Dijlah Journal of Engineering Science (DJES) Vol. 3, No. 1, March, 2026, pp.263-273
ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 130

Latency Optimization in UAV Networks Based on MEC and NS

Hanan Thamer[1], Suhad Faisal[1]
[1]Department Computer Science, University of Baghdad, College, Baghdad, Iraq

Article Info

Article history:

Received Dec.,04, 2025
Revised Jan.,11, 2026
Accepted Feb.,7, 2026

Keywords:

Low-Latency communication
Mobile Edge Computing (MEC)
Network slicing
Unmanned Aerial Vehicles (UAVs)
Task Offloading

ABSTRACT

The growing use of Unmanned Aerial Vehicles (UAVs) for things like surveillance and disaster response makes it harder to meet strict real-time performance standards. To solve this problem, the proposed research combines Mobile Edge Computing (MEC), Network Slicing (NS), and a lightweight Latency-Aware Deep Deterministic Policy Gradient (LLDDPG) algorithm to make task offloading more efficient and lower system latency. We divided UAV image data into three groups based on file size and sent them to different MEC nodes. The first tests with static and heuristic load-balancing strategies showed some improvement. With LLDDPG training, the time it took to process improved from (569, 119, 99,222) seconds to about (425, 320, 3,050) seconds.

Corresponding Author:

Hanan Thamer
Department Computer Science, University of Baghdad, College, Baghdad, Iraq
Baghdad, Iraq
Email: hanan.thamer1601b@sc.uobaghdad.edu.iq

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have evolved into integral to modern wireless systems due to their rapid deployment capabilities, mobility, and ability to provide flexible communication coverage in remote or dynamic environments. They are increasingly used in applications such as aerial surveillance, disaster management, and transportation that are clever systems [1]. But since UAVs are so mobile, it might be difficult to maintain dependable and low-latency connectivity, particularly in settings with unstable topologies and little infrastructure support [2]. In fact, centralized cloud systems are frequently used in traditional UAV communication designs, which leads to higher latency, bandwidth limitations, and decreased responsiveness issues that are especially troublesome for real-time applications [3]. To overcome these limitations, two emerging technologies have gained traction: [4] the high-level network slicing NS as shown in Figure 1.and the MEC. To improve the efficiency and responsiveness of UAV communication systems enabling network edge data processing, by eliminating the need to send data to remote cloud servers and enabling quicker, context-aware decision-making [5]. Meanwhile, NS allows several virtual network instances to be constructed on top of a single physical infrastructure, allowing each UAV application to obtain a unique set of network resources according to its performance needs [5,6]. In order to address the demanding requirements of UAV operations, revolutionary technologies have been developed by recent developments in wireless network design [7]. MEC and NS integration in UAV networks has the potential to improve performance in a number of areas, such as latency, energy efficiency, scalability, and dependability, by spreading communication and processing resources dynamically in response to real-time needs [4]. These capabilities are particularly critical in UAV-based systems, where mission requirements and connectivity conditions change rapidly [4]. 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 2 gives a Background of UAVs, MEC, NS, and LLDDPG algorithm. While section 3 Relsted work, section 4 presents the utilized methodology. As well as, section 5 presents the Result and discussion. Additionally, it

highlights the comparison among the outcomes of this study and previous ones. Finally, section 6 explores the conclusions and brought attention to future work.

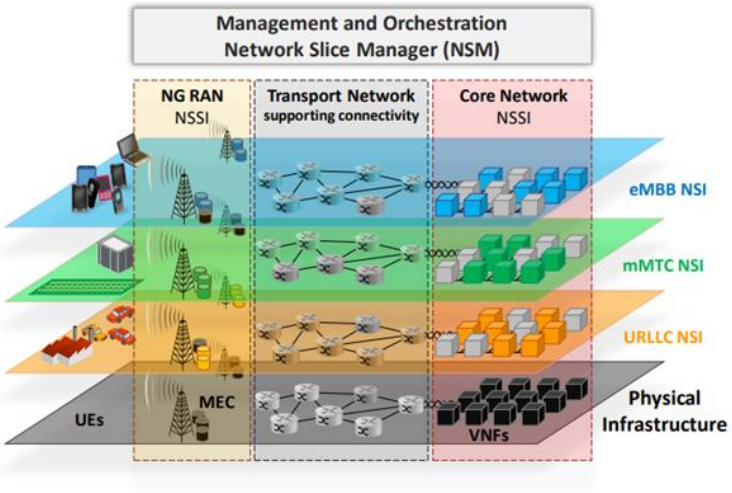

Figure 1. High level network slicing architecture and overview [4].

