Joint Speed Control and Energy Replenishment Optimization for UAV-assisted IoT Data Collection with Deep Reinforcement Transfer Learning

Nam H. Chu, Dinh Thai Hoang, Diep N. Nguyen, Nguyen Van Huynh, and Eryk Dutkiewicz

Abstract—Unmanned aerial vehicle (UAV)-assisted data collection has been emerging as a prominent application due to its flexibility, mobility, and low operational cost. However, under the dynamic and uncertainty of IoT data collection and energy replenishment processes, optimizing the performance for UAV collectors is a very challenging task. Thus, this paper introduces a novel framework that jointly optimizes the flying speed and energy replenishment for each UAV to significantly improve the overall system performance (e.g., data collection and energy usage efficiency). Specifically, we first develop a Markov decision process to help the UAV automatically and dynamically make optimal decisions under the dynamics and uncertainties of the environment. Although traditional reinforcement learning algorithms such as Q-learning and deep Q-learning can help the UAV to obtain the optimal policy, they often take a long time to converge and require high computational complexity. Therefore, it is impractical to deploy these conventional methods on UAVs with limited computing capacity and energy resource. To that end, we develop advanced transfer learning techniques that allow UAVs to "share" and "transfer" learning knowledge, thereby reducing the learning time as well as significantly improving learning quality. Extensive simulations demonstrate that our proposed solution can improve the average data collection performance of the system up to 200% and reduce the convergence time up to 50% compared with those of conventional methods.

Index Terms—UAV, IoT data collection, Markov decision process, deep reinforcement learning, transfer learning.

I. INTRODUCTION

Over the last ten years, the Internet of Things (IoT) has been rapidly evolving to meet ever-increasing demands in industries and many aspects of our daily lives. According to [2], the IoT's share in wireless connections will increase from 33% in 2018 to 50% by 2023, which is equivalent to 14.7 billion connections. Consequently, the IoT system scale has been growing exponentially, posing various challenges to service providers and network operators. First, the uneven distribution of IoT devices over large areas makes traditional wireless Access Points (APs) (e.g., WiFi APs) inefficient in collecting IoT data. Second, traditional APs are even unable

to collect IoT data in various situations. For example, IoT devices are distributed at hard-to-reach locations such as on top of trees, outside high-rise buildings, or even under bridges. In this case, cellular base stations can act as data collectors in IoT data collection networks. However, the limited energy storage capacity and communication capability hinder IoT devices from transmitting their data over long distances. Thus, IoT data collection networks have been demanding effective and flexible solutions for data acquisition.

Recently, UAVs have been emerging as a promising approach to tackle the above challenges. In particular, when UAVs act as on-demand flying APs, thanks to their aerial superiority, they can establish good line-of-sight (LoS) links for the IoT nodes. This leads to better wireless communication channels, and thereby improving the quality of service (QoS) in comparison with traditional approaches, especially for IoT applications that are sensitive to delay and/or require stable communications [3]. In addition, in remote areas without access to terrestrial infrastructures, UAVs can provide a much more economic solution to collect IoT data than traditional approaches, e.g., long-range ground broadcasting stations or high-cost satellite communications. Another important advantage of UAVs is that they can be promptly established in emergency circumstances, in which the existing infrastructure is disrupted and incapable of receiving data from IoT devices [4]. Due to the flexibility, mobility, and low operational cost, UAVs have been being deployed as flying APs for some real-world projects, e.g., Google's Loon and Facebook's Aquila [5], [6].

However, there are still some challenges that hinder the applications of UAVs in IoT data collection networks. In particular, unlike traditional solutions for collecting IoT data (e.g., deploying fixed APs), UAVs have limited energy resources supplied by batteries. Thus, some UAVs may not have enough computing capacity to deploy advanced techniques, e.g., a deep learning-based methods consisting a deep neural network, to optimize their operations. When the UAVs' batteries are depleted, they must replenish their energy by flying back to the charging stations to charge or replace their batteries. It is worth noting that given a fixed working duration, the more time the energy replenishment process takes, the less time the UAVs can spend for collecting IoT data. Alternatively, the energy replenishment process is highly dynamic since it depends on the distance between the UAV and the charging stations. Therefore, optimizing energy usage and the energy replenishment process is critical to achieving high system performance, but very challenging in practice. Moreover, the

UAVs often fly around to collect IoT data, while IoT nodes are statically allocated over different zones, and their sensing data are random depending on surrounding environments. To that end, optimizing operations of UAVs in different zones to maximize data collection efficiency is another major challenge that needs to be considered. To address the aforementioned problems, jointly optimizing the UAV's speed and energy replenishment process is an effective approach to maximize data collection efficiency and energy usage efficiency.

