Fırat Üniversitesi Deneysel ve Hesaplamalı Mühendislik Dergisi

Otonom Drone ile Kargo Teslimatı için Hibrit Bir Pekiştirmeli Öğrenme Yaklaşımı

Ebru KARAKÖSE[1*], Batuhan BAYRAKTAR [2]

[1,2]Sivil Havacılık Yüksekokulu, Fırat Üniversitesi, Elazığ, Türkiye.
[1]eozbay@firat.edu.tr, [2]batubayraktar26@gmail.com

Geliş Tarihi: 06.03.2025
Kabul Tarihi: 05.08.2025
Düzeltme Tarihi:29.06.2025
doi: https://doi.org/10.62520/fujece.1652790
Araştırma Makalesi

Alıntı: E. Karaköse ve B. Bayraktar, ’’Otonom drone ile kargo teslimatı için hibrit bir pekiştirmeli öğrenme yaklaşımı”, Fırat Üni. Deny. ve Hes. Müh. Derg., vol. 4, no 3, pp. 580-603, Ekim 2025.

Öz

Özellikle ulaştırma sektöründe ve kargo teslimatlarında drone kullanımı, önemli derecede ilgi çeken ve odaklanılan zorlu ve sınırlı konular arasındadır. Bu çalışmada, drone Unreal Engine yazılımıyla oluşturulan bir simülasyon ortamında, dışarıdan hiçbir bilgi almadan, hatta rota bilgisi bile olmadan haritanın merkezinden hareket ederek, kargoyu tamamen otonom bir şekilde teslim edebilmektedir. Drone’un görevleri arasında engelleri aşmak, hava koşullarından etkilenmemek, kargo aracını bulmak ve kargoyu sahiplerine teslim etmek gibi yetenekler yer almaktadır. Otonom olarak hareket eden drone’da kargo taşımacılığı ve navigasyon amaçları için RGB ve derinlik kameralarıyla birlikte üç farklı algoritma kullanılmıştır. Altı farklı kombinasyon oluşturularak birçok değişken açısından karşılaştırmalar yapılmıştır. Her kombinasyon, 150.000 adım boyunca eğitilmiş ve belirlenen metriklere göre değerlendirilmiştir. Drone, DQN, PPO ve hibrit Joint-DQN algoritmaları gibi pekiştirmeli öğrenme algoritmaları kullanılarak eğitilmiş ve hafıza için de LSTM algoritması kullanılmıştır. Bu algoritmalar simülasyon ortamında test edilmiş ve karşılaştırılmıştır. Ayrıca, RGB ve derinlik kameraları drone’a entegre edilmiş olup, her algoritma RGB ve derinlik kameralarıyla ayrı ayrı çalıştırılarak değerlendirilmiştir. Sistemde, drone hedefe doğru hareket ettikçe artı puan kazanmakta ve ters yönde hareket ettiğinde eksi puan almaktadır. Eğer drone bir engele çarparsa, simülasyon yeniden başlamaktadır. Elde edilen sonuçlar, algoritmaların önce engelleri aşmayı öğrendiğini sonra doğru yolu bulduğunu göstermiştir. Yeterli öğrenme süresi sağlandığında, drone görevini başarıyla yerine getirmiştir. Ayrıca, modeller performans açısından değerlendirildiğinde, DQN-RGB modeli en hızlı öğrenen model olarak tanımlanmıştır ve PPO algoritmaları tüm modellere kıyasla geride kalmıştır. Sonuç olarak, önerilen “Joint” katmanının öğrenme hızını yavaşlatsa da uzun vadede daha kararlı ve verimli bir model ortaya koyduğu belirtilmiştir.

Anahtar kelimeler: Otonom drone, Kargo teslimatı, Derinlik kamerası, Pekiştirmeli öğrenme, RGB kamera

[*]Yazışılan yazar

Firat University Journal of Experimental and Computational Engineering

A Hybrid Reinforcement Learning Approach for Cargo Delivery by Autonomous Drone

Ebru KARAKÖSE[1*], Batuhan BAYRAKTAR [2]

[1,2]School of Civil Aviation, Fırat University, Elazığ, Türkiye.
[1]eozbay@firat.edu.tr, [2]batubayraktar26@gmail.com

Received: 06.03.2025
Accepted:05.08.2025

Revision:29.06.2025

doi: https://doi.org/10.62520/fujece.1652790
Research Article

Citation: E. Karaköse and B. Bayraktar, "A hybrid reinforcement learning approach for cargo delivery by autonomous drone", Firat Univ. Jour.of Exper. and Comp. Eng., vol. 4, no 3, pp. 580-603, October 2025.

Abstract
The use of drones, particularly in the transportation sector and cargo delivery, is among the challenging and limited issues that attract significant attention and focus. In this study, a drone operates in a simulation environment created with Unreal Engine software, operating from the center of the map without any external information, not even route information, and delivers cargo completely autonomously. The drone's missions include overcoming obstacles, remaining unaffected by weather conditions, finding the cargo vehicle, and delivering the cargo to its intended recipient. Three different algorithms, together with RGB and depth cameras, were used for cargo transportation and navigation purposes in an autonomously moving drone. Six different combinations were created, and comparisons were made across a variety of variables. Each combination was trained for 150,000 steps and evaluated against predetermined metrics. The drone was trained using reinforcement learning algorithms such as DQN, PPO, and hybrid Joint-DQN algorithms, and the LSTM algorithm was also used for memory. These algorithms were tested and compared in the simulation environment. Additionally, RGB and depth cameras were integrated into the drone, and each algorithm was run and evaluated separately using the RGB and depth cameras. In the system, the drone earns positive points as it moves toward the target and receives negative points when it moves in the opposite direction. If the drone crashes into an obstacle, the simulation restarts. The results showed that the algorithms first learned to overcome obstacles and then found the correct path. Given sufficient learning time, the drone successfully completed its mission. Furthermore, when the models were evaluated in terms of performance, the DQN-RGB model was identified as the fastest learning model, with the PPO algorithms lagging behind all other models. As a result, it was noted that although the proposed "Joint" layer slows down the learning rate, it produces a more stable and efficient model in the long run.

Keywords: Autonomous drone, Cargo delivery, Depth camera, Reinforcement learning, RGB camera

[*]Corresponging author

1. Introduction

Before the advent of advanced algorithms and artificial intelligence, robotics faced significant challenges in areas such as navigation and route planning. Applications at that time were generally based on primitive techniques and procedures. Initial robots were typically programmed with specialized instructions for each movement, resulting in limited functionality and adaptability. In the late 20th century, advancements in computer science and technology paved the way for innovative approaches to problem-solving in robotics. The discovery of various path-planning algorithms marked a pivotal moment in the evolution of robot technology. Well-known algorithms such as Breadth First Search, Depth First Search, Dijkstra's algorithm, and A* Search were conceptualized and developed during this period. These algorithms introduced a paradigm shift from static, predetermined movements to dynamic, responsive actions, enabling robots to interact with their environments in unprecedented ways. However, these methods also had limitations; they often predicted inadequate paths, suffered from computational inefficiencies, and led to inadequate decisions by agents. Despite these challenges, they laid the foundation for advanced techniques and algorithms used in the ever-evolving field of robotics today [1-3]. In this context, deep learning methodologies developed to address these challenges have increasingly begun to surpass traditional software approaches when assessing robotics and real-world challenges. The amalgamation of extensive, well-annotated datasets and the advent of accessible, cost-effective computational resources have significantly amplified the capabilities of deep learning algorithms. The groundbreaking publication by Krizhevsky et al. [4] emphasized the viability of training deep neural architectures with abundant data to attain improved representations, marking a crucial shift toward the deep learning paradigm. Thanks to its exceptional capacity for generalization, deep learning has gained prominence across diverse disciplines, successfully addressing complex engineering challenges. These disciplines are evident in fields such as object detection, auditory recognition, conversational agents, robotic manipulation, and autonomous systems [5-6]. These advancements in deep learning have been instrumental in enhancing decision-making processes and learning algorithms, particularly through the foundational concepts of Markov Decision Processes (MDPs) and Reinforcement Learning (RL). MDPs provide a mathematical framework used to describe a stochastic process, comprising states, actions, transitions, and rewards. Transitions are utilized to determine the new state that the process will transition to after performing a specific action in a given state. Rewards define the amount of reward to be obtained following the execution of an action. The Markov property indicates that the next state of the process is dependent solely on the current state and the applied action. In other words, the next state of the process is independent of past states. This property renders MDPs an ideal framework for decision-making and learning algorithms [7]. RL is a method of finding optimal actions by allowing an agent to interact with its environment and learn from its experiences. An important aspect of the learning process in RL is storing interactions, known as experiences, in memory and selecting the necessary ones as needed. For this, a method known as Experience Replay is used, and more specifically, Prioritized Experience Replay (PER), which ranks these experiences according to their importance as determined by TD-Loss. RL has its own terms. You can see some of them below:

Agent: In the context of reinforcement learning, an agent is an autonomous entity that learns to make decisions by interacting with an environment.
Reward: A reward is the feedback signal given to the agent after performing an action; it provides a measure of success or failure and guides the agent's learning process.
Policy: A policy defines the agent's behavior at a given time; it maps the agent's states to the actions it should take in those states.
Episode: An episode refers to a sequence of states, actions, and rewards that ends in a terminal state, typically used in episodic tasks where the interaction between the agent and the environment naturally breaks down into distinct sequences.
Action-value function: Also known as the Q-function, it estimates the expected return (sum of future rewards) for taking a particular action in a given state under a specific policy.
On-policy learning: In on-policy learning, the agent learns the value function according to the current policy it's following, meaning it learns from the actions it takes.
Off-policy learning: Off-policy learning allows an agent to learn a policy independent of the policy used to generate the data, meaning it can learn from the actions of other policies as well.

Q-learning: Q-learning is a model-free reinforcement learning algorithm, an off-policy learner, where the agent learns an optimal policy even when it takes exploratory actions.

RL facilitates the agent's learning of the best actions by interacting with its surroundings. Furthermore, RL enables an agent in an MDP to learn which action is optimal in a given state, utilizing states, actions, and rewards. The agent’s objective is typically to maximize the cumulative reward or accomplish a specific task most efficiently. This approach involves the agent's adaptation to its environment through trial and error, seeking actions that lead to favorable outcomes and higher rewards. The theories of MDP and RL have found widespread applications in various domains. For instance, autonomous vehicles and robots utilize these concepts to explore their environments and determine optimal actions. Moreover, in the fields of artificial intelligence and machine learning, these theories contribute to the development and enhancement of learning algorithms. The ability of these techniques to provide effective solutions even in complex and uncertain scenarios renders them highly valuable. Non-Markovian decision problems refer to situations where future states are determined not only by the current state and action but also by past states and actions. These types of problems are typically more complex and challenging to solve, as incorporating historical information increases the dimensionality of the state space and complicates the learning process. It is of significant importance to acknowledge that many real-world problems, such as the navigation of autonomous vehicles, financial predictions, or energy management, are inherently non-Markovian. The solution to these problems often necessitates more complex algorithms and techniques. Consequently, the development of solutions for non-Markovian decision problems stands as a crucial research area within the realm of artificial intelligence and machine learning [1-2, 7-9].

In addition, studies involving drones also reveal innovative approaches in many areas. Among these studies, applications related to efficient road planning, energy efficiency, and the development of the most appropriate policies for racing stand out as real-world applications. It has been observed that faster results are obtained, especially when these applications are performed using drone technology. In some studies, focusing on efficient path planning, the authors proposed the Advanced TD3 model, which is designed for energy-efficient route planning for edge-level drones and takes into account factors such as wind [10-14].

Based on the challenges faced in robotics and the advancements made in deep learning, in this study, three different algorithms, together with RGB (Red, Green, Blue) and depth cameras, are used for cargo transportation and navigation purposes in an autonomously moving drone. Six different combinations are created for these three algorithms with two cameras, which will be explained in detail in the following sections, and comparisons are made in terms of many metrics, such as stability, performance, system status, and awards won.