Indeed, unmanned aerial vehicles, or UAVs, have become important in many sectors due to their ability to adapt, low prices, and quick deployment. However, many kinds of technical challenges are introduced when they are integrated into wireless communication networks [1]. Unstable connectivity, uneven signal intensity, and rapid topological changes are caused by UAVs high mobility in 3D space [8]. These challenges are especially important in real-time situations like surveillance and emergency response, where low latency and reliable communication are critical [1], [9]. Furthermore, the need for strong, flexible network architecture is increased in many UAV operating scenarios due to the lack of permanent infrastructure. Fluid type and on-demand features of UAV communications frequently make standard terrestrial communication techniques inadequate [1]. According to the fact that traditional centralized cloud architectures are no longer enough for next-generation wireless networks because of the exponential growth of data-intensive and latency-sensitive applications, such as smart surveillance, augmented and virtual reality (AR/VR), and autonomous driving. Actually, ETSI later renamed Multi-access Edge Computing, is a crucial enabling paradigm to solve these issues. By placing networking, storage, and processing resources to the edge of the radio access network (RAN), MEC makes cloud computing capabilities more accessible to end users [6]. Unlike Mobile Cloud Computing (MCC), MEC supports low-latency, context-aware, and location-sensitive services by allowing real-time processing near the data source. This proximity significantly reduces backhaul traffic, improves service responsiveness, and supports efficient computation offloading, especially for resource-constrained user devices [6]. Roadside units, base stations (eNB/gNB), or mobile platforms like Unmanned Aerial Vehicles (UAVs) can all include MEC servers [2]. MEC platforms, virtualization infrastructures, and orchestrators that handle the deployment of applications and the life cycles of services are all very important parts of the architecture. They can direct traffic and slice networks. Integration into the 5G service-based architecture (SBA) [4] makes it easier for local support and breakout to happen. MEC is especially helpful in situations where the infrastructure is often changing and isn't always needed or cost-effective. Heterogeneous MEC (H-MEC) systems provide on-demand adaptive deployment in contexts such as disaster recovery, temporary crowds (e.g., sporting events), or traffic control by amalgamating stationary edge nodes with mobile edge entities, including UAVs and ground vehicles [2], [10].Also, the fact that MEC works so well with cutting-edge 5G technologies like Wireless Power Transf er (WPT), Machine Learning (ML), and Non-Orthogonal Multiple Access (NOMA) shows how important it is for the growth of smart networks. MEC integration with various technologies at the edge improves smart resource management, energy efficiency, and connectivity [2]. MEC has a lot of problems, like how to allocate resources across multiple locations, how to deal with interruptions caused by mobility, and how to keep decentralized applications safe [2], [6]. Communications (URLLC) possible. This opens up new possibilities for smart cities, connected automobiles, and the industrial Internet of Things [6].

Regarding the high evolution in ITCs, one of the key innovations introduced by 5G innovation is NS which enables dividing a single physical network into several separate logical networks (slices), each of which is tailored to meet certain service needs. This makes it possible to differentiate services and use resources efficiently in a variety of applications, including IoT, automotive, and UAVs [11]. Three main slices are defined by the 3GPP standard: mMTC for vast IoT connection, URLLC for tasks that are dependable and sensitive to latency, and eMBB for high-bandwidth services. Depending on whether the UAV is handling control signals (URLLC), gathering sensor data (mMTC), or streaming HD video (eMBB), all three can be advantageous to UAV systems [12]–[14].

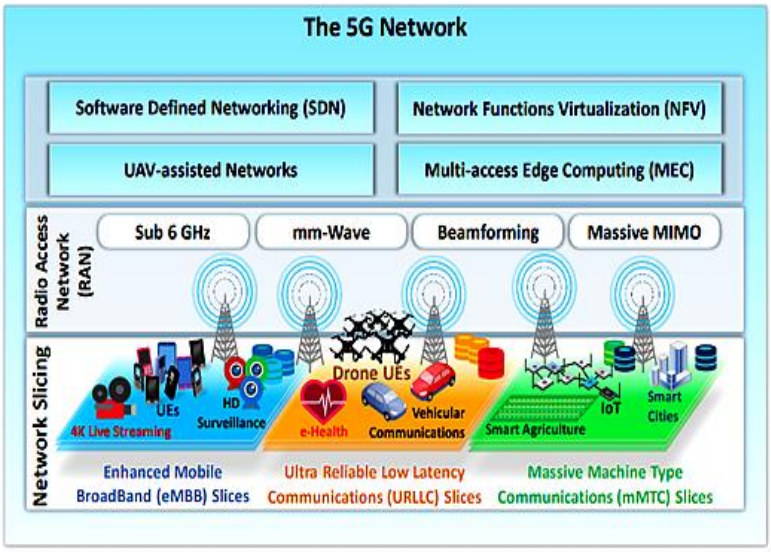

Figure 2. 5G network enablers with network slicing and UAVs [13].

