IoT Data Analytics Using DRL

BTP Report

by Team B25AH02

Name	Roll No
Sushant Gadyal	**S20220010218**
Bathina Santosh Kiran	**S20220010035**
Dinesh Peddina	**S20220010062**

Guide Name: Dr. Abhishek Hazra

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

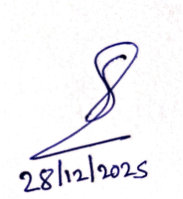
28th DEC, 2025

Final BTP Report

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**IoT Data Analytics Using DRL**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2025 to December 2025 under the supervision of Dr. Abhishek Hazra, Indian Institute of Information Technology, Sricity. The matter presented in this report has been submitted by me for the award of any other degree of this or any other institute.



Signature of the student with date

(Sushant Gadyal)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

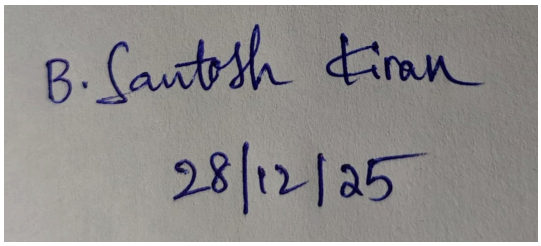
Signature of BTP Supervisor with date

(Dr. Abhishek Hazra)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**IoT Data Analytics Using DRL**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2025 to December 2025 under the supervision of Dr. Abhishek Hazra, Indian Institute of Information Technology, Sricity. The matter presented in this report has been submitted by me for the award of any other degree of this or any other institute.



Signature of the student with date
(Bathina Santosh Kiran)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

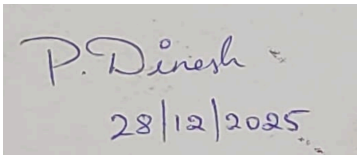
Signature of BTP Supervisor with date

(Dr. Abhishek Hazra)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**IoT Data Analytics Using DRL**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2025 to December 2025 under the supervision of Dr. Abhishek Hazra, Indian Institute of Information Technology, Sricity. The matter presented in this report has been submitted by me for the award of any other degree of this or any other institute.



Signature of the student with date

(Dinesh Peddina)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date

(Dr. Abhishek Hazra)

ABSTRACT

In recent years, Unmanned Aerial Vehicles (UAVs) have played a very crucial role in different domains like aerial photography, agriculture, defense, weather observation, environmental monitoring, and many more. They are being increasingly used as mobile edge computing nodes to process computation-intensive tasks.
They have played a very crucial role in the Internet of Things (IoT) by providing a Quality of Service (QoS).

Quality of Service mainly comes by reducing the latency and energy required for computing the tasks. So UAVs cannot handle all the tasks that are being offloaded to it and suffer from limited battery capacity and computational resources, making efficient task offloading a critical challenge.

This project proposes a Deep Reinforcement Learning (DRL) based task offloading framework, where a Master UAV dynamically decides whether to execute tasks locally or offload them to IoT devices, neighbouring UAVs, or cloud servers.

A custom OpenAI Gym environment is designed to model system dynamics such as CPU availability, bandwidth variation, task size, and battery consumption. A Deep Q-Network (DQN) Agent learns optimal offloading decisions by minimizing a combined cost of execution delay and energy consumption. Experimental results demonstrate that the proposed approach enables adaptive and energy-efficient task offloading, improving overall system performance.

Contents

1 Introduction

The integration of UAVs with Internet Of Things (IoT) technology has enhanced their use in various areas, providing superior solutions where traditional approaches are infeasible or inadequate. This collaboration combines UAV aerial capabilities with IoT data insights to address challenges in fields like agriculture and environmental monitoring. UAVs are resource-constrained devices, i.e. although they are capable of performing various kinds of tasks, they have limited energy and computational resources.

Task offloading in UAV-IoT networks is challenging because system conditions change continuously. Task sizes vary, available CPU resources fluctuate, wireless bandwidth changes, and UAV battery levels decrease over time. Poor offloading decisions can result in high energy consumption and large processing delays, which reduce the overall performance of the system.