A. Related Works

1) UAV's Energy Replenishment Process: In [7]–[11], the authors consider a UAV-assisted IoT data collection system where a charging station is deployed to prolong UAV serving time. To minimize the age of information (AoI) under the constraint of the UAV's charging rate and battery capacity, the authors in [7] propose a least-charging-timed Metropolis-Hasting trajectory and a least-visit-time-based trajectory. They also point out that the UAV's charging rate has much more influence on the low bound of AoI than the UAV's battery capacity. The study in [8] aims to minimize the data collection time by optimizing the UAV trajectory and the order of IoT devices that the UAV is going to visit. Specifically, they employ a deep deterministic policy gradient-based algorithm to find a route between two positions, and a Q-learning based scheduler to determine the order of visiting positions where the IoT devices or charging station are located. Furthermore, a transfer learning model is introduced to speed up the training process. However, the effectiveness of the proposed transfer learning technique is not well investigated. Similarly, the study in [9] aims to minimize the total time that the UAVs need to collect data from backscatter sensor nodes. If the UAVs' remaining energy is insufficient to complete the task, they can return to a charging station for charging. The authors first use the Gaussian mixture model to group IoT nodes into different clusters and formulate the trajectory optimization problem as a semi-Markov decision process. Then, deep reinforcement learning (DRL) approaches are proposed to find the optimal policy for the UAV.

In [10], the authors consider that a UAV is wirelessly being charged during the data collection task. They first formulate the problem as a Markov decision process (MDP), then propose a Q-learning algorithm to maximize the energy efficiency and system throughput. In [11], the authors propose a blockchain-enabled UAV-assisted framework to provide security for IoT data collection network. A charging coin is introduced to reward UAVs when they successfully collect IoT data. Then, the UAVs can use collected coins to recharge their batteries at a charging station. In addition, they develop an adaptive linear prediction model to reduce the number of transactions in the system, resulting in a decrease in energy consumption.

All of the above works assume that the UAVs always fly at a constant speed during the data collection process. However, in practice, a UAV can choose different speeds during its data collection process depending on its surrounding environment. Alternatively, the UAV's speed can strongly influence the

system's efficiency because it has a substantial impact on energy consumption during the data collection process [12]. Thus, optimally controlling UAVs' speed can significantly improve the energy usage and data collection efficiency of the system, especially in UAV-assisted IoT data collection networks where UAVs have limited battery capacities. Unfortunately, this important factor is not investigated in all the above studies.

2) Speed Control for UAVs: In the literature, only a few works investigate the speed control problem for UAV-assisted IoT data collection networks [13]–[16]. In [13], the authors aim to minimize the flight time for a data collection task by jointly optimizing the UAV's speed, data collection duration, and the IoT devices' transmit power. Their numerical results show that the UAV's optimal speed depends on the distance between sensors, sensors' energy, and the data upload requirements. In [14] and [15], the authors aim to maximize the data collection efficiency by controlling UAV's speed according to the IoT device density. In particular, the authors in [14] first introduce an analytical model for the transmission between the UAV and IoT nodes, then the UAV's speed is optimized based on this model. In [15], the authors reveal a tradeoff between system throughput and IoT devices' energy efficiency. By optimizing the UAV's speed, altitude, as well as the MAC layer frame length, we can achieve the balance between the two conflict factors.

All the above works (i.e., [13]–[15]) apply the conventional optimization theories, which statically optimize the UAV's speed during the IoT data collection process. Therefore, their algorithms need to rerun whenever the environments are changed, leading to a high computational complexity. As a result, optimization-based solutions are inefficient in addressing the high dimensional state space as that in the considered problem. More importantly, they cannot be used in scenarios in which the complete information about the surrounding environment is unknown, like what we consider in this work (e.g., packet arrival probabilities for the whole network and flight time for replacing the battery). In this context, reinforcement learning emerges as the best approach to address the highly dynamic and uncertainty of the environment since it can help the UAV adapt its behavior according to the environment's changes. In [16], the authors employ deep Q-learning to control the UAV's speed during its data collection task, where the UAV can also wirelessly charge the IoT devices while collecting their data. This work aims to minimize the data packet loss by selecting the best devices to be charged and interrogated, together with the optimal UAV's speed. Their simulation results show that the UAV's speed is proportional to the number of IoT devices and inversely proportional to the data queue lengths of IoT devices. Similar to [13]–[15], the study in [16] does not consider the impacts of UAV's energy consumption and energy replenishment processes during the data collection task. It is worth highlighting that the energy replenishment process is a critical factor that cannot be ignored since the UAVs' energy is limited.