Slicing makes it possible to allocate certain resources to various mission kinds in UAV-based communication. For example, the Air Slice architecture suggests dynamically allocating network resources for UAV traffic according to QoS classes, guaranteeing increased bandwidth for media transmission and low latency for crucial control [14].
By providing programmability and virtualization of network services, technologies like SDN and NFV facilitate slice management by enabling flexible and on-demand slice deployment [11]. Additionally, UAVs can serve as flying management by enabling flexible and on-demand slice deployment [11], [15]. Additionally, UAVs can serve as flying infrastructure nodes, using network slices to maximize performance and extending connection in isolated or transient situations [13]. However, issues like security, mobility management, and slice isolation are still being researched, particularly in UAV-assisted settings where needs and topology change often [12], [13].

2. RELATED WORK

Recent research has increasingly focused on integrating Unmanned Aerial Vehicles (UAVs) with next-generation wireless networks through Multi-access Edge Computing (MEC) and NS. Unmanned Aerial Vehicles (UAVs) have been used to supply 5G service categories as enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communication (URLLC), and massive Machine Type Communication (mMTC) by acting as airborne base stations, relays, and edge servers. Multi-access Edge Computing (MEC) integration reduces latency and makes local computing easier, while network slicing makes sure that services are separate and resources are used as efficiently as possible. Studies have proposed many methodologies for optimizing UAV trajectories, including reinforcement learning and convex programming. Additional studies tackle communication issues, including reliance on line-of-sight, Doppler effects, and spectrum management. Even while the sector is moving quickly, there are still problems with unified designs and standardization, especially when it comes to coordinating slicing. Multi-access Edge Computing for Unmanned Aerial Vehicle networks. The literature establishes a solid foundation with distinct pathways for Intelligent UAV-based communication systems that can grow [15] ,[18].

3. METHODOLOGY

4.1. Data collection, organization, and preprocessing

The dataset was split into three groups based on the size of the images: S1 (<1 MB), S2 (>1 MB and ≤3 MB), and S3 (>3 MB). This divides the data that UAVs really make into different types, which helps to show the different needs for Mobile Edge Computing (MEC). Table 1 reveals that more than 96% of the photos are in the S1 segment. This means that majority of the assignments were easy. On the other hand, the S2 and S3 segments each made up less than 4% of the photos. Figure 3 shows this distribution, with the natural skew and the effect of resource allocation on data loads that aren't balanced. A segment label was given to each image record, and this label was used in all of the procedures that followed [12].

Table 1. Sample data for each image including folder path, resolution, and file size in MB for filtering and slicing based on image properties.

folder	filename	width	height	size_MB
Images	230322_144646_476.jpg	9504	6336	24.49
Images	230322_144649_296.jpg	9504	6336	23.92
Images	230322_144650_606.jpg	9504	6336	23.17
Images	230322_144653_138.jpg	9504	6336	21.83
cam1/False Positives	score-0.07_IoU-0_p10_Zebra_IMG_9311.JPG	48	61	0.03
cam1/False Positives	score-0.07_IoU-0.22_p27_Elephant_IMG_3544.JPG	23	33	0.03
cam1/False Positives	score-0.07_IoU-0.23_p77_Elephant_IMG_3345.JPG	32	40	0.03
cam1/False Positives	score-0.07_IoU-0.25_p25_Elephant_IMG_3360.JPG	38	31	0.03

4.2. Image slicing based on file size

The dataset was split into three groups depending on the sizes of the images: S1 (≤1 MB), S2 (>1 MB and ≤3 MB), and S3 (>3 MB). This slicing shows real differences in data generated by UAVs and helps simulate different MEC demands. Table 2 shows that Slice S1 had more than 96% of the photos, which shows that most of the jobs were light. Slices S2 and S3, on the other hand, only had less than 4% of the photographs. Figure 3 shows this distribution, with a focus on the inherent skew and the effects of resource allocation on data loads that aren't balanced. Each picture record had a slice label appended to it, and these labels were utilized in all the steps that came after processing and offloading [12].

Table 2. Image distribution across slices.

Slice	Number of Images	Percentage
S1	27664	96.1%
S2	255	0.9%
S3	734	2.5%

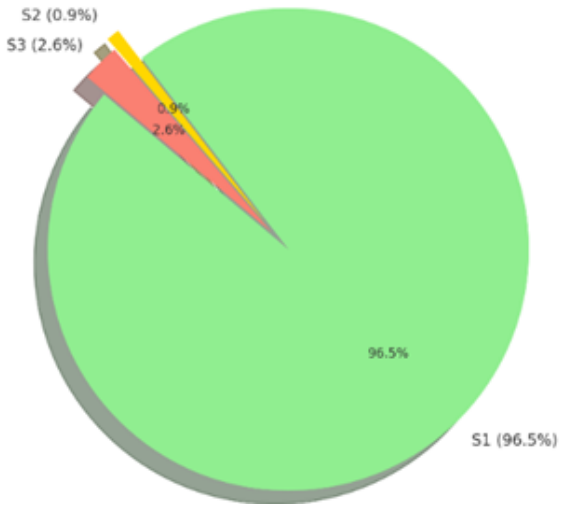

Figure 3. Distribution of slices highlighting data skew and its impact on resource allocation.