1.1. Related works

Prior to the development of deep learning, researchers initially focused on drone-related tasks involving simple balancing and basic maneuvers. These activities required the use of action planning strategies and the tabula rasa approach to maintain a collection of situations. Incredible success has been seen in a variety of contexts as the power of computing has grown and deep learning has been incorporated into the field of reinforcement learning. Recent advances in technology have increased interest in various algorithms to solve both on-policy and off-policy configurations. These algorithms include DQN (Deep Q Network), PPO (Proximal Policy Optimization), SAC (Soft Actor Critic), Rainbow, A2C, and so on. In some studies, repetitive units and attention networks such as LSTM-RNN (Long Short Term Memory-Recurrent Neural Network) have even been used, resulting in better planning results in complex contexts.

The fact that RL requires safe training and processing steps in real-world applications requires high security measures to be taken in areas that require online training or agent-real-world interaction (for example, autonomous vehicles and robotics). Current RL simulation platforms often have limited capabilities or difficult processes. To overcome these deficiencies, suggestions were presented in the study [15]. The authors reviewed some existing platforms and communication protocols to create an effective framework for RL applications. Another study, Gozen and Ozer emphasized that the problem of visual object tracking in drone videos has gained importance with the increase in drone applications and the use of cameras. Although single-object tracking has been widely studied in traditional videos, there are not enough studies on this subject in drone footage. In this context, new drone datasets have emerged as an important step to address the problem of visual object tracking. In the study, the importance of new reward functions and appropriate action sets for drone datasets to improve the performance of RL-based trackers is stated. Experiments on the VisDrone and OTB-100 datasets demonstrated the effectiveness of the proposed approach. Additionally, significant contributions have been made by introducing a deep RL-based visual object tracker in drone videos [16].

In the study by Kim et al., various applications for Micro- Air Vehicles (MAVs) were mentioned. Since the limited sensitivity of GPS poses challenges for indoor autonomous navigation, Simultaneous Localization and Mapping (SLAM) and stereo vision-based solutions were proposed. A system that allows quadcopters to automatically navigate indoors and find specific targets using a single camera was presented. A classifier trained with the deep learning model Convolutional Neural Network (ConvNet) produced flight commands by imitating the actions of the expert pilot. Real-time experiments showed that the system accurately locates targets and performs autonomous navigation with a high success rate [17]. In the study by Kim and Chen, a distributed DRL (Deep Reinforcement Learning) for UAV (Unmanned Aerial Vehicle) navigation in highly dynamic environmental conditions was presented. The UAV navigation was split into two simpler tasks, with each task handled using the Long Short Term Memory (LSTM)-based DRL network, which processes only a fraction of the interactive data. Additionally, a new DRL loss function was introduced to seamlessly combine the solutions of both tasks. The results obtained demonstrated the efficiency of the approach when compared with existing DRL techniques [18].

UAVs are important in the field of IoT in the air, as they offer the opportunity to perform IoT services from significant altitudes [19]. To this end, Wang et al. addressed the challenges of autonomous UAV navigation in large and complex environments by framing it as an MDP and introducing the Learning with Non-Expert Assistance (LwH) algorithm. In contrast to previous methods that focused heavily on reward shaping, this study emphasized a sparse reward structure, where rewards were given only on task completion and provided unbiased solutions. LwH was evaluated according to various criteria through a specially created UAV navigation simulator. The results demonstrated the superiority of LwH in managing sparse rewards and the ability to generate navigation policies on par with dense reward environments [20]. Some studies were conducted with the idea that the drone should automatically avoid objects or situations that may cause harm. Drones with the capacity to avoid obstacles come to the forefront with trainings that focus mainly on supervised learning or RL. To emphasize this, Shin et al. examined two types of RL in their study. These types were the discrete action space, which excels in opinion dataset optimization but can lead to unnatural behavior, and the continuous action space, where an integrated U-net-based segmentation model with the actor-critic network is proposed. In performance evaluation in three different environments and against human pilots, the proposed algorithm revealed superior effectiveness, matched by the competence of expert pilots [21].

In a different study, Tiwari and Nadimpalli discussed the effectiveness of the model-based RL concept in continuous control tasks. By using a simple RL model motivated by the random search method, automatic guidance of an aircraft such as a quadcopter in indoor spaces was achieved. The proposed approach used a mapping strategy to represent the thrust state of the vehicle and kept physical model requirements to a minimum. Additionally, it was shown that by using natural policy gradients, high-performance solutions can be achieved for continuous control problems such as quadcopters. The unique contribution of this study is that it offers an alternative algorithm that is not model-based and provides fast learning [22]. In their study, Munoz et al. discussed the training of drones using DRL in delivery missions and aimed to improve existing RL methods in drone delivery. The authors proposed and developed a DDQN-based (Deep Q-network-based) RL approach. Additionally, a joint neural network (JNN) was developed to process state information. The proposed method performed better than previous solutions, achieved higher rewards, and reduced the variance of the results. JNN, which has a joint structure, provided more effective learning by integrating image and scalar state data [23].

Additionally, there are different applications related to drones in the literature, especially in studies conducted in recent years. Chen et al. developed a deep learning-based detection method for object detection from real images in drones and autonomous vehicles. Here, it was aimed at increasing the accuracy rate of not only large-sized objects but also small-sized objects in the object detection phase. It was stated that the results obtained by trying scenarios on two data sets for drones and one for self-driving vehicles were quite successful for both large and small objects [24]. Another study is a detailed review that touches upon current studies. Especially the studies in the last ten years were examined, and the progress made in drone applications and the techniques used during this period were mentioned. The evaluations performed were given in the form of visuals and graphs. In order to assist researchers and guide beginners in this field, a list has been created by examining many articles, and certain topics such as artificial intelligence, road planning, etc. have been reported in detail. Additionally, the study examined also mentioned the applications and advantages of using drones in the field of transportation, which is focused on in this article. In drone applications, the areas where more studies have been done and the areas where sufficient studies have not been done are stated, thus explaining where to focus as a research area. The main issues that need to be focused on in the future have been determined, and it has been mentioned that research on autonomous aerial vehicles will increase [25]. Thus, it made us think that the autonomous drone algorithm developed in this study might be the right choice. In terms of delivery problems, a routing application using trucks and drones together was developed by Thomas et al., especially for last-mile deliveries. By using a truck and a multi-drone, it was aimed at reducing delivery time, which is one of the most important priorities for delivery problems. For this purpose, a mixed-integer linear programming model was created, and an intuitive approach was proposed by considering two stages in terms of truck and drone. It was stated that experimental results were evaluated in terms of customer density, drone usage was higher, and an advantage was provided in terms of delivery time [26].

Using drones in delivery has great advantages but also limitations. For this reason, Bi et al. emphasized that truck-drone applications have become more important in order to overcome drone limitations, and an application was carried out on this subject. Studies were carried out to provide optimum time with DRL using real maps for route planning optimization. As a result, comparisons were made with different RL algorithms, and it was decided that the proposed algorithm was suitable and advantageous [27]. Although the aforementioned approach and the approach proposed in this study are in similar areas, they offer solutions from different perspectives. This study offered a more realistic and customizable environment in the "Unreal Engine" environment, while the aforementioned study used a "gym-based" environment, and the results were obtained and evaluated for both action-constrained and without-action-constrained situations. The presence of an action constraint allows the model to learn faster and yield clearer results. However, considering the real world, these action constraints are not very realistic. Therefore, the model in this study was trained without action constraints, and despite the lack of action constraints, quite successful results were achieved. In another study where RL was used in drone applications, it was planned to maximize the gliding range of unmanned aerial vehicles. This problem was considered a complex nonlinear problem because it has a 12-dimensional state space and a 2-dimensional action space. An RL-based MRL, that is, a model-free RL approach, was proposed to solve such a problem. The success of the proposed method was demonstrated through analyses performed under complex and variable environmental conditions. While this method has continuous state-action spaces, the RL-based method proposed by us in this article has variable state-action spaces that are affected by factors such as delivery points and traffic data, and minimization of delivery time is determined as the reward function. As a result, both of these studies clearly demonstrate the flexibility of RL in different applications in terms of drone applications [28].

1.2. Motivation

Nowadays, autonomous cargo transportation with drones is still limited. Even with such limited use, drones can deliver cargo by following predetermined routes or by being controlled. Communication during the mentioned process is generally carried out via the remote control, and communication problems arising from the remote control also occur. Therefore, it has become necessary to evaluate the limitations and conduct studies to enable autonomous cargo transportation. In a study by Karaköse et al., multiple deliveries were carried out using a single drone without using a truck. The delivery order was determined by the TSP algorithm, and a QR code-based distribution system was designed for customers during the delivery phase [29]. However, using only a single drone has disadvantages in terms of delivery time and has some limitations. Therefore, Karaköse evaluated the application of a genetic algorithm-based hybrid truck-multi-drone system to overcome this problem in another study. In this system, which provides a great advantage in terms of delivery time, different scenarios for TSP were created, and the proposed algorithm was implemented. A truck and multiple drones communicated with each other and worked very harmoniously. When the simulation results were examined, it was seen that the use of multiple drones provided a great advantage in terms of time, as predicted. In addition, the optimization study carried out for real-world delivery problems, taking into account traffic data, to ensure that the vehicles work harmoniously and provide a time advantage, yielded successful results [30]. Unlike the aforementioned studies, the intended application in this article is to implement a new and different application by making each drone independent from each other and from people and enabling them to perform their duties. The drone finds the cargo vehicle or cargo owner autonomously, without any instructions or route information. While doing this, the tool does not use any program but only the RL algorithm written in Python created in the study. In this way, the task can be performed without the need for extra data or other commands. If the reasons that led to the emergence of this study and the planned gains are listed:

- Eliminating the route and movement planning required for drone cargo delivery.
- Eliminating all intermediary software used for cargo delivery by drone.
- To minimize human labor and error in this regard.
- To contribute to the literature with a different and new perspective and to provide a new algorithm.

1.3. Problem statement

Unreal Engine software is used by game developers and companies that develop large-scale games. The same program is also applied in the fields of shooting short films, making realistic models, AI (Artificial Intelligence), and simulation. In this study, the simulation map containing the drone has been developed on Unreal Engine 4.25.4. The reason why Unreal Engine is preferred is that it is more modern and has better visual capabilities than other simulation programs. A rectangular city containing dozens of houses, dwellings, parks, and roads has been designed in the Unreal Engine environment. After the Unreal Engine environment has been set up to be suitable for the drone, it has been trained in this environment with RL. In the study, a random image taken from the simulation map created in the Unreal Engine environment is given in Figure 1.

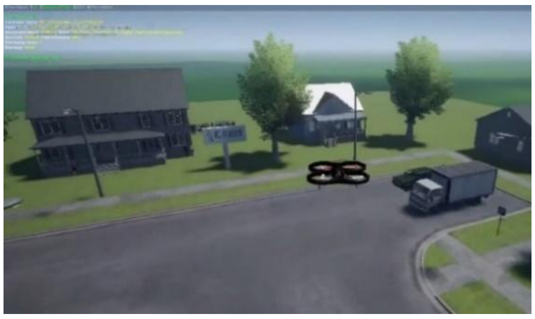

Figure 1. A random simulation image of the city designed in the Unreal Engine environment

In this simulation environment, two different cameras and three different algorithms have been used for the drone. The cameras are RGB and depth cameras. The reason why these two cameras have been chosen is that they are the most commonly used camera types in real life. At the same time, the purpose of using the cameras is to provide the necessary data entry and comparison to feed the algorithm. As with the camera, multiple options have been used for the algorithm. Three different algorithms have been included in the study, and all of them have been successful when given enough time. These algorithms are the DQN algorithm, the PPO algorithm, and the hybrid Joint-DQN (Joint-Deep Q Network) algorithm designed for this study. Additionally, LSTM (Long Short-Term Memory) has been used as a separate algorithm for memory. Algorithms make decisions according to a certain "reward" and "punishment" method. Drones and algorithms have two main data sources. The first of the data sources is the image seen by the camera, and the second is the points, which can be negative or positive, given to each movement made by the drone under the management of the algorithm. The definition of the problem in this study is simply to test and realize whether better results can be achieved with a hybrid model by combining pre-existing models in certain proportions.