In this project, a DRL-based task offloading framework is proposed for UAV-IoT networks. A Master UAV learns to decide whether a task should be executed locally, offloaded to IoT devices, sent to neighboring UAVs, or processed in the cloud. The decision is made by considering factors such as task size, computational requirements, available CPU resources, bandwidth, and UAV battery level. The objective is to minimize both energy consumption and task execution delay.

2 Related Works

Deep Reinforcement Learning (DRL) has shown strong potential in solving resource management problems in dynamic environments. Previous works have applied DRL to optimize energy and latency in IoT and Industrial IoT systems, demonstrating improved adaptability compared to traditional methods. DRL-based approaches have also been extended to UAV-assisted networks, where learning-based agents dynamically allocate computational and communication resources to improve service performance.

Despite these advancements, many existing approaches do not consider task scheduling dependencies or ignore important network parameters such as dynamic bandwidth variation. In addition, task partitioning and partial task execution across multiple UAVs are often not addressed. To overcome these limitations, this work builds upon existing DRL-based frameworks by incorporating task scheduling, bandwidth awareness, and adaptive task offloading, aiming to further reduce energy consumption and execution delay in UAV-IoT networks.

- A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, “Collaborative ai-enabled intelligent partial service provisioning in green industrial fog networks,” *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2913–2921, 2021.

- A. Hazra, M. Adhikari, T. Amgoth, and S. Srirama, “Intelligent Service Deployment Policy for Next-Generation Industrial Edge Networks,” *IEEE Transactions on Network Science and Engineering,* vol. 9, no. 5, pp. 3057–3066, 2022.

- S. Varasala, V. M. R. Tummala, S. N. Reddy, S. K. Talada, A. Hazra, and M. Gurusamy, “Deep reinforcement learning for task partitioning and partial offloading in UAV networks,” in *Proc. IEEE 100th Vehicular Technology Conference (VTC-Fall)*, 2024.

3 Problem Statement & Contribution

3.1 Problem Statement

UAV-enabled IoT networks involve dynamic task generation under strict constraints of UAV battery life, computational capacity, and communication bandwidth. Processing tasks locally at the UAV may lead to excessive energy consumption and delay, while offloading tasks to IoT devices, neighboring UAVs, or cloud servers introduces additional transmission overhead and latency. Existing static and heuristic-based offloading strategies fail to adapt efficiently to such dynamic environments.

Therefore, there is a need for an intelligent decision-making framework that can dynamically determine optimal task execution locations in UAV-IoT networks, with the objective of minimizing overall energy consumption and task execution delay.

3.2 Group Members' Contributions

The whole Deep-Reinforcement Learning based agent was developed together as a team, with each member contributing to specific features of the system.
The contributions of each member are detailed below.

3.2.1 Sushant Gadyal

Sushant played a key role in **designing and implementing a Deep Reinforcement Learning-based task offloading framework** for UAV-enabled IoT networks.

His contributions includes:

- Developed a realistic **simulation environment** that emulates dynamic task arrivals, fluctuating computational capabilities, varying bandwidth availability, and UAV battery limitations to support comprehensive performance evaluation.

- Proposed and realized a learning-based task offloading framework for UAV-assisted IoT systems, where a **Master UAV** autonomously selects the most suitable execution option among local processing, IoT nodes, peer UAVs, and cloud servers under dynamic conditions.

- Implemented a **DQN-based decision model** for task offloading, with well - defined state, action, and reward representations to jointly optimize energy consumption and execution delay.

- Developed an **enhanced reward formulation** that better captures system objectives, leading to improved training stability and faster convergence of the learning agent.

- Integrated a **micro-deadline feasibility** check to filter out time-constraint violations, thereby improving the reliability and robustness of task execution decisions.

3.2.2 Bathina Santosh Kiran

Santosh Kiran played a pivotal role in improving the task offloading decisions of the DRL- based framework by developing and integrating a **pre-execution classification** module.
His contributions include:

- Developed a module that evaluates energy and time constraints to determine whether tasks can be executed **locally before invoking** the DRL-based offloading strategy.

- Created a robust **feasibility check mechanism** to eliminate infeasible local execution options, thereby minimizing unnecessary exploration and enhancing decision-making efficiency.

- Integrated the classification module with the DRL framework, establishing a hierarchical decision-making process that improved learning stability and overall system performance.