It can be observed that all of the aforementioned works do not jointly optimize energy replenishment and speed control activities simultaneously. However, they are among the most

important factors to achieve high efficiency in terms of energy and data collection for UAV-assisted IoT data collection networks. In addition, these proposed approaches require a high computational complexity that may be inefficient to deploy on UAVs. Moreover, most RL-based optimal operation policies for UAV-based collectors (i.e., [8]–[10], [16]) rely on conventional Q-learning or deep Q-learning based algorithms that are prone to overestimating state-action values. This problem can make the learning process unstable [19]. In addition, the work in [8] applies transfer learning to speed up the learning process of the proposed DRL algorithm. However, the impacts of transfer learning are not well investigated. Note that transfer learning does not always improve or even can cause negative impacts on the learning process [17]. Furthermore, the study in [8] does not consider the UAV's speed control, one of the most important factors influencing to decisions of energy replenishment. To fill these gaps, this study develops a highly efficient solution based on deep reinforcement transfer learning for UAV-assisted IoT data collection networks. Specifically, our proposed approach can effectively address the overestimation problem and stabilize the learning process by adopting recent advanced techniques in RL, including deep Q-learning [18], deep double Q-learning [19], and dueling neural network architecture [20]. In addition, the proposed solution can simultaneously optimize the UAV's speed and energy replenishment processes and allow the learned knowledge to be effectively "shared" and "transferred" between UAVs. Thus, leveraging the transfer learning technique can improve the learning quality and reduce the learning time, thereby leading to a decrease of computational complexity. Therefore, our proposed solution can be effectively implemented on UAVs.

B. Contributions

Given the above, to jointly optimize the speed control and battery replacement activities for a considered UAV under the dynamic and uncertainty of IoT data collection process, we propose a dynamic decision solution leveraging the Markov decision process (MDP) framework. This framework allows the UAV to make optimal decisions (regarding the flying speed and battery replacement activities) based on its current observations about the surrounding environment. Although Q-learning can be used to find the optimal policy for the UAV, its convergence rate is slow, especially in a highly complex problem as the one considered in this paper where we need to jointly optimize the speed and energy replenishment activities for the UAV. In addition, Q-learning algorithms usually suffer from overestimation problems when estimating action values, especially for complicated problems with hybrid actions like what we consider in this paper (i.e., speed selection and energy replenishment actions) [19]. Thus, we develop a highly-effective Deep Dueling Double Q-learning (D3QL) to address these challenges. The key ideas of D3QL are to (1) separately and simultaneously estimate the state values and action advantages, making the learning process more stable [20], and (2) address the overestimation by using two estimators (e.g., deep neural networks), resulting in the stability of estimating action values. To further reduce the learning time and enhance

the learning quality, we develop transfer learning techniques to allow the UAV to learn more knowledge from other UAVs learning in similar environments. In addition, these techniques also help the UAV leverage knowledge obtained from different environments to improve its policy, making our solution more applicable and scalable. Therefore, our proposed solution can be deploy on resource-constrained devices, e.g., UAVs. Extensive simulation results demonstrate that our proposed solution, i.e., D3QL with transfer learning (D3QL-TL), can simultaneously optimize the energy usage and data collection, and thereby leading to the best performance compared to other methods. To the best of our knowledge, this is the first study investigating a UAV operation control approach taking the dynamic of the IoT data collection, energy limitation, and impact of the energy replenishment process into considerations. Our major contributions can be summarized as follows.

- Propose a novel framework that allows the UAV to jointly optimize its flying speed and battery replacement activities under the dynamic and uncertainty of data collection and energy replenishment processes. In addition, this framework can not only allow the UAV to dynamically and automatically make optimal decisions through real-time interactions with the surrounding environment but also enable the "share/transfer" learning knowledge among UAVs working in the same and/or similar environments.
- Develop a highly-effective DRL algorithm leveraging recent advances of deep Q-learning, deep double Q-learning, and dueling neural network architecture to stabilize the learning phase, thereby quickly obtaining an optimal operation policy for the UAV.
- Develop advanced transfer learning techniques that allow UAVs to "share" and "transfer" their learning knowledge, thereby reducing the learning time and improving learning quality for the UAV. In addition, these techniques help UAVs to utilize the knowledge and information learned from different environments, making our approach more scalable and applicable in practice, e.g., scenarios with multiple UAVs.
- Perform extensive simulations to demonstrate the efficiency of our proposed approaches and reveal critical elements that can significantly impact on the performance of UAV-assisted IoT data collection networks.

The rest of this paper is structured as follows. The system model and operation control formulation are described in Section II and III, respectively. Section IV presents the proposed D3QL and D3QL-TL algorithms. Then, the simulation results are analyzed in Section V. Finally, Section VI concludes our paper.

II. SYSTEM MODEL

In this work, we consider a UAV-assisted IoT data collection system where a UAV is deployed to collect IoT data over a considered area, as illustrated in Fig. 1. We assume that the considered area is divided into N zones. The IoT devices are distributed randomly over these zones to execute various tasks, e.g., sensing temperature and humidity. In practice, the
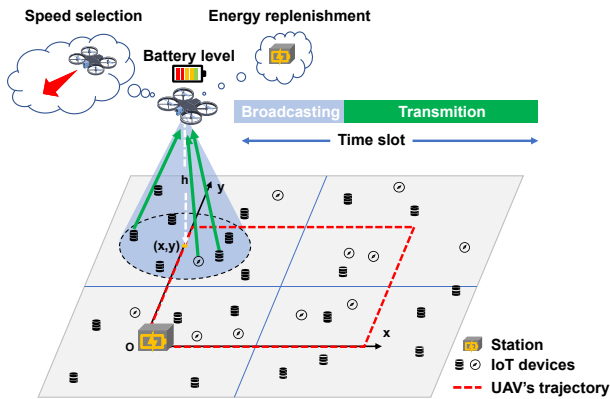
Fig. 1: System model for UAV-assisted IoT data collection network.