1.4. Contribution

The main contribution of this paper is to present a detailed examination of a vision-based RL system designed for mobile robotic planning tasks. Training is performed with two different algorithms, off-policy DQN and on-policy PPO, and two different cameras, RGB and depth, to observe how the algorithms react to changing settings. In order for the planned task to be successful, the agent must be able to pick up packages from a moving vehicle in a dynamic environment and deliver them to appropriate houses. Various metrics are used in the system to evaluate and compare results. These metrics are: dataset sizes, training times (the time required to develop a solid policy with a success rate of 80% or higher), successful deliveries detected by the system, delivery time (the time required for the drone to successfully deliver a package, taking into account distance and learned policy), reliability interval, stability of Q values, and the impact of neural network hyperparameter selection.

Furthermore, the decision to train the model without action constraints sets it apart. This approach, while more realistic, makes it applicable to real-world drone operations where such constraints are absent or difficult to implement. This study aims to implement route and motion planning for drone cargo delivery, eliminate intermediary software, and minimize human error. This focus on complete autonomy and minimal external input is a key innovation compared to systems that require predetermined routes or constant remote control.

2. Proposed Method

Reinforcement learning serves as the brain of the drone in this study. The algorithms used are the fastest-learning DQN, the proven PPO, and hybrid joint-DQN algorithm. Many tasks, such as planning the movement, creating a route, avoiding obstacles, and detecting targets, are performed by these algorithms. In addition, the LSTM algorithm, like others, provides reinforcement learning for the drone. The purpose of this algorithm is to serve as memory. Because if the simulation stops as a result of a crash or error in the system, the system loses the information it last learned, and additional time is needed to compensate.

DQN algorithm: The DQN algorithm combines deep learning and Q-learning techniques. As a result, a Q-network is used rather than a Q table. Each state-action pair’s estimated Q values are contained in the Q-network. There are two other neural networks. The “Q-network,” the first, is used as a training tool to create the best state-action values. The “Experience Replay” network is the second and last network. In order to generate data for the Q-network’s training, this network interacts with the environment. Q-value is an abbreviation for quality, and it refers to a quality measurement.

PPO algorithm: Unlike the DQN algorithm, the PPO algorithm does not condense everything into a single equation. In actuality, PPO is an optimization algorithm. The algorithm records the outcomes of both the current and earlier calls to a function. Later, it increasingly eliminates and gives more weight to the motions it wants to achieve from a probabilistic standpoint.

Joint-DQN algorithm: The Joint-DQN algorithm differs from standard DQN by the inclusion of a layer called a "Joint" layer. The key difference between the two lies in the preprocessing layer of the DQN, which refines the visual input before feeding it to the Q-network. The joint layer requires a mathematical operation to process the camera image. This layer uses a mathematical tangent function designed to process image data from the camera. This mathematical function eliminates noise, resulting in images more suitable for the task. A fast algorithm like DQN can benefit from more specific data thanks to the design of the Joint-DQN algorithm. This design also enables the Joint-DQN algorithm to provide more precise data. The pseudocode of the proposed Joint-DQN algorithm is shown in Figure 2.

```
Decision Process with Joint Neural Network (JNN)
1:  Inputs:
        Current State, $s_t$, which includes:
          - Depth Image, $I_t$ (144x256 pixels)
          - Scalar Vector, $V_t$ (velocity, angle to goal, elevation angle, distance to geo-
    fence)
2:  Output:
        Selected Action, $a_t$
3:  Procedure:
4:  Feature Extraction:
5:  $F_t \leftarrow$ Flatten(CNN($I_t$))    ▷ Process the depth image $I_t$ via a CNN and flatten the
    output.
6:  Joint Layer Concatenation:
7:  $J_t \leftarrow$ Concatenate($F_t, V_t$)    ▷ Join the image features ($F_t$) and scalar data ($V_t$) into a
    unified representation ($J_t$).
8:  Value Estimation:
9:  $Q(J_t, \cdot) \leftarrow$ Q-Network($J_t$)    ▷ Feed the joint vector ($J_t$) into the Q-Network to
    estimate Q-values for all possible actions.
10: Action Selection:
11: $a_t \leftarrow \arg\max_a Q(J_t, a)$    ▷ Select the action ($a_t$) with the highest Q-value.
12: Action Execution:
13: Execute the selected action ($a_t$) in the environment.
```

Figure 2. Pseudocode of the proposed Joint-DQN algorithm

LSTM algorithm: In areas where data evolves over time, such as time series analysis, sentiment analysis, and text production, LSTM is frequently employed. It is used to process information that has been gradually learned. What distinguishes LSTM from other memory-capable algorithms is its independence in deciding what to remember and what to forget. The RL structure implemented in this study is given in Figure 3. Here, first the drone starts to move on the map, then it tries to find the cargo vehicle and deliver the cargo. While doing all this, it uses three different RL algorithms, such as DQN, PPO, and Joint-DQN, and the whole process takes place in the Unreal Engine environment.

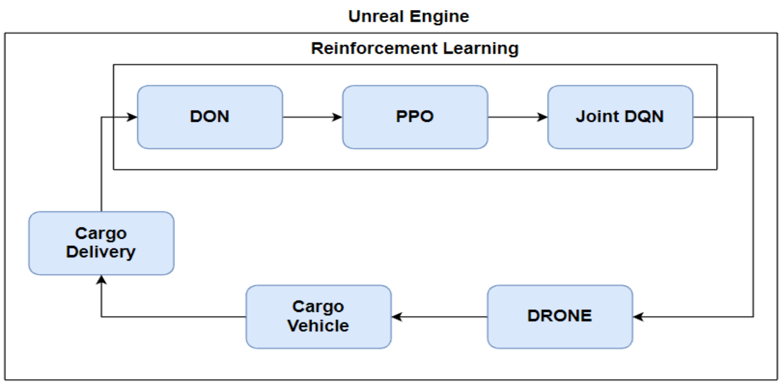

Figure 3. The developed RL architecture for autonomous cargo delivery

2.1. Algorithm design

If the stages of method design are explained at the most fundamental level: The drone performs unit movements, each of which is referred to as a “step.” When a drone’s simulation is successful or unsuccessful, such as when it crashes into something, a new simulation is launched. This new simulation is referred to as an “episode”. The drone receives plus points in every step when going to the cargo vehicle to pick up the cargo or leaving it to the cargo owner after receiving it. In the opposite case, it receives negative points. Hitting an obstacle is the worst-case scenario and will result in minus 500 points. If the entire episode is successful, a bonus of 150 points is given. In this way, the aim is to make the fewest mistakes rather than deliver the cargo at first. Figure 4 shows visuals showing the movement of the drone on the map created in the Unreal Engine environment.

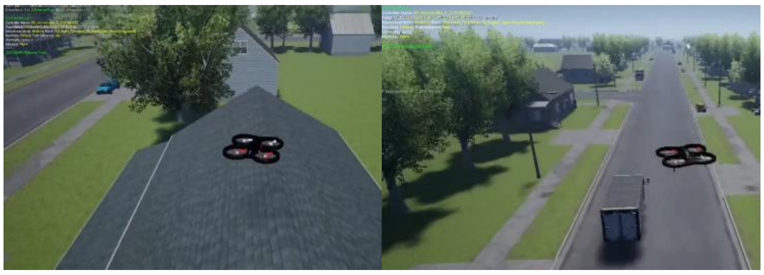

Figure 4. Drone movements on the simulation map

Each episode in the simulation scenario will proceed as follows:

- The drone is activated on the empty land in the middle of the city. To avoid potential loss scenarios, the drone always begins at the middle of the map and continuously picks up cargo from a moving cargo vehicle.
- The drone's first task is to find the cargo vehicle that is constantly moving around the city because the cargo vehicle carries the cargo that the drone must leave. As the distance decreases as the vehicle progresses, it will gain "+" points.
- The drone must go to the cargo vehicle and pick up the cargo. It will earn a lot of "+" points since, by taking up the cargo, it completes the first crucial task.
- The system then chooses any home or individual at random to be the cargo owner. The package delivery to this cargo owner is the drone's new assignment. As the drone moves towards the target and the distance between them decreases, it will gain "+" points.
- After the drone delivers the package, the episode ends successfully, and the next episode begins. A large number of "+" points are received as a result of package delivery.
- The episode will end, and the next one will begin if an error like a collision is made. A collision results in a large number of "- " points. Additionally, walking away from the cargo vehicle when there is no cargo or from the house that owns the cargo when there is cargo earns "- " points.

This basic progress described will be applied to each algorithm and camera type used in the study, and all results will be compared. The flow diagram explaining the purpose and reward-punishment structure of the proposed RL algorithm is given in Figure 5.

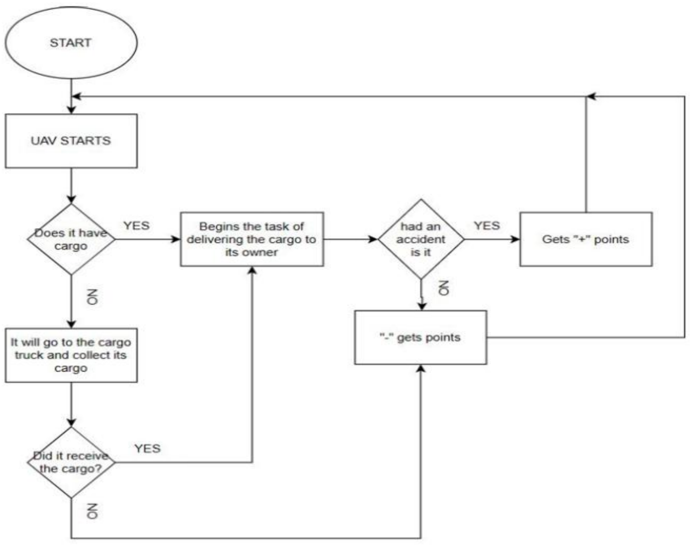
START
UAV STARTS
Does it have cargo
YES
NO
Begins the task of delivering the cargo to its owner
had an accident is it
YES
NO
Gets "+" points
"-" gets points
It will go to the cargo truck and collect its cargo
Did it receive the cargo?
YES
NO

Figure 5. Flow chart of the proposed RL system

3. Experimental Results

Considering that two cameras and three algorithms have been used in the study, six different combinations have been obtained as a result. These combinations are DQN-RGB, Joint-DQN-RGB, PPO-RGB, DQN-DEPTH, Joint-DQN-DEPTH, and PPO-DEPTH. It is not appropriate to apply a joint layer in PPO. In order to challenge the models, 150,000 steps have been allowed to be trained for each model. Although this 150,000-step training varied from model to model, it generally lasted between 24 and 28 hours. The resolution received from the cameras is 224x224. The reason why lower resolution is preferred compared to other image processing studies is to challenge the algorithms and measure their reactions to difficult conditions. To assess the model's efficacy in this case, numerous metrics have been considered. The majority of these characteristics are based on gathered scores. Depending on the model and camera, the number of metrics used to evaluate the model's performance ranges from 37 to 43. While the model with Joint-DQN is cramped in a narrower range in the compared metrics, the classical DQN model has a very wide range. The results for DQN-RGB without joint are shown in Figure 6. Variables used for models; agent_timesteps_total, done, episode_len_mean, episode_reward_mean, episode_reward_max, episode_reward_min, episodes_this_iter, episode_length, max_q, min_q, mean_q, cpu_util_percent, ram_util_percent. If the variables used are explained in detail;

- The metric agent_timesteps_total provides the number of steps taken during the simulation.
- The done value indicates the point at which the mission was successful for the first time.
- Episode_reward_min specifies the minimum value, while episode_reward_max indicates the maximum value.
- Episode_reward_mean represents the average score collected in each simulation.
- Episodes_this_iter reports the number of episodes in each iteration.
- Episode_length indicates the number of steps taken during the episode.
- Cpu_util_percent and ram_util_percent represent CPU and RAM usage rates.