3.2.3 Dinesh Peddina

Dinesh played a key role in enhancing the **evaluation and analysis** of the DRL-based task offloading framework by creating a dedicated visualization component.
His contributions include:

- Implemented **plots to track the DRL agent's performance** during both training and testing phases, highlighting learning progression and convergence patterns.
- Constructed **side-by-side comparisons of training and testing** rewards to quickly assess the agent's reliability and performance consistency.

The combined efforts of the team enabled the creation of a robust, efficient, and DRL agent. Each member's specialized skills were instrumental in overcoming the project's technical challenges and successfully meeting its goals.

4 Proposed Methodology

The methodology for developing the Deep Reinforcement Learning(DRL) based agent includes the following component:

4.1 System Architecture and Network Model

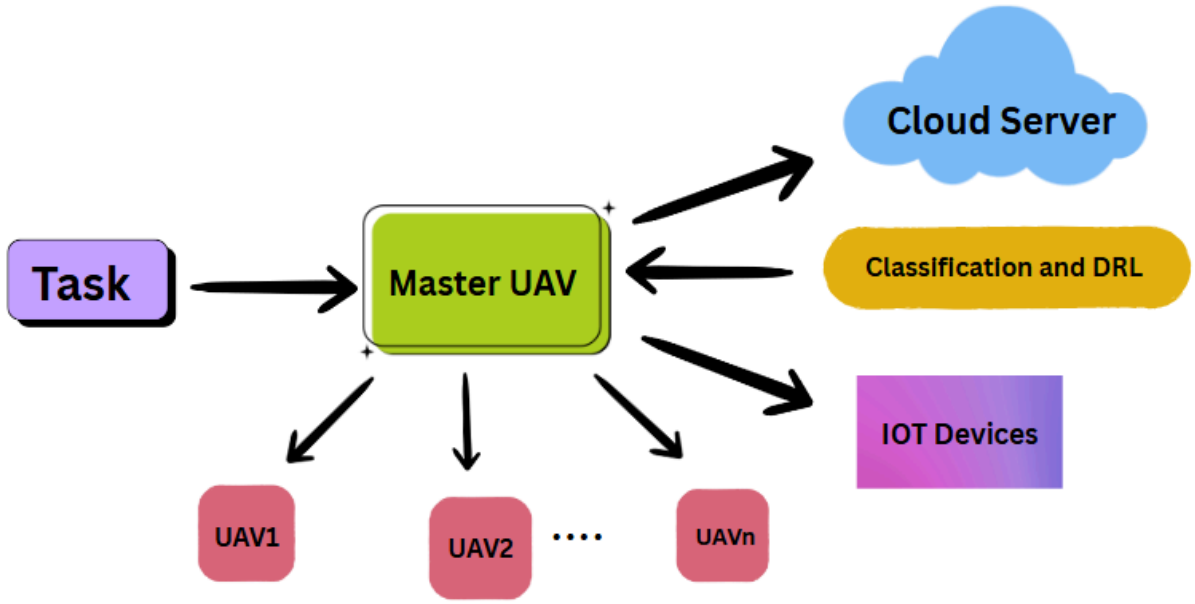


Figure 1: Diagram of the Network Model

4.1.1 System Components

- The system comprises a **Master UAV**, multiple **IoT devices**, **auxiliary UAVs**, and **cloud servers.**
- The Master UAV functions as the central decision-making unit for task execution and offloading.

- IoT devices and auxiliary UAVs provide limited computational support, while cloud servers offer high processing capability with higher communication latency.

4.1.2 Task Execution Flow

- Computational tasks are dynamically generated and assigned to the Master UAV.
- Each task is defined by parameters such as **task size** and **computational requirement**, which influence execution cost.
- Based on available computational resources, battery level, and network conditions, tasks are either executed locally on the Master UAV or offloaded to IoT devices, auxiliary UAVs, or cloud servers.

4.1.3 Practical Relevance

- The architecture closely models real-world **UAV-enabled IoT environments** where centralized decision-making is necessary due to limited onboard resources.
- The presence of heterogeneous computing nodes enables adaptive task allocation while balancing **latency, energy consumption,** and **computational efficiency.**

4.1.4 Tools and Technologies Used

- The system environment is implemented using **Python** and modeled as a reinforcement learning environment using the **OpenAI Gym** framework.
- **NumPy** is used for numerical computations and dynamic resource modeling.