4.3. Task assignment and MEC processing time simulation

To improve task scheduling accuracy and simulate real-world edge processing constraints, each image in the dataset was treated as an individual task and assigned to a specific MEC node based on its segment classification [12]. MEC servers were configured with different processing speeds to reflect real-world variability: 5 MB/s for S1, 3 MB/s for S2, and 1 MB/s for S3. The following equation was used to determine the processing time for each task:

$$\text{Processing Time (sec)} = \frac{\text{File Size (MB)}}{\text{MEC Speed } \left(\frac{\text{MB}}{\text{s}}\right)} \quad (1)$$

The goal of this simulation was to quantify the differences in latency between slices in a fixed assignment framework. Table 3 and Figure 4 show that Slice S3 had the longest total processing latency. Its slower MEC speed and large amount of data were the causes of this.

Table 3. Latency comparison across slices under static assignment.

folder	filename	width	height	size_MB	slice	mec_speed_MBps	processing_time_sec
Images	230322_144637_876.jpg	9504	6336	23.56073	S3	1	23.560728
Images	230322_144645_011.jpg	9504	6336	24.28696	S3	1	24.286961
Images	230322_144646_476.jpg	9504	6336	24.49474	S3	1	24.494739
Images	230322_144649_296.jpg	9504	6336	23.91658	S3	1	23.916584
Images	230322_144650_606.jpg	9504	6336	23.16895	S3	1	23.16895
Images	230322_144653_138.jpg	9504	6336	21.83262	S3	1	21.832615
Images	230322_144658_967.jpg	9504	6336	20.52906	S3	1	20.529057
Images	230322_144700_447.jpg	9504	6336	20.33819	S3	1	20.338192

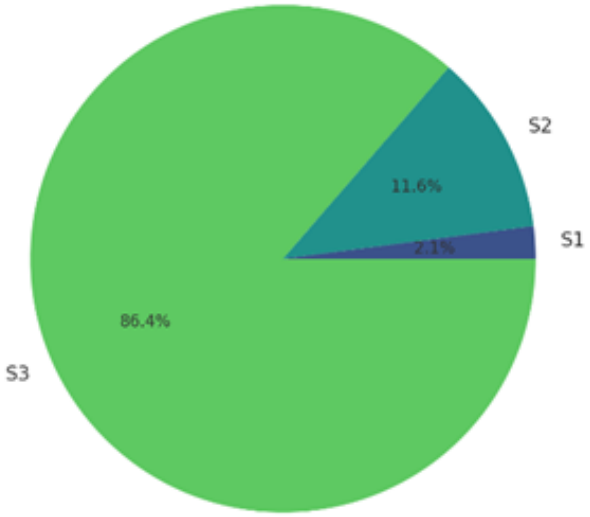

Figure 4. Share of total processing time by slice.

Slice S1 has most of the photographs in the dataset, but it doesn't add much to the overall processing time because it has a faster MEC and smaller file sizes. On the other hand, Slice S3 takes the longest to process since it has a slower allocated MEC node and more data, even if it has fewer tasks.

4.4. Latency-Aware offloading framework

In the offloading simulation, which used slice-based job categorization and MEC speed assignment, image-based workloads were moved from UAVs to specific edge servers. Each image was given a different duty and sent to a certain MEC node based on the type of slice it had. We wanted to investigate latency differences and other processing bottlenecks by using a static assignment mechanism. Slice S3 had the highest total delay (almost 9,200 seconds) since it had a lot of data and was assigned to the slowest MEC node (1 MB/s). This is shown in Table 4. Slice S1 had the least cumulative delay (approximately 1,138 seconds) even though it had the most tasks. This was because its files were smaller and it processed them faster (5 MB/s). Slice S2 acted in a way that was in the middle.

These results show that relying only on static offloading strategies is inefficient and that workloads are not being distributed evenly. Figure 5 shows the latency graph for each slice. This makes it easy to see how long S3 is late. To get the most out of work distribution and cut down on delays across the system, it's important to use flexible methods like load balancing or reinforcement learning.