numbers of IoT nodes in these zones are different due to different sensing demands in these zones. We consider time to be slotted (as in [14], [15]) with an equal duration, and each time slot is split into two consecutive intervals, broadcasting and transmission, respectively. In the broadcasting interval, the UAV uses a dedicated channel to broadcast a wake-up signal [21] to all IoT nodes in its communication range. After acquiring this signal, these nodes will send their data to the UAV during the transmission interval. We assume that the communication link from IoT devices to the UAV adopts the OFDMA technique, while the communication link from the UAV to IoT devices uses the OFDM technique, as in [22]. In this way, the IoT devices can simultaneously transmit data to the UAV. Let p_n denote the probability of a data packet successfully collected by the UAV in a time slot in zone n. Because the IoT nodes are distributed unevenly over N zones, p_n may vary over these zones.

As considering a UAV-assisted IoT data collection network, we can assume that the UAV flies at a fixed altitude h (similar as that of [4], [13], [14], [22], [23]) and follows a predefined trajectory to sweep through all IoT devices of the system in each round (as in [13]–[15]). In this paper, unlike [13]–[15], we consider a more realistic scenario where the UAV is equipped with a battery that has limited energy storage. It is worth mentioning that the energy consumption for the wireless data collection process (i.e., broadcasting the wake-up signal and receiving data packets) is much lower than that of the flying operation [12]. In addition, the decision of the UAV at each time slot (i.e., speed selecting or returning for energy replenishment) does not influence the power consumption for the wireless data collection process. Thus, our model can straightforwardly capture the communication power consumption by adding a constant to the UAV's energy consumption at each time slot, similar to that in [10]. To that end, in this work, we only focus on optimizing energy consumption for the UAV's flying operation, similar as that in [7], [12]. In a time slot, we assume that the UAV's velocity is constant (as in [13], [22]), but in different time slots the UAV can choose to fly at different speeds, e.g., $v_w = \{v_1, \ldots, v_A\}$. Each speed may cost a different amount of energy. For example, if the UAV flies faster, it may use more energy per time slot [12].

Note that UAVs in UAV-assisted IoT data collection systems often fly at a low speed to maintain the reliability of the data collection process.

When the UAV's energy is depleted, it will fly back to the charging station placed at a fixed location to replace the battery, as illustrated in Fig. 1. Furthermore, during the flight, the UAV can decide to go back to the charging station to change the battery, for example, when it is near the charging station and its energy level is low. Once the energy replenishment process is accomplished, the UAV will fly back to its trajectory and continue its task. Here, we consider that the UAV has a maximum of E energy units for its operation (i.e., flying to collect IoT data) and a backup energy storage for flying back to the charging station for battery replacement. During the energy replenishment process, including back-and-forth flights and battery replacement, the UAV cannot collect data. Suppose that it takes the UAV t_f and t_b time slots to fly from its current location to the station and to replace the battery, respectively. Indeed, the battery replacement time, i.e., t_b, may be known in advance, while the return flight time, i.e., t_f, is highly dynamic depending on the distance between UAV's current location and the charging station. In addition, t_f also depends on the return speed of the UAV, denoted by v_r. Assuming that UAV flies with a constant speed when returning to the station, the duration of energy replenishment is calculated by the equation $t_e = 2t_f + t_b$. Therefore, the energy replenishment process is also dynamic due to the dynamic of flying time t_f.

In practice, the surrounding environment is highly dynamic and uncertain. Specifically, the UAV does not know the probabilities of receiving a packet in different areas in advance, as they are very uncertain depending on sensing tasks. It is important to note that the UAV may collect more data when moving in a zone with a high probability of receiving packets, i.e., a high value of p_n. As a result, to maximize the data collection efficiency, the UAV must gradually learn this knowledge in order to adapt its operations accordingly, e.g., flying speed and energy level status. Moreover, the returning flight time depends on the distance between the UAV's current position and the station, which is highly dynamic. Therefore, if the UAV appropriately decides when to return for battery replacement (e.g., when it is near the station and its energy level is low), the energy replenishment time will be reduced significantly, resulting in high system performance. In contrast, if the UAV goes back to replace its battery when its energy level is high and it is far from the station, it will waste both time and energy, leading to low system performance. Thus, optimizing the UAV's operations to maximize the long-term system performance is a very challenging task. In the following sections, we will present our proposed learning algorithms that can effectively and quickly obtain the UAV's optimal operation policy under the limited energy of the UAV and the uncertainty of the data collection process.