- The DRL agent is implemented using **Tensorflow (Keras)** employing a Deep Q-Network(DQN) architecture for learning optimal task offloading decisions.
- **Matplotlib** is used for visualizing training and testing performance to analyze learning behavior and convergence.

4. 2 Task Modeling and Feasibility Analysis

- Each computational task is analytically modeled in terms of **execution delay** and **energy consumption** for different execution modes.
- These analytical models provide the foundation for evaluating whether a task can be feasibly executed at a specific node under current system conditions.

4.2.1 Local Task Execution

- For local execution, the computation delay is determined by the task's CPU cycle requirement and the processing capability of the Master UAV.
- The corresponding energy consumption depends on the UAV's computational power and operating frequency.
- Local execution is well-suited for **latency-sensitive tasks** but is constrained by the limited battery capacity of the UAV.

$$\mathbb{T}_{i,m}^{\text{Master UAV}} = \frac{\Gamma_{i,m}\mathcal{O}_i\mathcal{D}_i}{\mathscr{C}_m^{\text{CPU}}}$$

$$\mathbb{E}_{i,m}^{\text{Master UAV}} = \Gamma_{i,m}\mathcal{O}_i\mathcal{D}_i k \left(\mathscr{C}_m^{\text{CPU}}\right)^2$$

4.2.2 Offloading to IoT devices

- When a task is offloaded to an IoT device, the total execution cost includes:
    - Energy consumption generated by transmitting the task from Master UAV to IoT device.
    - Computation delay occurred by transmitting the task from Master UAV to IoT device.
    - Energy consumption generated by processing the task in the IoT device.
    - Computational delay occurred by processing the task in the IoT device.
- Due to limited computational resources, IoT devices are primarily suitable for **lightweight tasks.**

$$\mathbb{T}_{m,d}^{\text{upload}} = \frac{\Gamma_{i,d}\mathcal{O}_i}{\mathbb{R}_{m,d}} \qquad \mathbb{T}_{i,d}^{\text{IoT}} = \frac{\Gamma_{i,d}\mathcal{O}_i\mathcal{D}_i}{\mathscr{C}_d^{\text{CPU}}}$$

$$\mathbb{E}_{m,d}^{\text{upload}} = \frac{\Gamma_{i,d}\mathcal{O}_i\mathscr{P}_m}{\mathbb{R}_{m,d}} \qquad \mathbb{E}_{i,d}^{\text{IoT}} = \Gamma_{i,d}\mathcal{O}_i\mathcal{D}_i k\left(\mathscr{C}_d^{\text{CPU}}\right)^2$$

$$\mathbb{T}_i^{\text{IoT}} = \mathbb{T}_{m,u}^{\text{upload}} + \mathbb{T}_{i,u}^{\text{IoT}}$$

$$\mathbb{E}_i^{\text{IoT}} = \mathbb{E}_{m,d}^{\text{upload}} + \mathbb{E}_{i,d}^{IoT}$$

4.2.3 Offloading to Auxiliary UAVs

- Auxiliary UAV offloading considers both transmission and processing overheads.
- Compared to IoT devices, auxiliary UAVs offer higher processing capability and reduced execution delay.
- This option provides a balanced trade-off and is suitable for **moderately complex tasks.**

$$\mathbb{T}_{m,u}^{\text{upload}} = \frac{\Gamma_{i,u}\mathcal{O}_i}{\mathbb{R}_{m,u}} \qquad \mathbb{T}_{i,u}^{\text{UAV}} = \frac{\Gamma_{i,u}\mathcal{O}_i\mathcal{D}_i}{\mathscr{C}_u^{\text{CPU}}} \qquad \mathbb{T}_i^{UAV} = \mathbb{T}_{m,u}^{\text{upload}} + \mathbb{T}_{i,u}^{\text{UAV}}$$

$$\mathbb{E}_{m,u}^{\text{upload}} = \frac{\Gamma_{i,u}\mathcal{O}_i\mathscr{P}_m}{\mathbb{R}_{m,u}} \qquad \mathbb{E}_{i,u}^{\text{UAV}} = \Gamma_{i,u}\mathcal{O}_i\mathcal{D}_i k\left(\mathscr{C}_u^{\text{CPU}}\right)^2 \qquad \mathbb{E}_i^{UAV} = \mathbb{E}_{m,u}^{\text{upload}} + \mathbb{E}_{i,u}^{\text{UAV}}$$