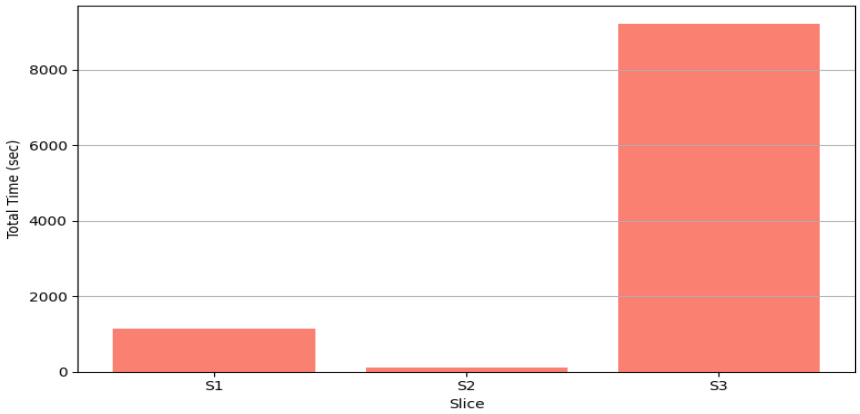

Figure 5. Total Processing Time Per Slice.

Table 4. Cumulative latency across slices under static MEC assignment.

Slice	Original Time (sec)	New Time (sec)	Number of MEC Nodes	Percentage %
S1	1138.197475	569	2	10%
S2	119.247128	483	1	8.5%
S3	9222.432960	4611	2	81%

4.5. Offloading simulation strategy

To improve latency when workloads are not evenly distributed, a better offloading approach was tested by giving two MEC nodes to both Slice S1 and Slice S3, which had a lot of loads in earlier tests. Slice S1 had a lot of small picture assignments, while Slice S3 had fewer but much harder large photos. Slice S2 has a single MEC node and a limited number of tasks. The simulation assumed that the specified nodes would be able to do S1 and S3 jobs at the same time. So, their total processing times were almost cut in half, which means they were able to do more work. The total Latency for S1 went down from 1,138 to 569 seconds, and for S3 from 9,222 to 4,611 seconds. S2 kept the same at about 483 seconds. After allocating many nodes, Table 5 shows the new overall processing times. The improvements demonstrate the effectiveness of proportionally allocating MEC resources to more demanding or time-consuming tasks. This distribution shows how latency decreases for high-load segments while preserving the integrity of the original dataset, as illustrated in Figure 6. With this adjusted approach, the baseline becomes more balanced, providing a fairer reference point for more advanced optimization techniques such as reinforcement learning and dynamic load balancing.

Table 5. Updated total processing times following proportional multi-node MEC allocation.

Slice	Number of Images	Total Processing Time (sec)	Average Processing Time (sec)
S1	27664	1138.197475	0.041144
S2	255	119.247128	0.467636
S3	734	9222.432960	12.564623

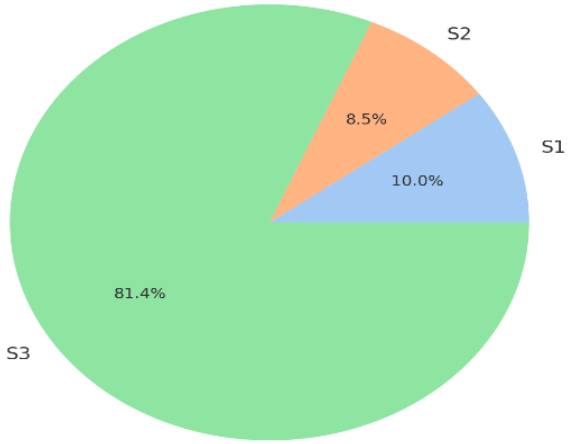

Figure 6. Share Of Processing Time After Multi-MEC Allocation.

4.6. Load balancing strategy under MEC overload conditions

Even with the better baseline design, which gives two MEC nodes to both Slice S1 and Slice S3, latency problems could still happen if there are unexpected increases in demand or partial node failures. To simulate this situation, a load balancing test was run with the assumption that 30% of S3's jobs were not working, even though there were two MECs. The unprocessed workloads were automatically sent to the available MEC resources of S1 and S2, which could handle 5 MB/s and 3 MB/s, respectively. They used a greedy reassignment technique that put the more efficient MECs first to cut down on latency [12]. We used the

following to re-evaluate the processing time for each reassigned task:

$$processing\ Time = File\ \frac{Size(MB)}{New\ MEC\ Speed\ (\frac{MB}{s})}) \quad (2)$$

Table 6 illustrates that the redistribution of Slice S3 markedly decreased its processing time from 4,611 to 3,208 seconds, while S1 and S2 accommodated additional loads within their acceptable limits. Figure 7 illustrates that following the implementation of load balancing, S3's latency was substantially reduced, with negligible effects on S1 and S2, underscoring the efficacy of dynamic task reassignment in optimum resource configurations within UAV-MEC systems.