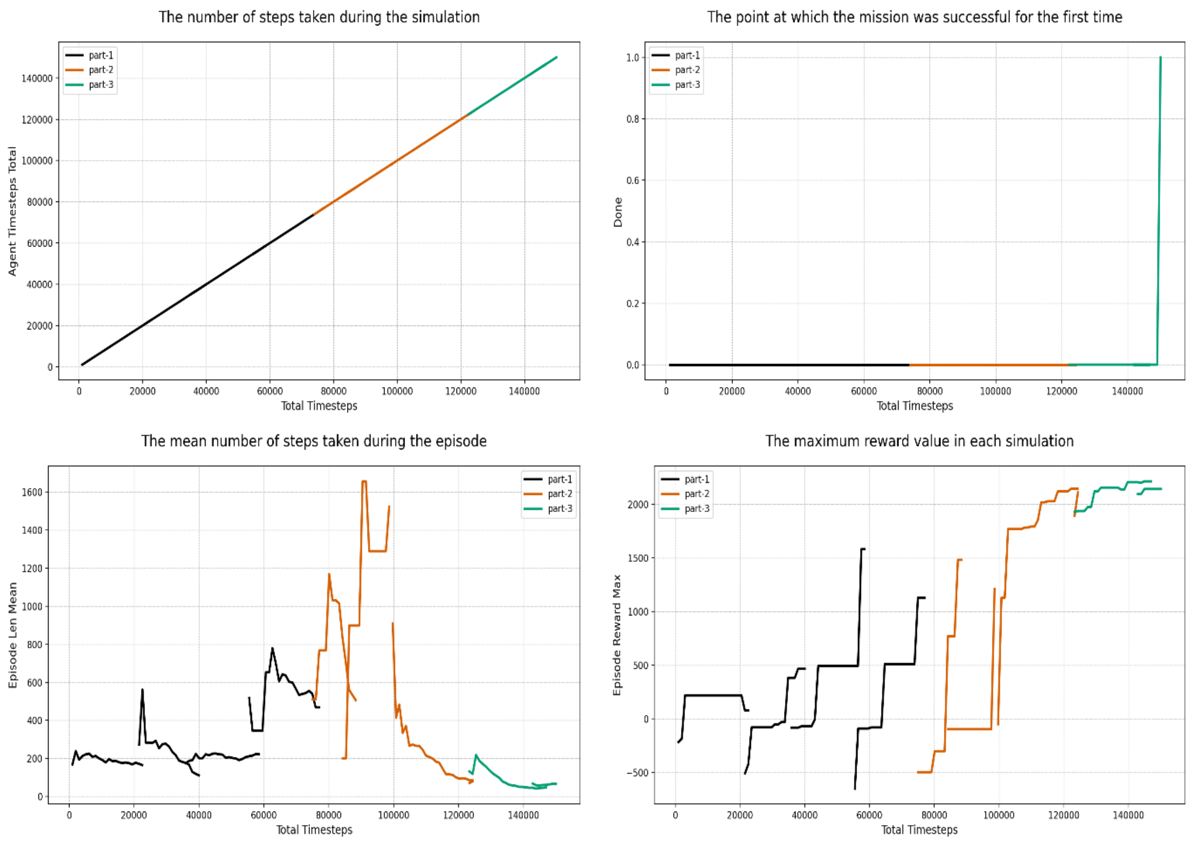

Figure 6. The results for DQN-RGB without joint

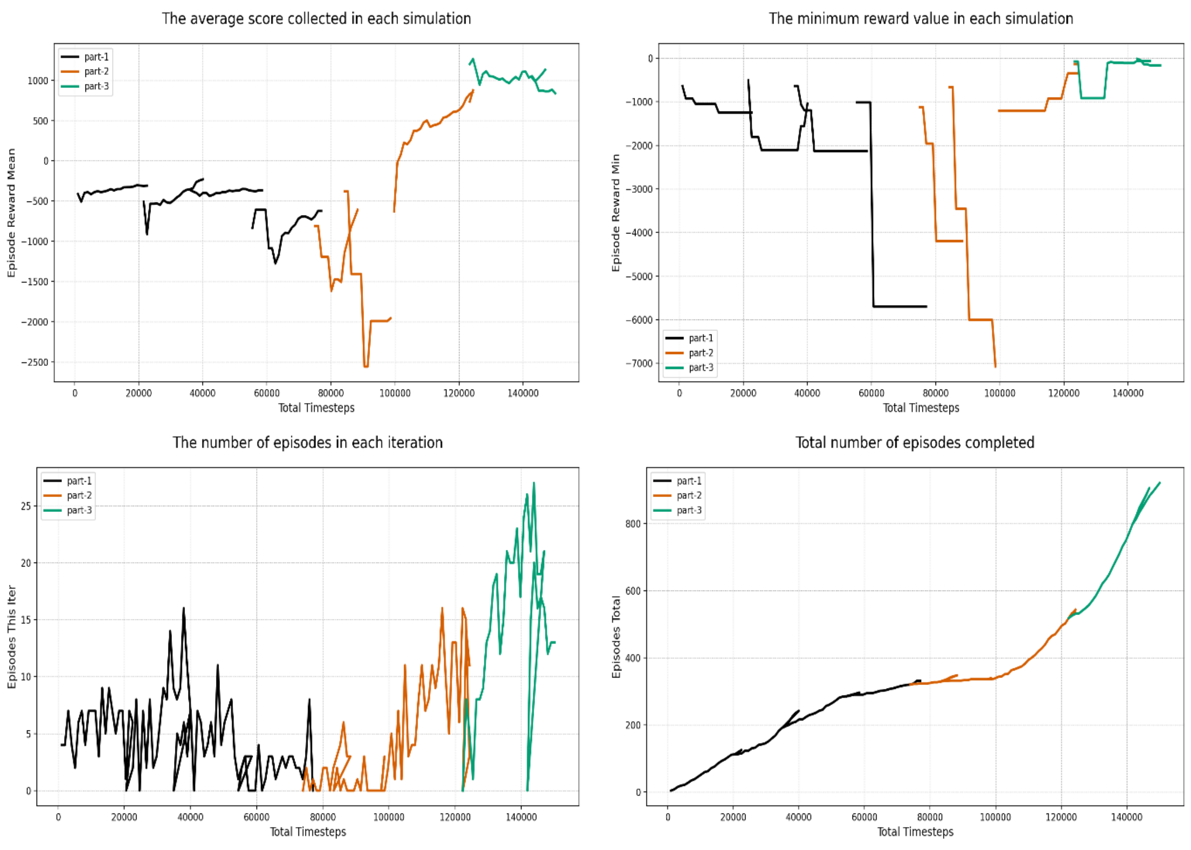

Figure 6. (Continue) The results for DQN-RGB without joint

Metrics related to reward indicate the average, minimum, and maximum reward scores within each episode. Joint-DQN-DEPTH has increased healthily in all three variables. This is valid for the DQN-DEPTH model without joints, but the positive and negative ends are much wider. If the simulation results obtained are examined; As seen in Figure 6, the graph is shown in three different colors because there are two collapses in the system. Collapses are caused by heavy RAM usage. The system reboots when the RAM limit is exceeded. When it restarts, the system is shown in a different color in the graphics, and the period after the collapse is represented by different colors. Since the information kept in memory is lost after the collapse occurs, it is possible to see a certain amount of decline followed by an increase. Again, in the same model, there is a certain amount of decline after each collapse. Despite this, there is an upward movement in three charts. These graphics are “episode_reward_max”, “episode_reward_mean”, and “episode_reward_min”. While these move upward, “episode_len_mean” decreases. This is a very good sign for system operation.

The DQN-RGB model without joint in Figure 6 needs to be examined more closely as it collapses twice. Before the collapse, the “episode_len_mean” value was generally between 200 and 600, but after the first collapse, it permanently fell below 500. After the last collapse, although it could exceed 100 at first, it fell below 100 towards the end. This is one of the important signs that the model has matured. Although there are occasional drops in reward-related metrics, the trend is upward. The fact that these metrics tend to increase shows that there is no problem. Regardless of the short-term problem, productive results are achieved when training is given enough time, which shows that the model can be considered successful. A system collapse also occurred in the Joint-DQN-RGB model shown in Figure 7. That’s why the graphs are shown in two different colors. Again, heavy RAM utilization is the cause of these collapses. Moreover, each collapse takes with it the information it acquired during the 15 to 20 thousand steps before the moment of collapse. The purpose here is to test the boundaries of the algorithms under time constraints; hence, this is not done, even though additional time should be given after each collapse. In addition, in the Joint DQN-RGB model, while all values are initially horizontal, all values except "episode_reward_max" experienced a dramatic decrease after the collapse. As a result, it is observed that the joint layer is more susceptible to collapse compared to DQN-RGB without joint.

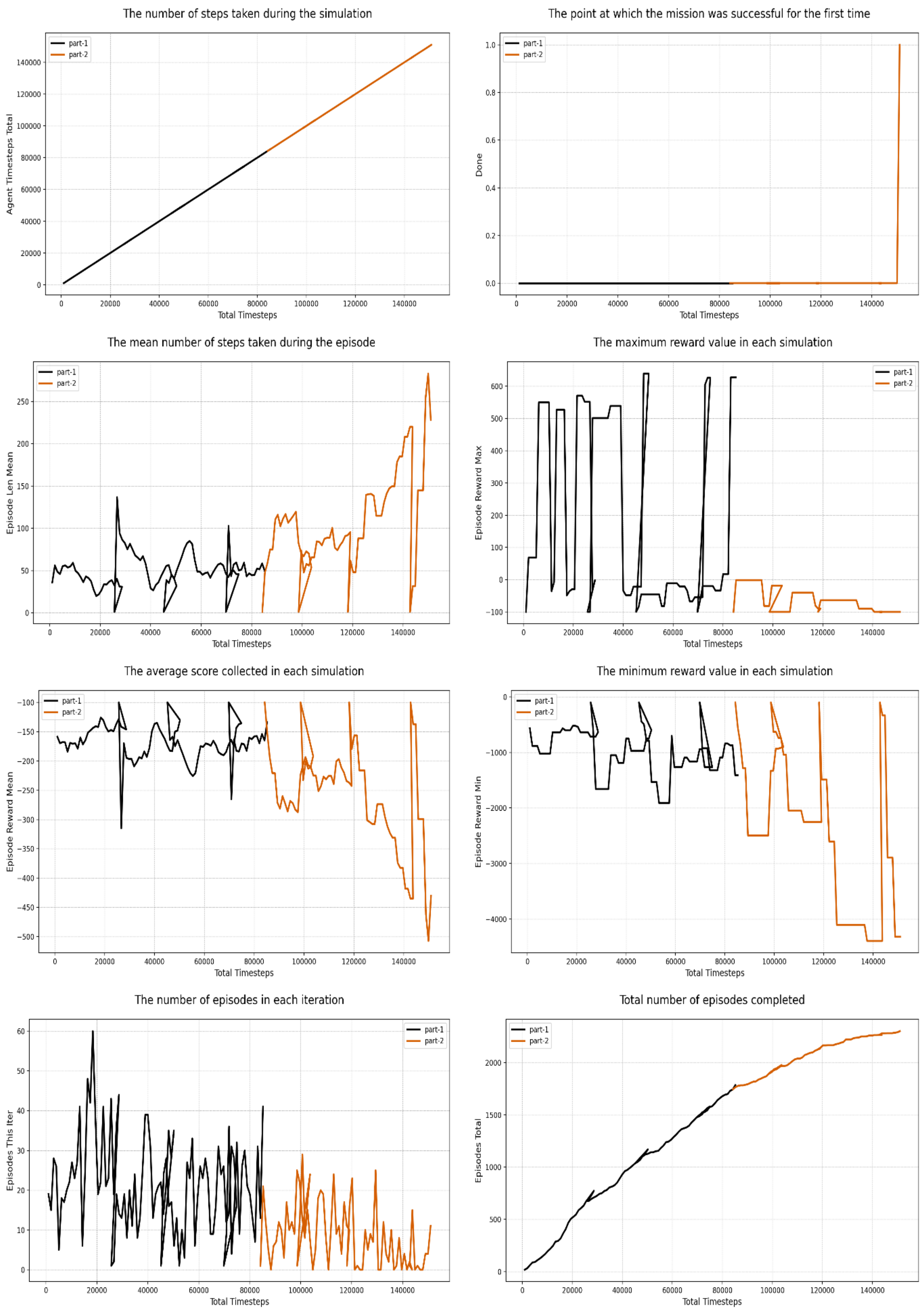

Figure 7. The results for Joint-DQN-RGB

The system collapses observed in the DQN-RGB without joint and Joint-DQN-RGB models were primarily due to excessive RAM usage. Exceeding the RAM limit caused the system to reboot. When a collapse occurs, information held in memory is lost, causing a temporary decrease in performance. Solutions can be found in cases of system collapses by implementing measures such as memory optimization, gradient clipping, and batch size adjustments. However, while methods such as gradient clipping can help stabilize training time, they also have the potential to lead to numerical instability and higher memory usage during backpropagation. Alternatively, smaller batch sizes can be used to reduce instantaneous RAM consumption, but this could affect training speed or convergence. Therefore, in this study, to better test the algorithms' limitations under time constraints, no additional training time was considered after these collapses. Furthermore, because training data is recorded regularly, in the event of a RAM collapse, data learned by the model but not yet recorded is lost, and the system restarts from the most recent recording after the collapse.