4.2.4 Offloading to Cloud Servers

- Cloud offloading involves higher communication delay due to long-distance data transmission.
- However, the high computational power of cloud servers results in minimal processing delay.
- This execution mode is appropriate for **computation-intensive** but **delay-tolerant tasks.**

$$\mathbb{T}_{m,c}^{\text{upload}} = \frac{\Gamma_{i,c}\mathcal{O}_i}{\mathbb{R}_{m,c}} \qquad \mathbb{T}_{i,c}^{\text{Cloud}} = \frac{\Gamma_{i,c}\mathcal{O}_i\mathcal{D}_i}{\mathscr{C}_c^{\text{CPU}}} \qquad \mathbb{T}_i^{Cloud} = \mathbb{T}_{m,c}^{\text{upload}} + \mathbb{T}_{i,c}^{\text{Cloud}}$$

$$\mathbb{E}_{m,c}^{\text{upload}} = \frac{\Gamma_{i,c}\mathcal{O}_i\mathscr{P}_m}{\mathbb{R}_{m,c}} \qquad \mathbb{E}_{i,c}^{\text{Cloud}} = \Gamma_{i,c}\mathcal{O}_i\mathcal{D}_i k\left(\mathscr{C}_c^{\text{CPU}}\right)^2 \qquad \mathbb{E}_i^{Cloud} = \mathbb{E}_{m,c}^{\text{upload}} + \mathbb{E}_{i,c}^{\text{Cloud}}$$

4.2.5 Feasibility Validation and Pre-Decision Classification

- Before invoking the DRL agent, a **feasibility validation mechanism** evaluates whether local execution satisfies predefined energy and delay constraints.
- If local execution is found to be infeasible, it is excluded from the action space.
- This pre-decision classification reduces unnecessary exploration and improves learning efficiency by preventing invalid or impractical decision choices.

4.3 DRL-based Task Offloading and Decision Framework

4.3.1 MDP Formulation

- The task offloading problem is modeled as a **Markov Decision Process (MDP)** to capture the sequential and dynamic nature of decision-making.
- The **state space** represents the current system status, including:
    - Task characteristics (data size and computation requirement)
    - Remaining energy of the Master UAV.
    - Available computational resources of IoT devices, auxiliary UAVs, and cloud servers.
    - Network conditions such as available bandwidth.
- The **action space** consists of multiple execution choices:
    - Local execution at the Master UAV,
    - Offloading to IoT devices.
    - Offloading to auxiliary UAVs.
    - Offloading to cloud servers.

4.3.2 Deep Q-Network (DQN) Architecture

- A **Deep Q-Network (DQN)** is employed to learn the optimal offloading policy from system interactions.

- The DQN takes the current system state as input and outputs **Q-values for all possible actions.**

- A multi-layer neural network is used to approximate the Q-function, enabling the agent to handle high-dimensional state spaces effectively.

4.3.3 Training Strategy and Exploration Policy

- An **experience replay mechanism** is adopted to store past interactions and randomly sample them during training.

- This approach reduces correlation between consecutive samples and improves training stability.

- An **epsilon-greedy exploration strategy** is used to balance exploration and exploitation:

    - Initially, the agent explores more actions,

    - Over time, exploration is reduced to favor learned optimal decisions.

4.3.4 Enhanced Reward Shaping Mechanism

- To guide the learning process toward practical and efficient decisions, an **enhanced reward shaping mechanism** is introduced.
- The reward function jointly considers:
    - Execution delay.
    - Energy consumption
    - Successful and timely task completion.
- Additional incentives are provided for:
    - Faster task execution.
    - Energy- efficient task handling.
    - Valid execution decisions.
- This reward design accelerates convergence and promotes stable and energy-aware policies.

4.3.5 Micor-deadline Evaluation and Task Prioritization

- A **micro-deadline evaluation mechanism** is incorporated to identify time-critical tasks.

- Tasks with estimated execution times close to predefined thresholds are marked as urgent.
- Urgent tasks are given higher priority by adjusting the reward signal, encouraging prompt execution.
- This mechanism improves **Quality of Service (QoS)** by ensuring timely processing of latency-sensitive applications.