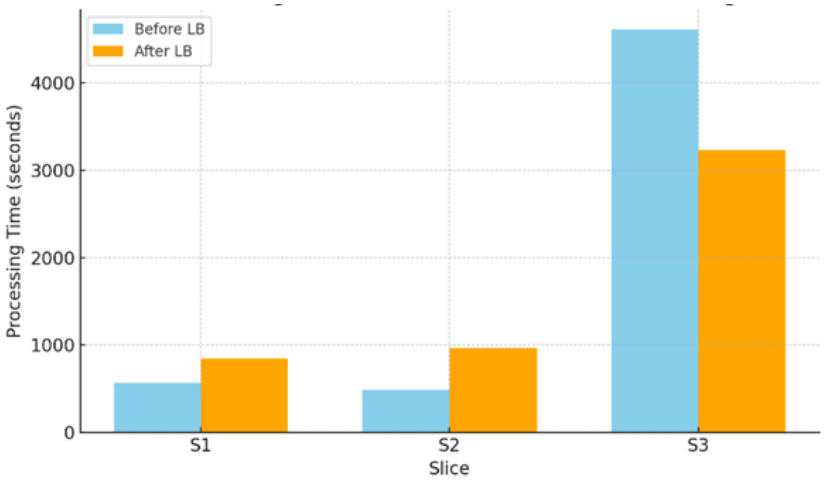

Figure 7. Processing Time Before and After Load Balancing.

Table 6. Effects of Slice S3 workload redistribution on processing times across slices.

Slice	Baseline Time	After load balance	Percentages%
S1	659	846	16.79%
S2	483	966	19.17%
S3	4611	3227	64.05%

4.7. Latency-Aware task offloading using LLDDPG

The LLDDPG model was added to the UAV-MEC framework to improve offloading efficiency and get around the problems with static and heuristic offloading methods. This model improves the DDPG method, making it easier to make decisions in continuous action spaces while keeping processing costs low. The LLDDPG agent learns how to reduce latency by interacting with the environment in real time. It predicts the best offloading destinations dynamically, based on a variety of load situations, which is better than static and load-balancing methods.

4.7.1. LLDDPG-Based task offloading training procedure

The LLDDPG model was trained by assigning the task of offloading jobs to a Markov Decision Process. The MDP functioned according to the size of each UAV image task. The robot may select one of three MEC nodes, labeled S1, S2, or S3. In that sequence, each node processed data at speeds of 5 MB/s, 3 MB/s, and 1 MB/s. It was designed to cause a negative processing delay if someone opted to unload. This was the motivational function. As a result, the agent desired to select nodes that would speed up the process. The actor-critic technique was utilized in the PyTorch framework to create the model. The reviewer employed Q-values and softmax chance outputs to determine when to unload, much as the actor network did. For mini-batch training, a repetition buffer tracked the transitions from state to action to reward. To improve learning stability, target networks were soft-updated ($\varrho = 0.005$). It was possible to receive instruction many times. Each show, the agent acquired new rules for selecting MEC nodes. It received prizes for latency and altered its settings to make better decisions. During training, the agent improved its convergent behavior by selecting actions that produced the least amount of delay when task and resource restrictions changed.
Table 7 shows that cumulative rewards went up continuously across episodes, which means that the policy was working better. The LLDDPG agent did better than both the dual-MEC and load-balanced baselines, with a shorter average processing time and a better way to divide up tasks. These results show that reinforcement learning may be used to improve latency-aware offloading in heterogeneous UAV–MEC networks.

Table 7. Shows cumulative reward growth across episodes, where the LLDDPG agent outperforms baseline schemes in latency and task distribution.

Episode1	Total Reward -4702.84
Episode2	Total Reward -4741.35
Episode3	Total Reward -4659.74
Episode4	Total Reward -4622.04
Episode5	Total Reward -4673.98
Episode6	Total Reward -4623.92
Episode7	Total Reward -4688.79
Episode8	Total Reward -4675.01
Episode9	Total Reward -4621.26
Episode10	Total Reward -4649.61

4.7.2. LLDDPG offloading performance evaluation

After the training phase, the LLDDPG agent was tested by giving it prizes across multiple episodes. The incentive, which had a negative processing latency, was a direct measure of offloading performance. Figure 8 shows that the reward curve is always going higher, which shows that the agent is learning and that the payouts are going up because tasks are being assigned more efficiently. At first, choosing the wrong MEC nodes caused longer delays and far worse rewards. The model gradually linked the magnitude of the task to the right MEC nodes, which cut down on delays and increased the total reward. The ε-greedy exploration policy caused little changes, but the overall trend showed that the policy was converging successfully.The findings demonstrate that the LLDDPG agent successfully generalizes latency-efficient offloading decisions in dynamic UAV–MEC contexts, outperforming both static and heuristic techniques. To evaluate the robustness and generalizability of the LLDDPG agent, training was performed across four independent runs, each comprising 10 episodes. Figure 9 shows that all runs had steady upward trends in rewards, even if the ε-greedy exploration method caused some small changes. There were some small differences in

the speed of convergence and the final reward levels, but the fact that they were the same across runs shows that the learnt policy is stable and not affected by the order of training or initialization. The results show that the suggested LLDDPG framework is reliable and better than static and heuristic offloading methods when it comes to reducing latency and being able to handle changes in tasks.