III. OPTIMAL OPERATION CONTROL FORMULATION

To overcome the uncertainty and highly dynamic of the data collection and energy replenishment processes under the limited energy storage of the UAV, we formulate the

UAV's operation control problem as the Markov decision process (MDP) framework. Typically, an MDP is determined by three components, state space $\mathcal{S}$, action space $\mathcal{A}$, and immediate reward function r. Based on the MDP framework, at each time slot the UAV can dynamically make the best actions (e.g., flying at appropriate speeds or return for battery replacement) based on its current observations (i.e., its location and energy level) to maximize its long-term average reward without requiring complete information about data collection and energy replenishment processes in advance.

A. State Space

In this work, we aim to maximize the efficiency of collecting data and energy usage efficiency, and thus there are some important factors which we need to take into considerations. The first important factor is the current location of the UAV. The main reason is that the UAV's location can reveal important information about the expected amount of data that can be collected by the UAV and the time it takes if the UAV chooses to fly back to the station for battery replacement. Specifically, the UAV will likely collect more data when moving in a zone with a high probability of receiving data than in a zone with a low probability of receiving data. In addition, the farther the distance between the UAV and the station is, the more time it takes to travel between these two positions. As mentioned above, the UAV always flies at a fixed altitude so that the UAV's position can be given by its 2D projection on the ground, i.e., (x, y) coordinates. The second crucial factor is the current UAV's energy level, denoted by e, which affects the decision of the UAV at every time slot. For example, the UAV should not select the battery replacement action (i.e., return the station to replace the battery) unless its energy level is low. Otherwise, it may waste time and energy for the flying back trip. To that end, this information is embedded into the state space of the UAV, which can be defined as follow:

$$\mathcal{S} = \Big\{ (x, y, e) : x \in \{0, \ldots, X\}; y \in \{0, \ldots, Y\}; \\ \text{and } e \in \{0, \ldots, E\} \Big\} \cup \{(-1, -1, -1)\}, \quad (1)$$

where X and Y are the maximum x and y coordinates of the UAV's trajectory, and E is the maximum energy capacity of the UAV. As a result, the system state can be indicated by a tuple $s = (x, y, e) \in \mathcal{S}$. Moreover, because of the energy replenishment process, it is necessary to introduce a special state, i.e., $s = (-1, -1, -1)$. This special state is only visited when UAV's energy is depleted (i.e., UAV's state is $(x, y, 0)$) or if the UAV selects the battery replacement action. Then, after the energy replenishment process completes, the UAV will return to the previous position (where it decided to go back for battery replacement or where its energy is depleted) with a full battery, i.e., $s = (x, y, E)$. This design ensures that the system process is continuous, i.e., no terminal state.

B. Action Space

During the operation, to maximize the system performance in terms of energy usage and data collection efficiency, the UAV needs to not only choose the most suitable flying speed but also decide when to go back to the station to replace the battery. It is worth mentioning that given different states, the action spaces for these states may be different. For example, at a non-working state, i.e., $s = (-1, -1, -1)$, the UAV cannot select a flying speed. Instead, the possible action at this state is to stay "idle" until the UAV returns to its trajectory with a full battery. In other words, the UAV will stay at the non-working state after performing an "idle" action until the energy replenishment process completes. As a result, we can define the action space for the UAV as follows:

$$\mathcal{A} \triangleq \{a : a \in \{-1, 0, 1, \ldots, A\}\}, \quad (2)$$

where action $a = -1$ is to indicate the "idle" action and action $a = 0$ is to express that the UAV will choose to return to the station for replacing the battery, namely battery replacement action. Actions $a = \{1, \ldots, A\}$ are to represent the speed level that the UAV selects to fly at the current time slot. In addition, given the state $s \in \mathcal{S}$ the action space based on state s, i.e., $\mathcal{A}_S$, consists of all possible actions that are feasible at this state. Thus, we can express $\mathcal{A}_S$ as follows:

$$\mathcal{A}_S = \begin{cases} \{-1\}, & \text{if } s = (-1, -1, -1), \\ \{0, \ldots, A\}, & \text{otherwise.} \end{cases} \quad (3)$$

C. Reward Function

As discussed above, two main actions (i.e., flying speed and battery replacement actions) have significant effects on the system performance. Specifically, choosing an appropriate flying speed at each time slot can maximize the efficiency of the collecting data process as well as energy usage. Alternatively, selecting the right time to return for battery replacement can reduce the energy replenishment time, thereby improving the overall system performance. For example, when the UAV is flying near the charging station and its energy is low, it should return to the charging station for battery replacement. Therefore, our proposed immediate reward function consists of (1) speed selection reward function, i.e., r_t^a, and (2) battery replacement reward function, i.e., r_t^b, as follows:

$$r_t(s_t, a_t) = \begin{cases} r_t^a(s_t, a_t), & \text{if } a_t \in \{\mathcal{A} \setminus \{0\}\}, \\ r_t^b(s_t, a_t), & \text{if } a_t = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where a_t is the selected action at time t.