4.4 Dynamic Resource Modeling and Performance Evaluation

- The system is modeled with dynamic variations in **bandwidth, computational resources, and energy levels** to reflect realistic operating conditions.
- The DRL agent is trained over multiple episodes using environment interactions, while testing is performed with exploration disabled to evaluate policy stability.
- Performance is evaluated using **cumulative reward trends** to analyze learning convergence.
- **Training and testing rewards a**re compared to check how well the learned policy performs in different scenarios.

5 Experimental Results

To evaluate the effectiveness of the proposed enhancements, experiments were conducted in multiple phases. Each phase incrementally introduces additional components to the baseline DRL framework.

In all experiments, performance is evaluated using cumulative reward, where the reward is defined as the negative sum of execution delay and energy consumption. Therefore, **higher reward values (i.e values closer to zero)** indicate better system performance.

The **x-axis** represents training episodes, while the **y-axis** denotes the total reward accumulated per episode.

5.1 Training Phase vs Testing Phase

5.1.1 Training Phase:

- During training, the agent learns the optimal task offloading policy through interaction with the environment. Initially, reward values fluctuate due to **exploration**. As training progresses, the reward curves become smoother and less negative, indicating improved decision-making and policy convergence.

5.1.2 Testing Phase:

- During testing, **exploration** is disabled and the agent follows the learned policy. The testing reward curves exhibit stable behavior with reduced variance, demonstrating good generalization and consistent performance.

5.2 Phase 1: Baseline DRL Framework

In the initial phase, a basic DRL -based offloading strategy was implemented without feasibility checks or rewards enhancements.

The training reward curve shows large fluctuations and remains highly negative, indicating inefficient exploration and frequent suboptimal decisions. The corresponding testing results also exhibit instability, reflecting poor generalization of the learned policy.

Figure 2: Training Results in Phase 1

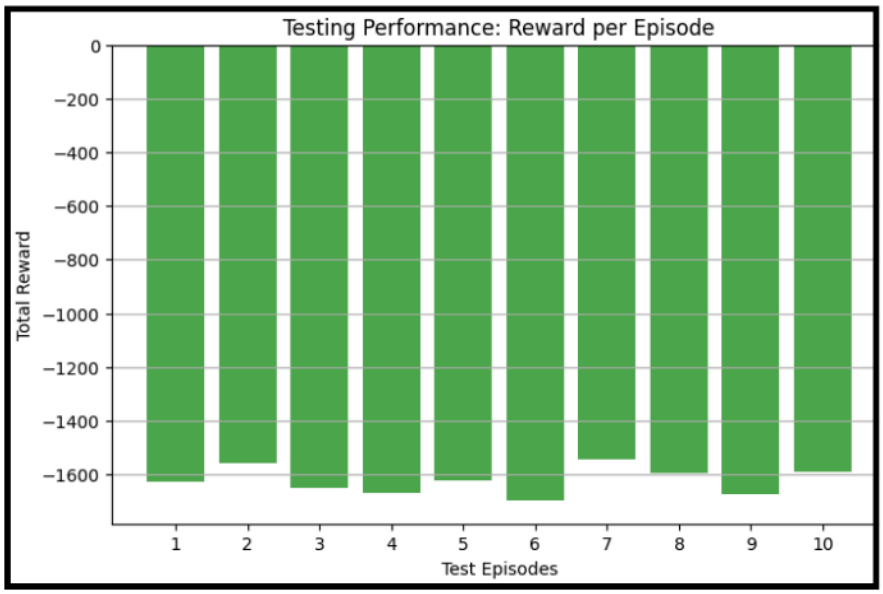


Figure 3: Testing Results in Phase 1

5.3 Phase 2: DRL with Feasibility Validation

With the introduction of feasibility-based classification in Phase 2, the reward curves shift upward (toward less negative values) and converge faster. This improvement indicates reduced invalid actions and more efficient learning during both training and testing phases.