With the exception of "q" values, the variables utilized for PPO are the same. Utilizing advantage functions to enhance the existing strategy and maximize these advantages is the core concept of PPO. As a result, it adjusts the strategy and employs advantage functions rather than computing the "q" value. In the PPO-RGB model, most values contain noise, as seen in Figure 8. This model also has differences compared to the PPO-DEPTH model. It is obvious that the PPO model will perform poorly for this task because of its different inherent structure from DQN. Additionally, the results obtained from the depth camera in the PPO model are more dynamic than those obtained from the RGB camera. RGB, on the other hand, is stuck in a short range because it contains more information.

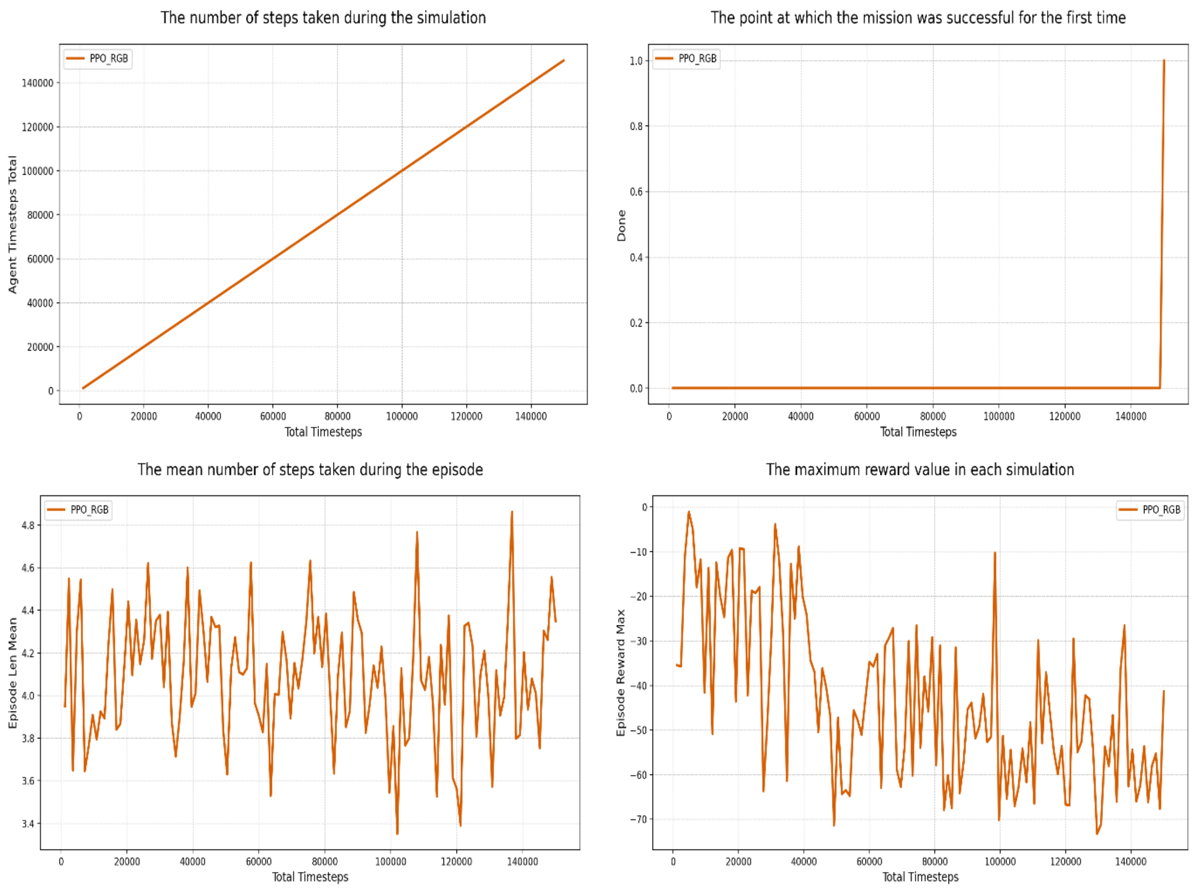

Figure 8. The results for PPO-RGB

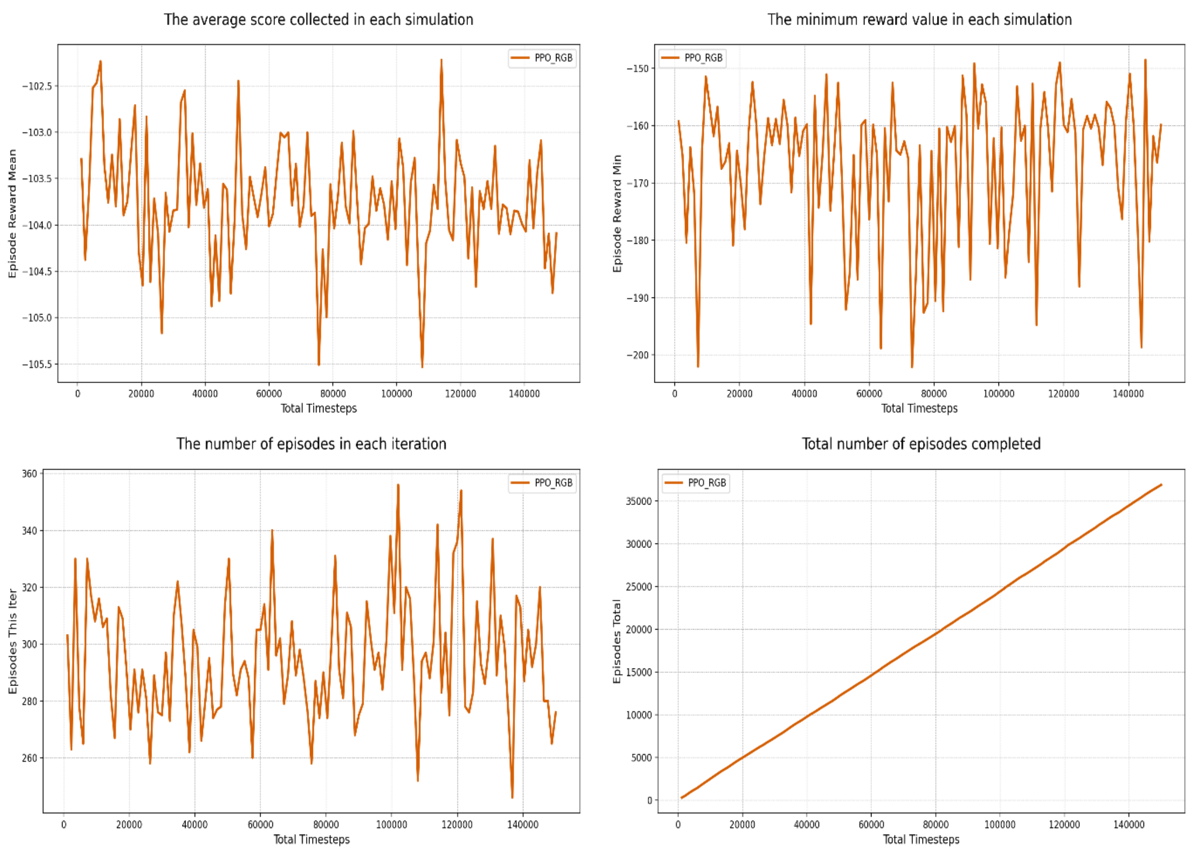

Figure 8. (Contineu) The results for PPO-RGB

Looking at the overall number of steps taken, or the number of steps the drone took up until it ran into something or finished the assignment, from a different perspective; According to Figure 9, the "episode_len_mean" variable of the DQN-DEPTH model without joint starts from around 200 and decreases to around 100 steps. Considering the graphs related to reward, the reward value increases rapidly while the steps taken/time spent per episode decrease. Although the increase in the reward value alone is a good sign for the model, the decrease in the number of steps clearly shows that the model learns faster.

