offloading strategy for maritime UAV. Electronics, 14(17): 3376. https://doi.org/10.3390/electronics14173376

[25] Wang, Y., Zhao, L., Chu, X., Song, S., Deng, Y., Nallanathan, A., Liang, K. (2022). Deep reinforcement

learning-based optimization for end-to-end network slicing with control-and user-plane separation. IEEE Transactions on Vehicular Technology, 71(11): 12179-12194. https://doi.org/10.1109/TVT.2022.3191882

[26] Zakaryia, S.A., Mead, M.A., Nabil, T., Hussein, M.K. (2025). Task offloading for multi-UAV asset edge computing with deep reinforcement learning. Cluster Computing, 28(7): 462. https://doi.org/10.1007/s10586-025-05382-1

[27] Oubbati, O.S., Chaib, N., Lakas, A., Lorenz, P., Rachedi, A. (2019). UAV-assisted supporting services connectivity in urban VANETs. IEEE Transactions on Vehicular Technology, 68(4): 3944-3951. https://doi.org/10.1109/TVT.2019.2898477

[28] Mezaal, Y.S., Shareef, M.S., Alameri, B.M., Saeed, S. R., Al-Hilali, A.A., Hussein, Z.K., Al-Majdi, K. (2024). Emerging wireless communication technologies in Iraqi government: Exploring Cloud, edge, and Fog computing. Heritage and Sustainable Development, 6(1): 169-182. https://doi.org/10.37868/hsd.v6i1.493