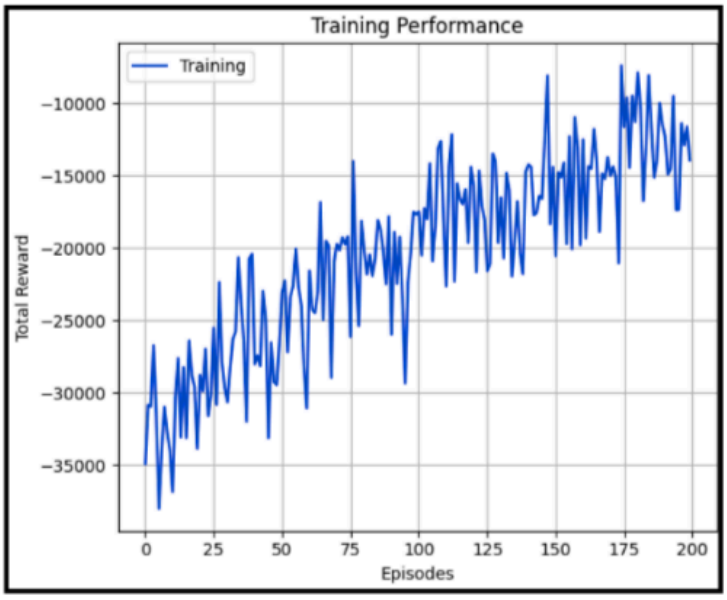


Figure 4: Training Results in Phase 2

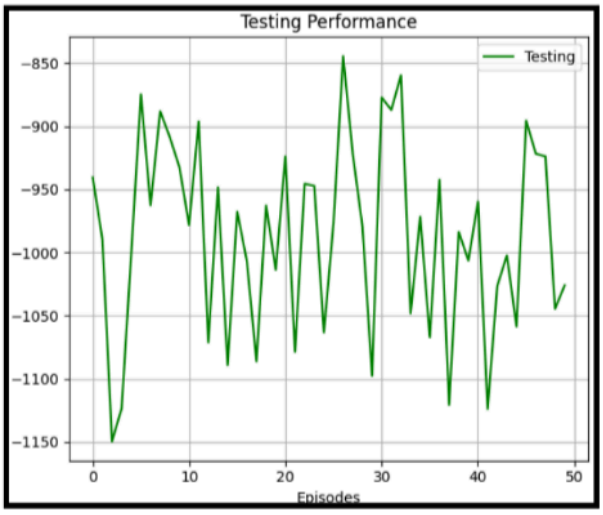


Figure 5: Testing Results in Phase 2

5.4 Phase 3: DRL with Enhanced Reward Shaping

Phase 3 demonstrates a further upward shift in reward values with significantly reduced variance. The enhanced reward shaping guides the agent toward energy-efficient and low-latency decisions, resulting in more stable training and improved testing performance.