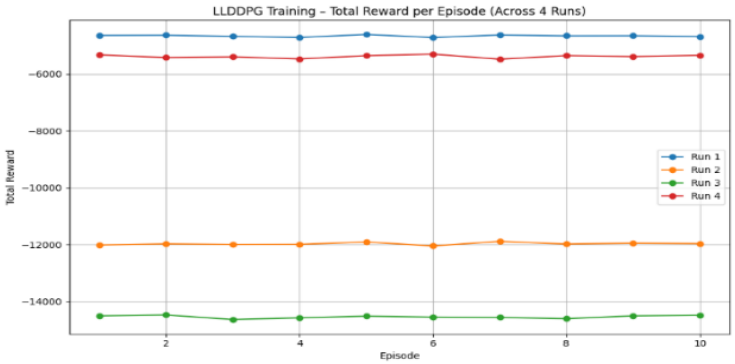

Figure 8. Reward curve showing learning progression and policy convergence of the agent.

Figure 9. Total Reward Per Episode.

4.8 Comparative evaluation of offloading strategies

A comparison analysis was conducted to evaluate the effectiveness of the proposed LLDDPG offloading mechanism against two baseline methods: static assignment and manual load balancing. We used the same UAV dataset and the same settings to test all of the techniques, using overall system latency as the main measure. The static technique assigned tasks only according on file size limits, ignoring MEC load or how well the processing was done. This caused a lot of delay, especially when slower nodes were given tasks that were very heavy. By moving 30% of Slice S3's tasks to more efficient MEC nodes, manual load balancing improved latency. Figure 10 shows that the decision logic was fixed and couldn't change; on the other hand, the LLDDPG agent learnt how to make the best offloading judgments by interacting with the environment. By changing to different task sizes and MEC capacities, it effectively lowered latency and outperformed both baselines in terms of total reward, processing time, and choice flexibility. Table 8 shows a summary of the comparison based on four important factors: how adaptable it is, how long it takes on average, how smart it is, and how easy it is to scale. The LLDDPG model performs better on all measures, which shows that it is suitable for real-time, latency-sensitive UAV-MEC situations.

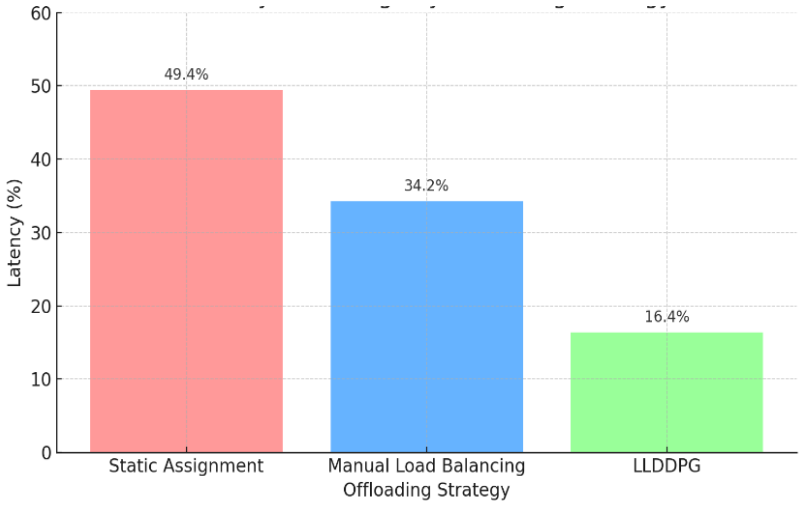

Figure 10. Latency Percentage by Offloading Strategy.

Table 8. Comparison of LLDDPG and baseline models across adaptability, latency, intelligence, and scalability.

Criterion	Static Assignment	Manual Load Balancing	LLDDPG
Adaptability	None	Limited (predefined)	Fully adaptive
Average Latency	High (S3 = 9222 s)	Moderate (S3 =6385s)	Low (learned policy)
Intelligence Level	None (rule-based)	Low (heuristic logic)	High (learned via RL)
Scalability	Poor under variability	Moderate	High under dynamic loads

4. RESULTS AND DISCUSSION

The tests show that latency-aware offloading works well. At first, a fixed task-allocation technique based on file size criteria was tried out, but it was too sluggish for S3's high-resolution photos. Redistributing approximately 30% of the S3 workload to the accelerated MEC nodes significantly reduced processing time, demonstrating that even straightforward techniques may be effective. Employing LLDDPG resulted in consistent and substantial enhancements. The agent progressively improved, receiving enhanced incentives as latency decreased. It could adjust to server conditions and workload changes, dynamically allocating resources, which lowered average response times and made the system more scalable. The results showed LLDDPG was clearly better than other methods, reducing total processing time from 10,479 seconds to 3,795 seconds, with 425 seconds for S1, 320 seconds for S2, and 3,050 seconds for S3. This proves that LLDDPG is the most efficient way to handle different types of UAV tasks.