1) Speed Selection Reward Function: Since our objective is to maximize the system performance by striking a balance between the data collection efficiency and energy usage efficiency, the speed selection reward function needs to capture this information. We define data collection efficiency as the number of collected data packets over a time slot and operation status of the UAV. For example, given a time slot, if the UAV is moving to collect data, it will receive a working reward, denoted by $\Omega > 0$. Otherwise, it will not receive the working reward, i.e., $\Omega = 0$. In this way, the working reward will encourage the UAV to collect data rather than return and then wait at the station for the battery replacement. The energy usage efficiency can be represented by the cost of choosing a flying speed, i.e., energy consumed by the UAV to fly at speed a during a time slot t. Clearly, the selected speed determines
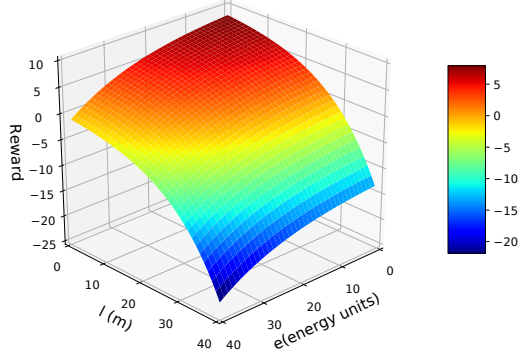
Fig. 2: An example of the proposed battery replacement reward function.

the energy consumption per time slot of the UAV, e.g., a low speed will cost the UAV less energy per time slot than that of a high speed [12]. At time slot t, the cost of performing action a is denoted by m_t^a, and the number of collected data packets at the current state s is denoted by d_t^s. Thus, the speed selection reward function can be expressed by:

$$r_t^a(s_t, a_t) = \Omega + w_1 d_t^s - w_2 m_t^a, \quad \text{if } a_t \in \mathcal{A} \setminus \{0\}, \quad (5)$$

where w_1 and w_2 are the weights to balancing between collected data and energy consumption of the UAV. It is worth noting that these weights can be defined in advance based on the service provider's requirements. For example, in case if the data is more important and valuable than energy, we can set the value of w_1 to be higher than that of w_2. In contrast, if the energy is scarce, we can set the value of w_1 to be lower than that of w_2. Therefore, the speed selection reward function can capture the UAV's data collection efficiency and energy consumption efficiency.

2) Battery Replacement Reward Function: Although the speed selection reward function gives the UAV sufficient information to learn the optimal speed control, it cannot help the UAV learn when it is good to return to the station for the battery replacement. First, if the UAV performs the battery replacement action, the values of Ω and d_t^s in (5) will be zero, leading to a negative value of r_t^a. Consequently, the UAV may consider this action as a bad choice and will not choose it in the future. Second, the speed selection reward function is unable to guide the UAV to learn where is good to return for replacing its battery, e.g., the further from the station the UAV is, the smaller reward for battery replacement action it may receive. Therefore, battery replacement action needs a different reward function, which needs to take into account of both the UAV's current energy level, i.e., e, and the distance between its current position and the station, i.e., l. However, in practice, it is challenging to design such a reward function for battery replacement action because the complex relationship between e and l makes more difficulties for the UAV to decide whether it should go back or keep flying to collect data. In particular, the UAV may choose the battery replacement action if both e and l are small, while the UAV should continue its collection task if any of these factors is large. To that end,

in the following, we propose a reward function for battery replacement action that can address this problem, and to the best of our knowledge, this is the first work in the literature addressing the battery replacement problem for UAV-assisted IoT data collection networks.

Suppose the UAV decides to return for battery replacement at time t, at which its energy and the distance to the station are e_t and l_t, respectively. Then, its immediate reward is derived from the battery replacement reward function as follows:

$$r_t^b(s_t, a_t) = c - w_3 \exp(w_4 e_t) - w_5 \exp(w_6 l_t), \quad \text{if } a_t = 0, \quad (6)$$

where c is a constant controlling the maximum value of r_t^b, which may affect the learning policy of the UAV. For example, if c is smaller than the smallest value of r_t^a, the value of returning reward function is always lower than that of the speed selection reward function, making the return action always "worse" than those of the speed selection actions. The second and third terms in (6) express the influence of the energy level e_t and the distance l_t to the UAV's decision for battery replacement. The tradeoff between the energy level e_t and the current distance l_t is controlled by four weights, i.e., w_3, w_4, w_5, w_6. Fig. 2 demonstrates an example of the proposed battery replacement reward function in which $c = 10$, $w_3 = w_5 = 1$, $w_4 = 0.06$, and $w_6 = 0.08$. It can be observed that as e and l are large (e.g., greater than 36 and 17, respectively), the UAV will receive negative rewards, meaning that the UAV is not encouraged to return to replace its battery if its energy is high or it is far from the station. In this way, this function will encourage the UAV to return to the station for replacing the battery when its current energy and distance to the station are small.