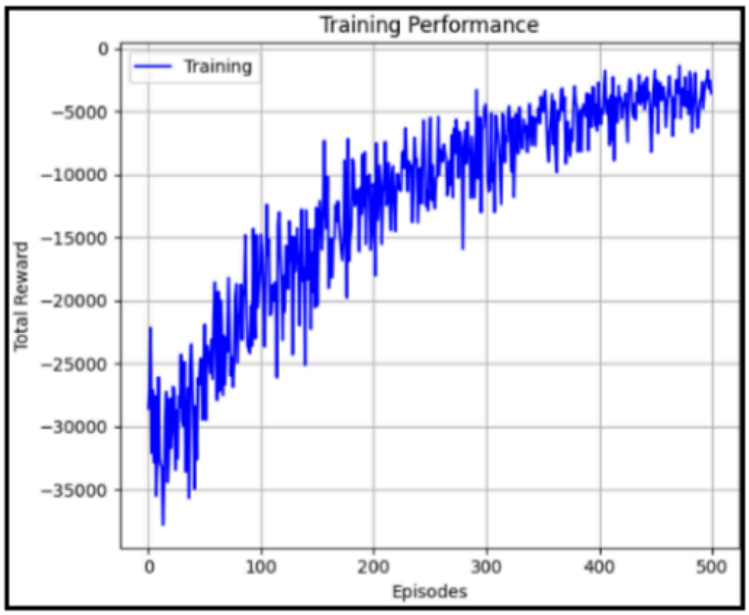


Figure 6: Training Results in Phase 3

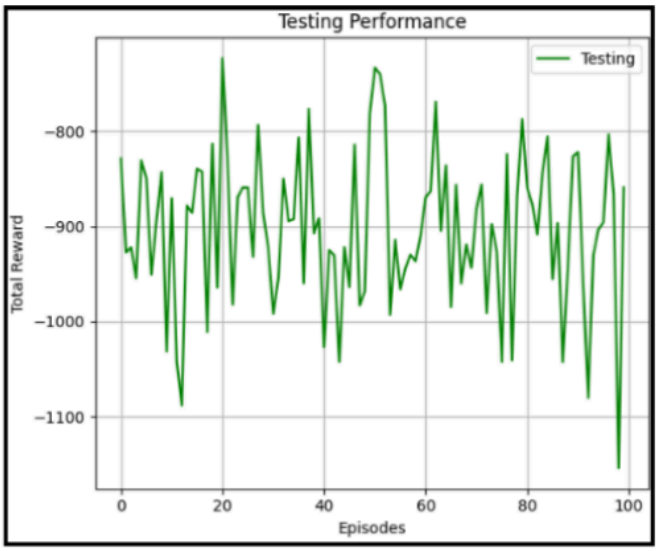


Figure 7: Testing Results in Phase 3

5.5 Phase 4: DRL with Micro-deadline Awareness

In the final phase, the reward curves achieve the highest values and exhibit the most stable behavior across episodes. The inclusion of micro-deadline awareness enables effective handling of time-critical tasks, leading to improved Quality of Service and overall system efficiency.

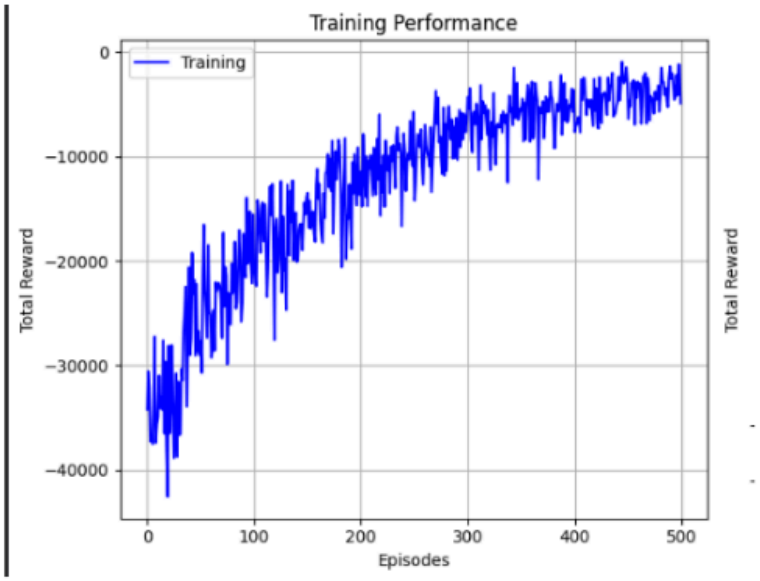


Figure 8: Training Results in Phase 4

Figure 9: Testing Results in Phase 4

6 Conclusion and Future Work

6.1 Conclusion:

- A DRL-based task offloading framework for UAV-enabled IoT networks was successfully designed and implemented.
- The integration of feasibility-based classification reduced invalid decision exploration and improved learning efficiency.
- Enhanced reward shaping and micro-deadline awareness led to better convergence, improved stability, and higher overall system performance.
- Phase-wise experimental results clearly demonstrate progressive improvements across all proposed enhancements.

6.2 Future Work:

- In the future, instead of a single Master UAV making all decisions, multiple UAVs can work together and share decision-making responsibilities. This would make the system more flexible and suitable for larger network areas.
- The current model assumes simplified network conditions. Future work can include realistic factors such as UAV movement, signal interference, and varying communication quality to better represent real-world environments.

- The proposed framework can be tested on actual UAV hardware or real-time simulation platforms to verify its practicality and performance in real-world applications.

7 Acknowledgement

We sincerely express our gratitude to **Dr. Abhishek Hazra**, our project supervisor, for his continuous guidance, valuable feedback, and encouragement throughout the development of this project on **DRL-based task offloading in UAV-enabled IoT networks**. His expertise and insights were instrumental in shaping the methodology and improving the overall quality of this work.

We also thank our team members for their cooperation and collaborative efforts, which contributed significantly to the successful completion of the project. Finally, we acknowledge our intuition for providing the necessary resources and a supportive academic environment.