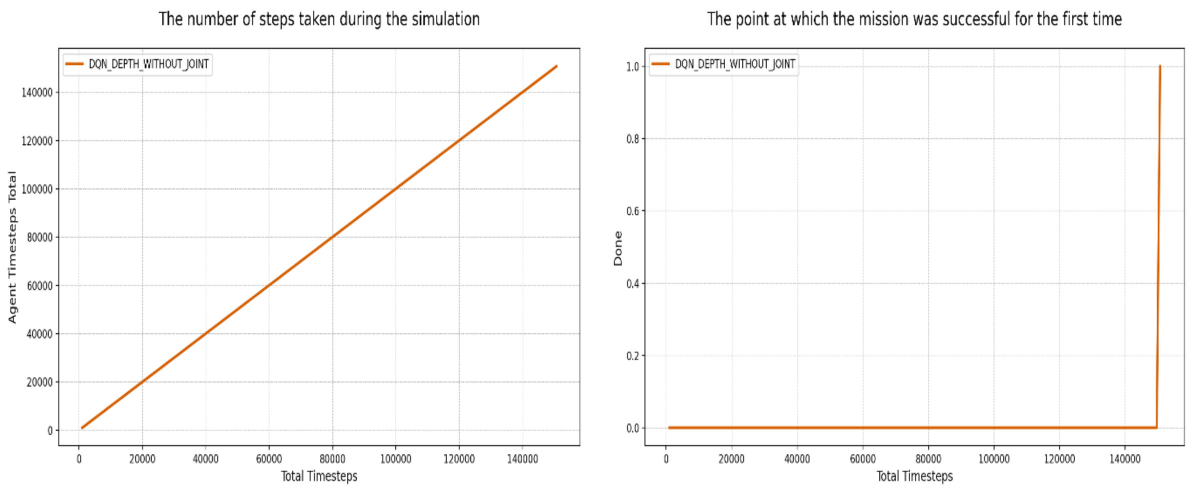

Figure 9. The results for DQN-DEPTH without joint

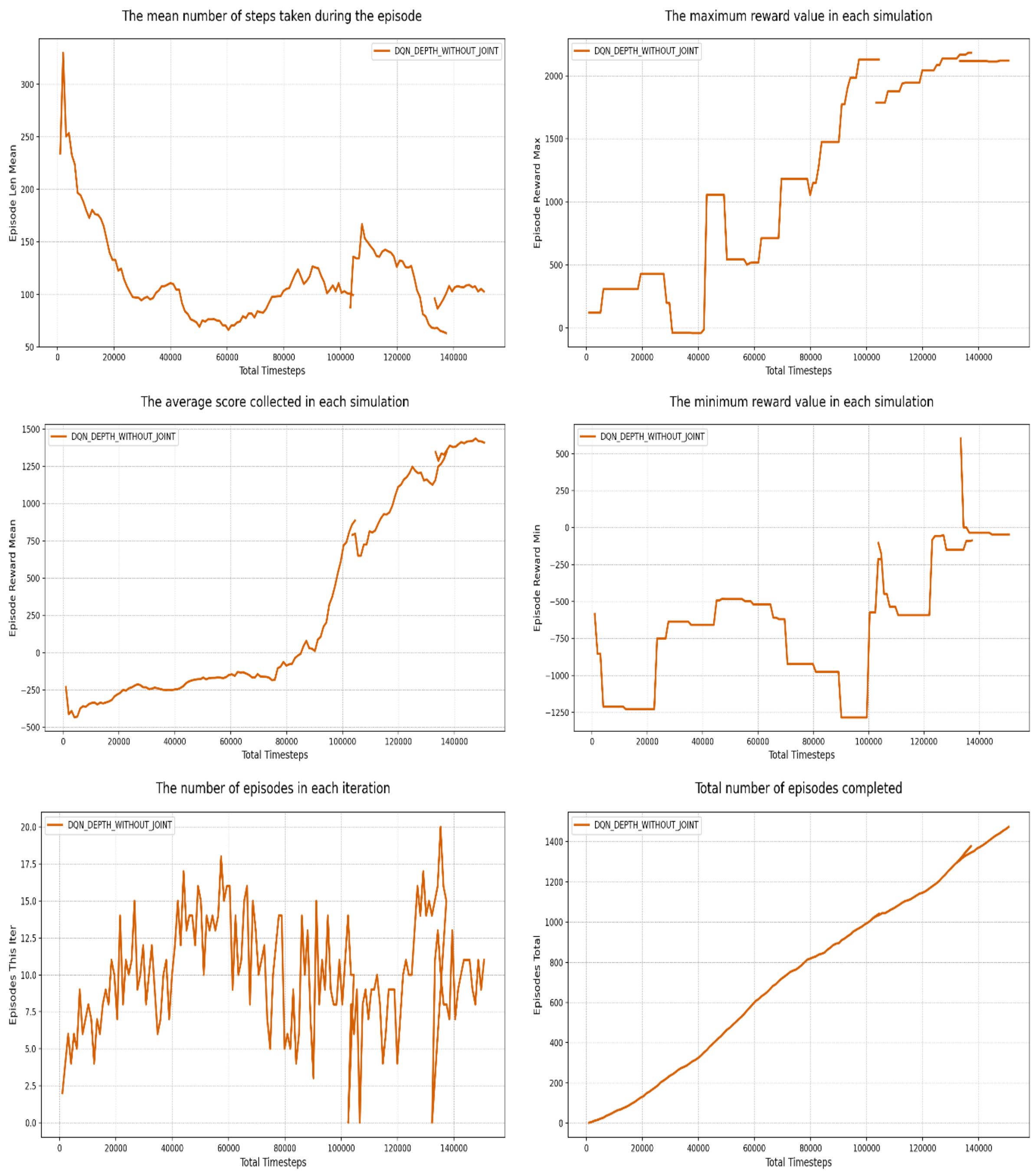

Figure 9. (Continue) The results for DQN-DEPTH without joint

For the Joint-DQN-DEPTH model, while the “epsiode_len_mean” decreases, the rewards increase, as shown in Figure 10, showing that the desired performance is achieved.

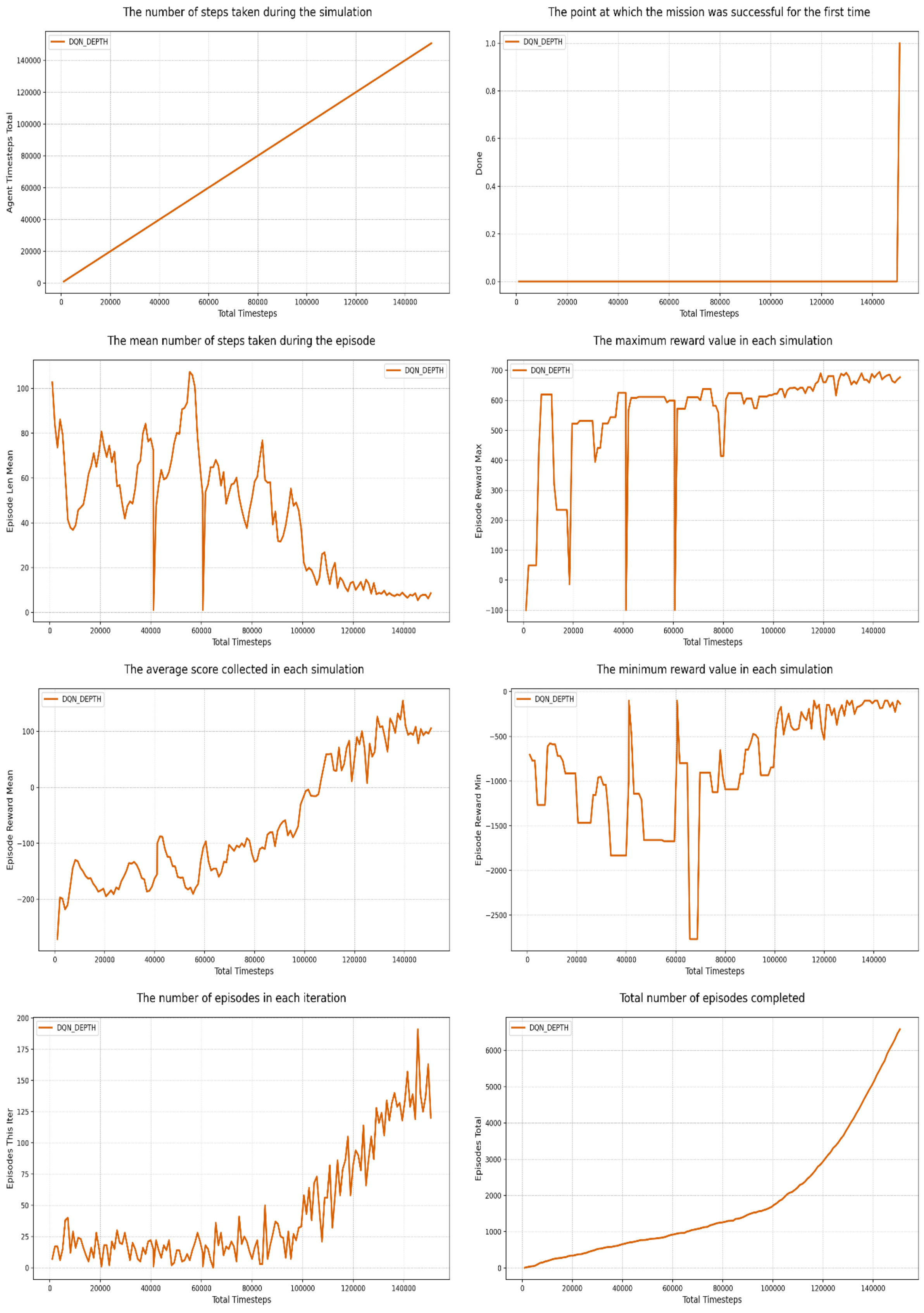

Figure 10. The results for Joint-DQN-DEPTH

In the PPO-DEPTH model given in Figure 11, the “epsiode_len_mean” value being 10 at the beginning and 20 towards the end is not a good sign. The reason for this situation is that reward-related metrics do not display a successful graph. The “reward_mean” and “reward_min” values have remained constant throughout almost the entire simulation and experienced a serious collapse in the last moments. However, this stable trend has negative values. Although the max value is more positive than the others, factors such as remaining at the local minimum value, task and algorithm incompatibility, time limitation, and collapse have negatively affected the situation. Additionally, the PPO-RGB model in Figure 8 also performs poorly, although not as much as the PPO-DEPTH.

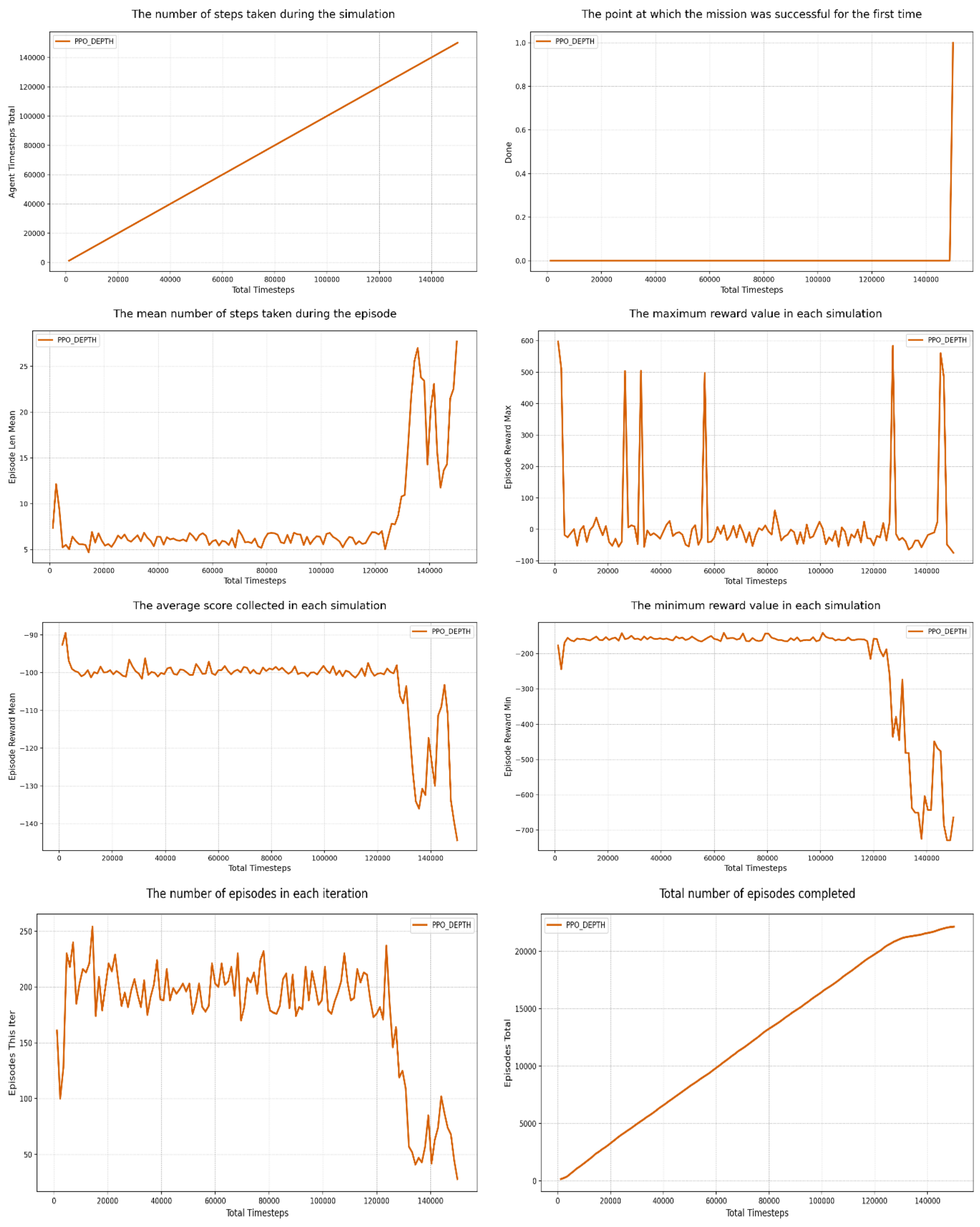

Figure 11. The results for PPO-DEPTH

Graphs can also be obtained depending on the "q" value, among other metrics. However, while graphs of "q" variables can be drawn for the DQN and Joint-DQN models, it is not possible to make a comparison since there is no "q" value in the PPO models. Results that are related to the "q" value are given in Figure 12, 13, 14, and 15. There are three graphs in these figures; these graphs are "q_min", "q_mean", and "q_max" values, respectively. In the DQN-RGB model without joints, two metrics except "q_min" increase smoothly, as seen in Figure 12. The "q_min" value increased until before the second collapse and then decreased. As seen in the Joint-DQN-RGB model in Figure 13, the "q_min" value increased smoothly despite the collapse, while the other two values decreased after the collapse. For the "q_min" value, in DQN models without or with joints, a certain amount of regression has been observed after each collapse, followed by an upward trend. The "q_min" value moves within a certain range between 110k and 150k. In the DQN-DEPTH model without joints, two values except "q_min" increase smoothly, as in Figure 14. This shows that the model gives the desired performance. The fact that the "q_min" value also tends to increase is a positive signal about the stability of the model. As seen in Figure 15, all values for the Joint-DQN-DEPTH model are rising. After the third quarter of the training, the "q_min" value began to increase. This suggests that the model could not complete the training process in the short time given it, but if it completes it, it can compete with other models. Likewise, other values have a similar trend to other models, but their maximum points are smaller.