D. Optimization Formulation

In this paper, our aim is to maximize the average long-term reward function by finding a UAV's optimal policy π^*, i.e., $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$. In particular, given the UAV's current energy and location, π^* determines an action that maximizes the long-term average reward function as follows:

$$\max_\pi \quad \mathcal{R}(\pi) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big(r_t(s_t, \pi(s_t))\big), \quad (7)$$

where $\mathcal{R}(\pi)$ is the long-term average reward obtained by the UAV according to the policy π, $\pi(s_t)$ is the selected action at state s at time slot t based on policy π, and $r_t(s_t, \pi(s_t))$ is the immediate reward by following policy π at time slot t. Thus, the optimal policy π^* will assist the UAV in dynamically making the optimal action according to its current observation, i.e., its position and energy level. Moreover, Proposition 1 shows the optimality of the proposed immediate reward function $r_t(s_t, a_t)$.

PROPOSITION 1. *"There always exists a maximum value of the immediate reward function."*

Proof: We first prove the optimality for each component of the immediate reward function, i.e., r_t^a and r_t^b. From (5), the speed selection reward function is a linear function of energy consumption m_t^a and an amount of collected data packet d_t^s.

Since the values of m_t^a and d_t^s are always positive and finite, there always exists a maximum value of the speed selection reward function $r_t^a(s_t, a_t)$. For the second component, i.e., $r_t^b(s_t, a_t)$, we can rewrite it as follows:

$$r_t^b(s_t, a_t) = c - (w_3 \exp(w_4 e_t) + w_5 \exp(w_6 l_t)), \quad \text{if } a_t = 0. \tag{8}$$

We now prove that $f(e_t, l_t) = (w_3 \exp(w_4 e_t) + w_5 \exp(w_6 l_t))$ is convex. Specifically, the Hessian matrix of $f(e_t, l_t)$ is calculated as follows:

$$H = \begin{pmatrix} w_3 w_4^2 e^{w_4 e_t} & 0 \\ 0 & w_5 w_6^2 e^{w_6 l_t} \end{pmatrix}. \tag{9}$$

Then, the determinant of H is computed by: $det_H = w_3 w_4^2 e^{w_4 e_t} w_5 w_6^2 e^{w_6 l_t}$. Since the weights in the immediate function are always positive, we have $det_H > 0$. Therefore, H is positive definite and $f(e_t, l_t)$ is convex, meaning that there always exists a minimum value of $f(e_t, l_t)$. As a result, $r_t^b(s_t, a_t)$ always has a global optimal value. Given above, there always exists a maximum value of the immediate reward function as its components are optimality and independent.

IV. OPTIMAL OPERATION POLICY FOR UAVs WITH DEEP REINFORCEMENT TRANSFER LEARNING

In the considered problem, the UAV does not have complete information about the surrounding environment in advance (e.g., the data arrival probabilities and the charging process) to obtain the optimal policy. In addition, the limited resource of UAV requires a low computational complexity solution. To deal with these challenges, we first propose a DRL algorithm that can help the UAV gradually learn the optimal policy without requiring complete information about surrounding environments in advance. We then develop a transfer learning framework aiming to reduce the learning time and improve the learning quality of the UAV, and thereby mitigating the computational complexity and resource consumption for the UAV. Consequently, our proposed solution is more applicable to aerial computing than conventional DRL approaches.

A. Deep Dueling Double Q-learning

In reinforcement learning, Q-learning is one of the most widely used algorithms because it can guarantee to converge to the optimal policy after the learning process [24]. However, the usage of Q-table to approximate the optimal state-action values $Q^*(s, a)$, namely Q-values, makes the Q-learning struggling in finding the optimal policy if the state and action spaces are very large [25], especially under the scenario considered in this work. In addition, the Q-learning and deep Q-learning may overestimate Q-values. This potentially results in a sub-optimal policy if overestimations are unevenly distributed over the state space [26]. Since all of the related works using reinforcement learning approaches (i.e., [8]–[10], [16]) propose Q-learning or deep Q-learning based algorithms, they are prone to low convergence and overestimation problems. To address these problems, we propose a Deep Dueling Double Q-learning algorithm (D3QL), in which three innovation techniques in DRL are adopted. First, the deep neural network is employed
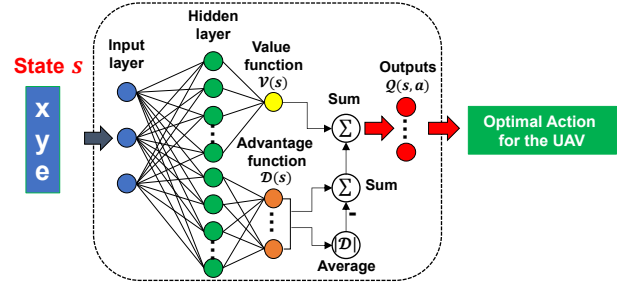

Fig. 3: The proposed deep dueling architecture for UAV-assisted IoT data collection.

to estimate the state-action function $Q^*(s, a)$ so that the curse-of-dimensionality problem of Q-learning can be effectively handled [18]. Second, the overestimation in Q-learning can be overcome by using the double deep Q-learning that separates the action selection and action evaluation processes instead of combining them in the Q-learning [19]. Finally, the learning process is stabilized by adopting the dueling neural network architecture where the state value function and advantage value function are estimated separately and simultaneously [20]. Thus, our proposed algorithm, i.e., D3QL, can take all the advantages of these techniques to improve the convergence rate, stabilize the learning process, and decrease the overestimation. As a result, the UAV can quickly achieve the optimal operation, and thereby maximizing both data collection and energy efficiencies for the system.