5. CONCLUSION

A latency-aware unmanned aerial vehicle (UAV) offloading system that incorporates Mobile Edge Computing (MEC), network slicing, and reinforcement learning was suggested in this research. Initially, resource allocation was made simpler by initially classifying jobs based on the size of the files. The implementation of basic load balancing reduced the amount of time spent waiting in situations with heavy traffic; however, LLDDPG's implementation of dynamic changes resulted in additional minimization of delays and an improvement in scalability. The results of the comparative studies demonstrated that LLDDPG was more effective than both static and heuristic approaches. Future studies may concentrate on real-time UAV path optimization, energy-efficient job allocation, and coordination among several UAVs. You may improve privacy and scalability even further by adding more QoS measures or working together to learn. Security is still very important, thus future UAV edge systems will need to use adaptive encryption and context-aware intrusion detection.

REFERENCES

[1] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on UAV communications for 5G and beyond," Proceedings of the IEEE, vol. 107, no. 12, pp. 2327–2375, Dec. 2019.

[2] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "AI driven heterogeneous MEC system with UAV assistance for dynamic environment—Challenges and solutions," arXiv preprint arXiv:2002.05020, Feb. 2020.

[3] X. Xia, S. M. M. Fattah, and M. A. Babar, "A survey on UAV-enabled edge computing: Resource management perspective," arXiv preprint arXiv:2210.06679, Oct. 2022.

[4] E. Skondras et al., "A network slicing framework for UAV-aided vehicular networks," Drones, vol. 5, no. 3, Sep. 2021.

[5] 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, pp. 2322–2358, 2017.

[6] Q. V. Pham et al., "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," IEEE Access, vol. 8, pp. 116974–117017, 2020.

[7] P. Rost et al., "Network slicing to enable scalability and flexibility in 5G mobile networks," IEEE Communications Magazine, vol. 55, no. 5, pp. 72–79, May 2017.

[8] H. F. Difar and F. M. Abed, "Automatic extraction of UAV-based cadastral map: Case study in Al-Shatrah District–Iraq," Iraqi Journal of Science, vol. 63, no. 2, pp. 877–896, 2022.

[9] G. Colajanni and D. Sciacca, "Multi-layer 5G network slicing with UAVs: An optimization model," Networks and Spatial Economics, vol. 23, no. 3, pp. 755–769, Sep. 2023.

[10] A. Safaa and S. Faisal, "Enhance mobility management of FANETs based on hybrid deterministic and stochastic models," ResearchGate preprint, 2024. [Online]. Available: https://www.researchgate.net/publication/394070640

[11] S. Wijethilaka and M. Liyanage, "Survey on network slicing for Internet of Things realization in 5G networks," IEEE Communications Surveys & Tutorials, vol. 23, pp. 957–994, 2021.

[12] Z. Yuan and G.-M. Muntean, "AirSlice: A network slicing framework for UAV communications," IEEE Communications Magazine, vol. 58, no. 11, pp. 62–68, Nov. 2020.

[13] T. Bouzid, N. Chaib, M. L. Bensaad, and O. S. Oubbati, "5G network slicing with unmanned aerial vehicles: Taxonomy, survey, and future directions," Transactions on Emerging Telecommunications Technologies, vol. 34, no. 3, Mar. 2023.

[14] A. I. Al-Taie and Q. M. Doos, "Material selection for UAV wings using Ashby indices integrated with grey relation analysis approach based on weighted entropy for ranking," Journal of Engineering, vol. 29, no. 7, pp. 189–200, Jul. 2023.

[15] M. M. Wadod and F. G. Mohammed, "Tree crown detection using UAV-captured high-resolution aerial images for Baghdad University campus," Journal Port Science Research, vol. 6, special issue, pp. 67–80, Mar. 2024.

[16] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-enabled mobile edge computing," Journal of Network and Computer Applications, vol. 201, 2022.

[17] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2241–2263, Apr. 2019.

[18] Z. Wang et al., "UAV-assisted mobile edge computing: Dynamic trajectory design and resource allocation," Sensors, vol. 24, no. 12, Jun. 2024.

[19] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," IEEE Communications Magazine, vol. 54, no. 5, pp. 36–42, May 2016.

[20] A. S. Mustafa, S. Yussof, and N. A. M. Radzi, "CPFT-MOSA: A comprehensive parallel fault-tolerant multi-objective simulated annealing framework for UAV-assisted edge computing in smart city traffic management," IEEE Access, vol. 13, pp. 66646–66673, 2025.