In addition, the hardware used in the simulation is as follows:

- GPU: RTX 3090 1980 MHz
- CPU: Ryzen 9 5950X 16 Core 5.1 GHz (8 cores have been used due to heat problems)

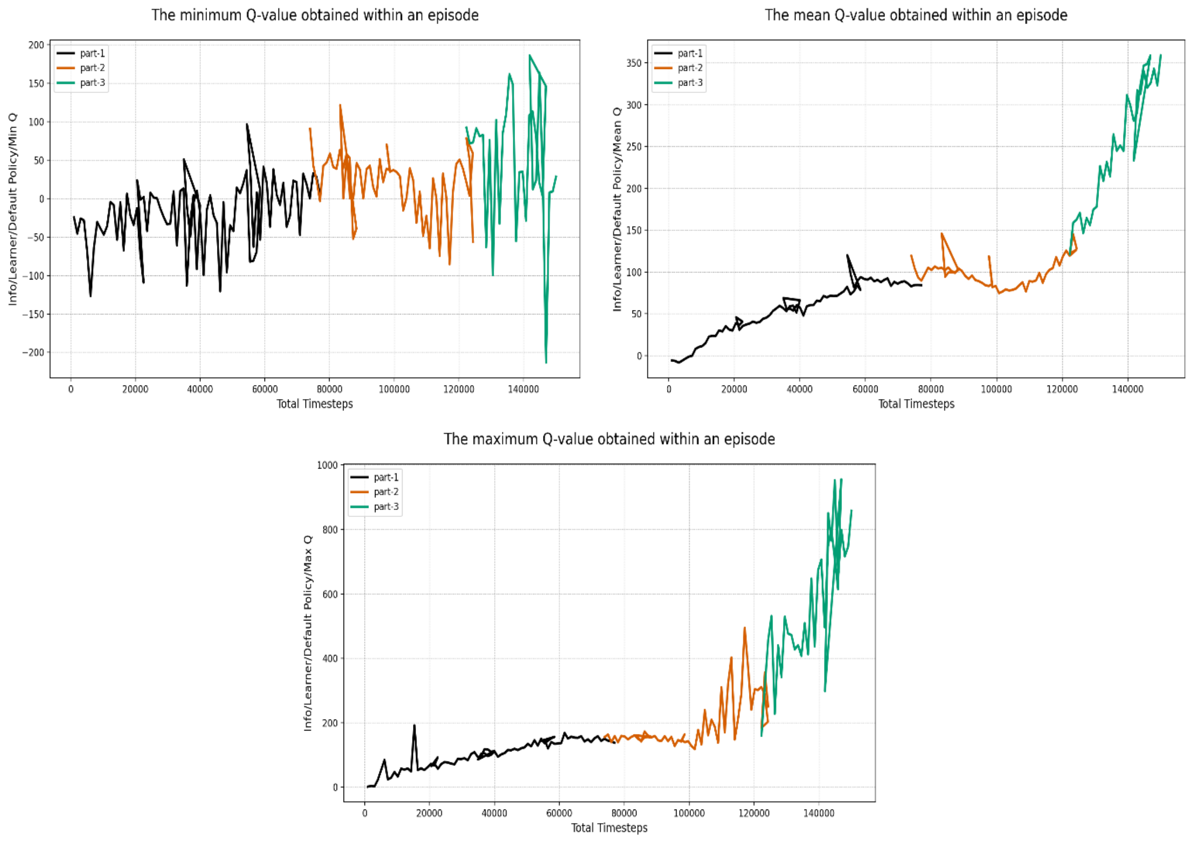

Figure 12. q values graphs for DQN-RGB without joint

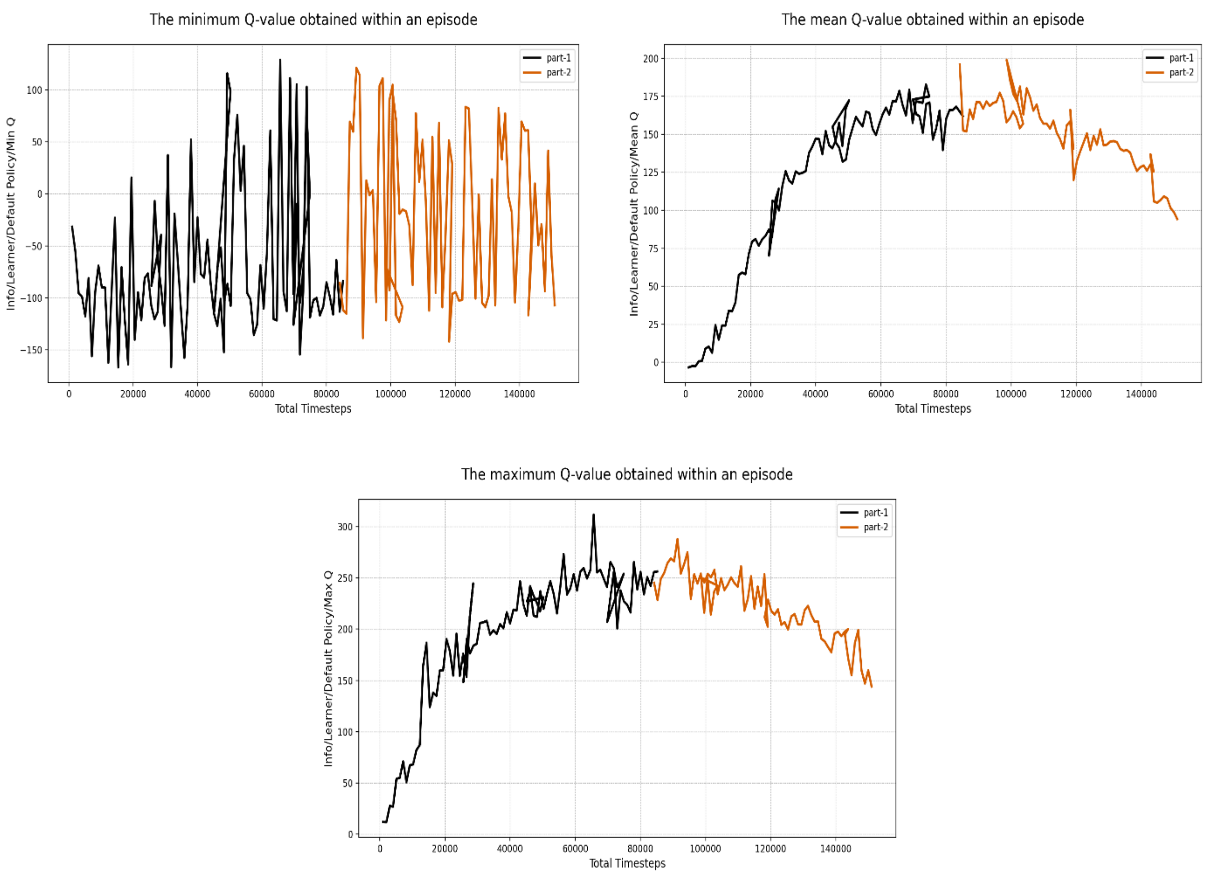

Figure 13. q values graphs for Joint-DQN-RGB

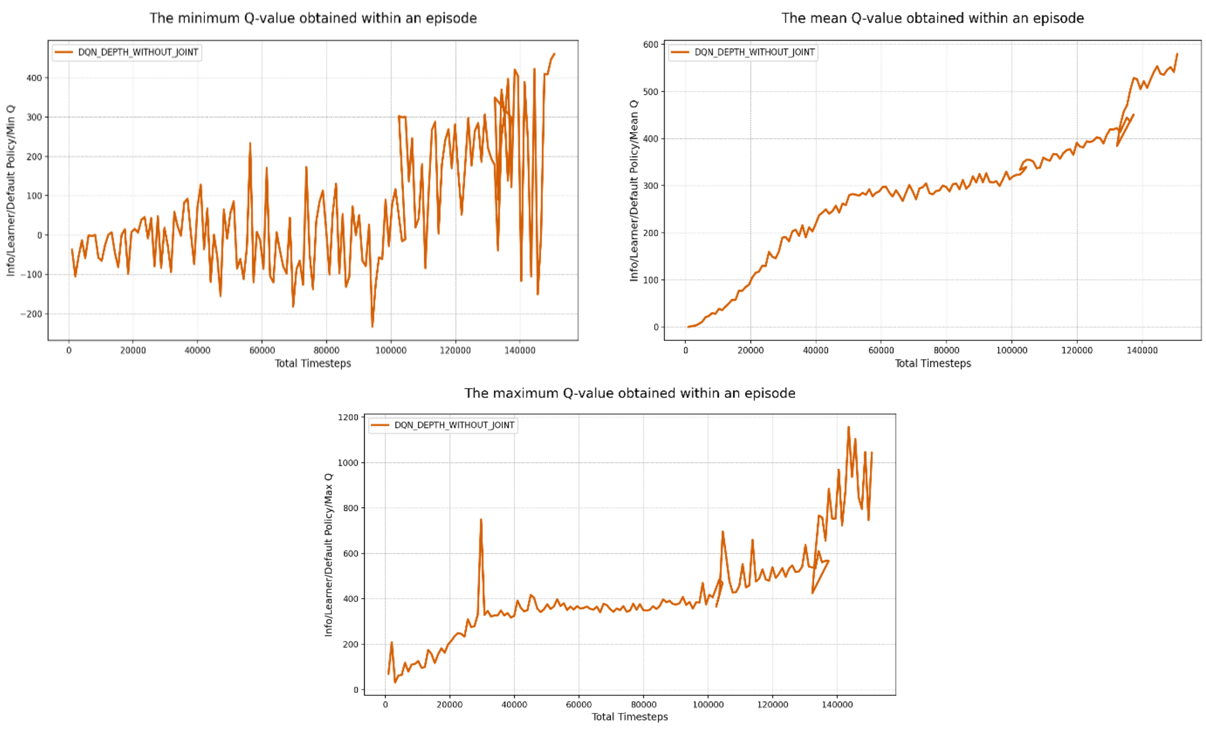

Figure 14. q values graphs for DQN- DEPTH without joint

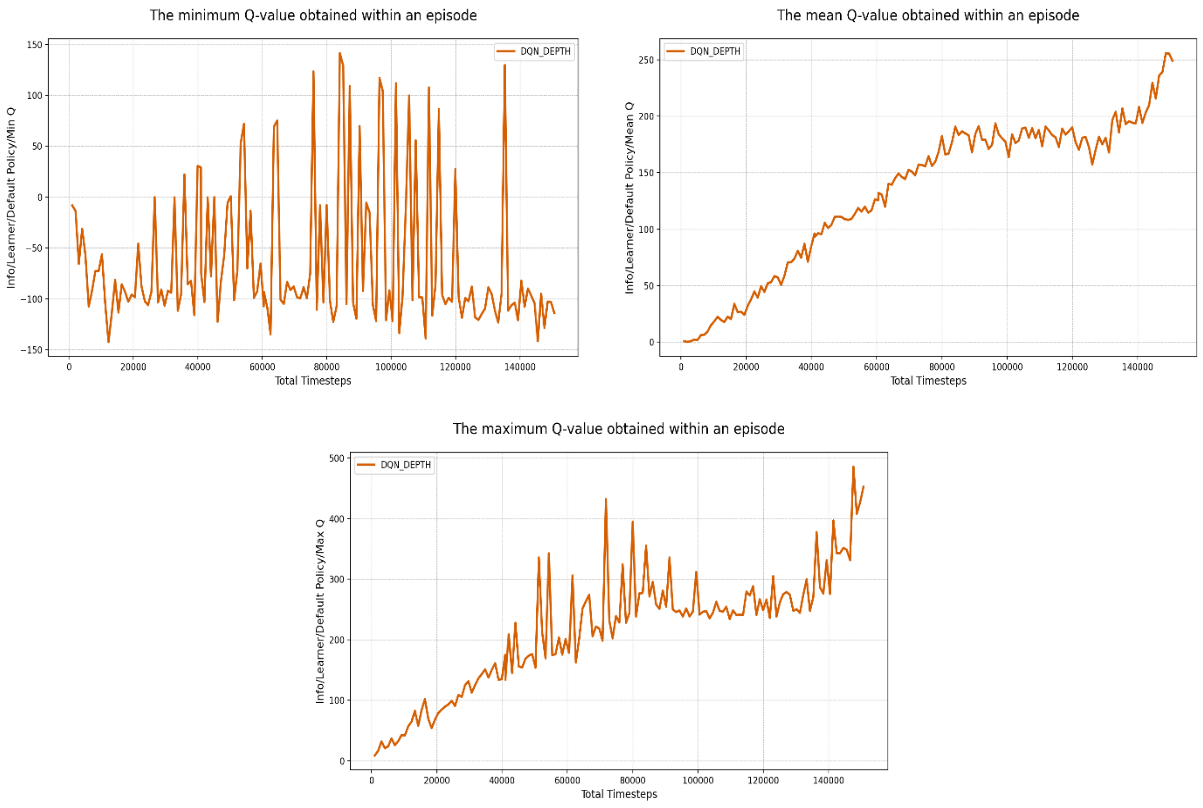

Figure 15. q values graphs for Joint-DQN-DEPTH

Summarized evaluations of the models used for a better understanding of their performance measurements are given in Table 1.

Table 1. Performance evaluation of the models

	DQN-RGB without joint	**Joint-DQN-RGB**	**PPO-RGB**	**DQN-DEPTH without joint**	**Joint-DQN-DEPTH**	**PPO-DEPTH**
Learning Speed	Identified as the fastest learning model.	Learning speed is slowed by the “Joint” architecture.	Significantly lags behind other models.	Identified as the other fastest learning model.	Learns slower compared to standard DQN.	Significantly lags behind other models.
Reward Trend	Shows a general upward trend despite system collapses.	Exhibits a dramatic drop after the collapse, except for “episode_reward_max”.	Exhibits a noisy and weak trend in the graphs.	Reward value increases rapidly as the number of steps decreases.	A steady increase is observed in reward values.	Reward metrics remained constant and negative, with a final collapse.
Episode Length Trend	The number of steps per episode decreases as the system matures.	Episode length decreases along with the drop in rewards.	Does not show a successful learning trend.	Demonstrates fast learning, dropping from ~200 to ~100 steps per episode.	Achieves the desired performance, as rewards increase while episode length decreases.	The increase in episode length (from 10 to 20) is not a positive sign.
System Stability (Collapses)	Experienced two collapses due to intensive RAM usage.	Experienced one collapse due to intensive RAM usage.	No collapses were experienced.	No collapses were experienced.	No collapses were experienced.	No collapses were experienced.
Post-Collapse Recovery	Recovers much faster after a collapse compared to Joint-DQN.	Highly sensitive to collapses with a slow recovery process.	Not Applicable (No collapses occurred).	Not Applicable (No collapses occurred).	Not Applicable (No collapses occurred).	Not Applicable (No collapses occurred).
Hardware Utilization (RAM)	Heavy RAM utilization led to system collapses.	Utilizes slightly more hardware resources than standard DQN.	Hardware usage is stable and not intensive.	Hardware usage is stable.	Slightly higher hardware utilization is observed.	Hardware usage is stable.
q-Value Stability	q_mean and q_max show stable increases, while q_min drops after the second collapse.	q_min value increases steadily even after the crash, while the other two values decrease.	Not Applicable (q-values not computed).	q_mean and q_max increase steadily, and q_min also shows an upward trend.	All q-values (min, mean, max) show an upward trend.	Not Applicable (q-values not computed).
Overall Performance Summary	A fast-learning model that achieves success with sufficient training but suffers from stability issues.	Learns slower but is a more decisive and efficient model in the long run. Highly sensitive to collapses.	Stated to be structurally unsuitable for this task, showing poor performance.	A stable and high-performing model that produces successful results when given enough training time.	Although it learns slower, it is a more decisive and effective model in the long run.	Clearly underperformed and failed due to incompatibility with the task.

4. Conclusion and Future Work

In this study, six different RL combinations have been tested for an autonomous drone performing a cargo delivery mission. Each combination has been trained for 150k steps and evaluated against various metrics. It has been observed that no matter which camera type or algorithm is used in the application carried out within the scope of the study, the algorithm can learn independently of the camera when enough time is given to learn. The algorithms used were compared for the first 150k steps for comparison purposes. If the results obtained are evaluated collectively; As expected, the fastest learner is the traditional DQN algorithm. It has been observed that the "Joint" structure reduces both the rewards received and the number of steps taken in the episode by half or even less than half. Even though the decrease in the reward may seem unfavorable, it is a very good situation to experience a decrease in the number of steps at the same rate. Because this means doing the same mission in a shorter time. System collapses have occurred only in the DQN-RGB and Joint-DQN-RGB models. After collapses, DQN-RGB has been recovered much faster than the structure containing joint. In fact, as a result of the collapses, the model containing up to 20k steps of joint has remained stagnant or fell further in most parameters. Joint-DQN is significantly more sensitive to collapses. This situation is exactly the same as with the depth camera. Only here, the positive and negative ends have been much more flexible. The "Joint" structure has slowed down learning, and it has taken longer for the model containing the joint to reach the level reached by the DQN without the joint layer. Furthermore, this model uses slightly more hardware power. But on the other hand, it has been worked on much more decisively, has been more successful in the long run, and has achieved the same result by making several times fewer steps. It has created a much more effective model, especially in the depth camera. Compared to other models, the PPO algorithm clearly lags behind other models. It has both learned slower and been less successful than others. When comparing the cameras, it has been noticed that there is not much difference between them. It has been observed that the RGB camera is ahead of the depth camera by a smaller margin, has more variables, and uses slightly more hardware (especially RAM).

In addition, better findings can be obtained by using different camera types or creating new algorithms, which is important if the study is to be further enhanced and for future studies. This situation can also be accomplished by creating an environment with more advanced features. Moreover, a training procedure that lasts for much longer periods of time, as opposed to a short amount of time, can produce more accurate findings and allow for greater system detail. Features can also be added for the drone, such as flight duration, charging time, customer satisfaction, and time limitations for picking up the cargo from the vehicle and delivering it to the customer. The system used can be more powerful in terms of RAM. Finally, by making the mentioned improvements and additions, future studies can focus on improving the ability to adapt to dynamic real-world environments.

5. Author Contribution Statement

The authors contributed equally to the article.

6. Ethics Committee Approval and Conflict of Interest

There is no need for an ethics committee approval in the prepared article. There is no conflict of interest with any person/institution in the prepared article.

7. Ethical Statement Regarding the Use of Artificial Intelligence

No artificial intelligence-based tools or applications were used in the preparation of this study. The entire content of the study was produced by the authors in accordance with scientific research methods and academic ethical principles.

8. References

[1] F. Wang, X. Zhu, Z. Zhou, and Y. Tang, “Deep-reinforcement-learning-based UAV autonomous navigation and collision avoidance in unknown environments,” Chin. J. Aeronaut., vol. 37, no. 3, pp. 237–257, 2024.

[2] J. Jagannath, A. Jagannath, S. Furman, and T. Gwin, “Deep learning and reinforcement learning for autonomous unmanned aerial systems: Roadmap for theory to deployment,” in Deep Learning for Unmanned Systems, A. Koubaa and A. T. Azar, Eds. Cham, Switzerland: Springer, 2021, pp. 25–82.

[3] A. Haque, N.-U.-R. Chowdhury, and M. S. Hossen, “Exploring the benefits of reinforcement learning for autonomous drone navigation and control,” Int. J. Adv. Netw. Appl., vol. 15, no. 1, pp. 5808–5814, 2023.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[5] C. Chen, Y. Zhang, Q. Lv, S. Wei, X. Wang, and X. Sun, “RRNet: A hybrid detector for object detection in drone-captured images,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW), Seoul, South Korea, 2019, pp. 100–108.

[6] D. Floreano, and R. J. Wood, “Science, technology and the future of small autonomous drones,” Nature, vol. 521, pp. 460–466, 2015.

[7] D. Abel et al., “Expressing non-Markov reward to a Markov agent,” Comput. Sci., 2022, Corpus ID: 253115270.

[8] L. Pawel, W. Lilian, K. Minwoo, and O. Hyondong, “Exploration in deep reinforcement learning: A survey,” Inf. Fusion, vol. 85, pp. 1–22, 2022.

[9] A. Ronca, G. P. Licks, and G. D. Giacomo, “Markov abstractions for PAC reinforcement learning in non-Markov decision processes,” arXiv:2205.01053, 2022.

[10] S. N. Yasar, and E. Karaköse, “Trajectory control of quadcopter in MATLAB simulation environment,” in Proc. Int. Conf. Decis. Aid Sci. Appl. (DASA), Chiangrai, Thailand, 2022, pp. 1127–1131.

[11] S. A. H. Mohsan, N.Q.H. Othman, Y. Li, M.H. Alsharif, and M.A. Khan, “Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends,” Intell. Serv. Robot., vol. 16, pp. 109–137, 2023.

[12] D. Hong, S. Lee, Y.H. Cho, D. Baek, J. Kim, and N. Chang, “Energy-efficient online path planning of multiple drones using reinforcement learning,” IEEE Trans. Veh. Technol., vol. 70, no. 10, pp. 9725–9740, 2021.

[13] K. Nagami, and M. Schwager, “HJB-RL: Initializing reinforcement learning with optimal control policies applied to autonomous drone racing,” in Proc. Robot.: Sci. Syst., 2021.

[14] E. Cetin, C. Barrado, G. Muñoz, M. Macias, and E. Pastor, “Drone navigation and avoidance of obstacles through deep reinforcement learning,” in Proc. IEEE/AIAA Digit. Avion. Syst. Conf. (DASC), San Diego, CA, USA, 2019, pp. 1–7.

[15] M. A. B. Abbass, and H. S. Kang, “Drone elevation control based on Python-Unity integrated framework for reinforcement learning applications,” Drones, vol. 7, no. 4, p. 225, 2023..

[16] D. Gozen, and S. Ozer, “Visual object tracking in drone images with deep reinforcement learning,” in Proc. Int. Conf. Pattern Recognit. (ICPR), Milan, Italy, 2021, pp. 10082–10089.

[17] D. K. Kim, and T. Chen, “Deep neural network for real-time autonomous indoor navigation,” arXiv:1511.04668, 2015.

[18] T. Guo, N. Jiang, B. Li, X. Zhu, Y. Wang, and W. Du, “UAV navigation in high dynamic environments: A deep reinforcement learning approach,” Chin. J. Aeronaut., vol. 34, no. 2, pp. 479–489, 2021.

[19] E. Karaköse, “Coordination of multi UAV’s equipped with IoT,” in Proc. Int. Conf. Adv. Technol., Antalya, Türkiye, 2018, pp. 169–172.

[20] C. Wang, J. Wang, J. Wang, and X. Zhang, “Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards,” IEEE Internet Things J., vol. 7, no. 7, pp. 6180–6190, 2020.

[21] S. Y. Shin, Y. W. Kang, and Y. G. Kim, “Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot,” Appl. Sci., vol. 9, no. 24, p. 5571, 2019.

[22] A. K. Tiwari, and S. V. Nadimpalli, “Augmented random search for quadcopter control: An alternative to reinforcement learning,” arXiv:1911.12553 [cs.LG], 2019.
[23] G. Munoz, C. Barrado, E. Çetin, and E. Salami, “Deep reinforcement learning for drone delivery,” Drones, vol. 3, no. 3, p. 72, 2019.
[24] Y. Chen, W. Zheng, Y. Zhao, T.H. Song, and H. Shin, “DW-YOLO: An efficient object detector for drones and self-driving vehicles,” Arab. J. Sci. Eng., vol. 48, pp. 1427–1436, 2023.
[25] F. Ahmed, J. C. Mohanta, A. Keshari, and P.S.Yadav, “Recent advances in unmanned aerial vehicles: A review,” Arab. J. Sci. Eng., vol. 47, pp. 7963–7984, 2022.
[26] T. Thomas, S. Srinivas, and C. Rajendran, “Collaborative truck multi-drone delivery system considering drone scheduling and en route operations,” Ann. Oper. Res., Jun. 2023.
[27] Z. Bi, X. Guo, J. Wang, S. Qin, and G. Liu, “Deep reinforcement learning for truck-drone delivery problem,” Drones, vol. 7, no. 7, p. 445, 2023.
[28] A. F. U. Din, I. Mir, F. Gul, M. R. A. Nasar, and L. Abualigah, “Reinforced learning-based robust control design for unmanned aerial vehicle,” Arab. J. Sci. Eng., vol. 48, pp. 1221–1236.
[29] M. Yilmazer, E. Karakose, and M. Karakose, “Multi-package delivery optimization with drone,” in Proc. Int. Conf. Data Anal. Bus. Ind. (ICDABI), Sakheer, Bahrain, 2021, pp. 65–69.
[30] E. Karaköse, “A new last mile delivery approach for the hybrid truck multi-drone problem using a genetic algorithm,” Appl. Sci., vol. 14, no. 2, p. 616, 2024