The D3QL algorithm is thoroughly described in Algorithm 1. Specifically, the learning process consists of some major steps. At the beginning of time step t, the UAV is at state s_t and performs action a_t according to ϵ-greedy policy. It then observes the next state s_{t+1} and receives reward r_t at the end of a time slot. This experience (i.e., a tuple $< s_t, a_t, r_t, s_{t+1} >$) can be used to train the neural network at step t. However, the high correlation in consecutive experiences generated by MDP can severely slow the learning process [27]. To that end, D3QL stores the experiences in a memory $\mathbf{B}$, then samples a mini-batch uniformly at random from $\mathbf{B}$. As a result, the variance of the update decreases, thereby improving the algorithm's convergence rate. In addition, the usage of the memory pool increases the data efficiency of the network's training since one experience can be used to update the neural network multiple times. The D3QL estimates $Q^*(s, a)$ by using a deep neural network (DNN), whose input layer has three inputs corresponding to the UAV's state dimensions, i.e., the x-coordinate, y-coordinate, and current energy level. In particular, feeding a state s to the DNN will return Q-values for all actions at this state, and each Q-value is given by one neuron of the DNN's output layer.

In this work, to improve the efficiency and stability of the learning process, we adopt a dueling neural network architecture [20] consisting of two parallel streams to simultaneously estimate the state-value function (i.e., $\mathcal{V}(s)$), and the advantage function (i.e., $\mathcal{D}(s, a)$), as illustrated in Fig. 3. Note that the state-value function $\mathcal{V}(s)$ specifies a the quality of a state s (i.e., how good to be at a state), whereas the action-value
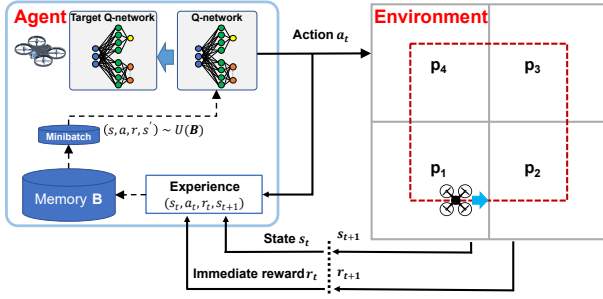
Fig. 4: The proposed D3QL-based model.

Algorithm 1 The D3QL Algorithm

1: Establish memory $\mathbf{B}$ and ϵ.
2: Establish Q-network Q with random parameters ϕ and target Q-network $\hat{Q}$ with parameters $\phi^- = \phi$.
3: **for** *step = 1 to T* **do**
4:    Choose action a_t according to the ϵ-greedy policy.
5:    Execute a_t, observe reward r_t and next state s_{t+1}.
6:    Save experience (s_t, a_t, r_t, s_{t+1}) in $\mathbf{B}$.
7:    Sample mini-batch of experiences randomly from $\mathbf{B}$, i.e., $(s, a, r, s') \sim U(\mathbf{B})$.
8:    Calculate Q-value $Q(s_k, a_k; \phi)$ and target Q-value G_t by (11) and (14), respectively.
9:    Take a gradient descent step with respect to the parameters of Q-network.
10:    Decrease the value of ϵ.
11:    Set $\hat{Q} = Q$ at every I steps.
12: **end for**

function $Q(s, a)$ determines the value of performing an action a at a state s. The advantage function is then defined by the relation between state-value and action-value functions under policy π as follows [20]:

$$\mathcal{D}^\pi(s, a) = Q^\pi(s, a) - \mathcal{V}^\pi(s). \tag{10}$$

In this way, the advantage function $\mathcal{D}^\pi(s, a)$ indicates the importance of action a in comparison with other actions at state s. Suppose that a state s is inputted to the dueling neural network, the Q-function is then calculated by

$$Q(s, a; \xi, \psi) = \mathcal{V}(s; \xi) + \mathcal{D}(s, a; \psi), \tag{11}$$

where ξ and ψ are weights and biases of the value and advantage streams, respectively. It is worth noting that directly using (11) may lead to mediocre performance of the algorithm since this equation is unidentifiable, i.e., $\mathcal{V}$ and $\mathcal{D}$ cannot be uniquely determined by a given Q. For example, if $\mathcal{V}$ increases the same amount that $\mathcal{D}$ decreases, Q is unchanged. In [20], the authors address this issue by subtracting the maximum value of the advantage stream as follows:

$$Q(s, a; \xi, \psi) = \mathcal{V}(s; \xi) + \left(\mathcal{D}(s, a; \psi) - \max_{a' \in \mathcal{A}} \mathcal{D}(s, a'; \psi) \right). \tag{12}$$

In this manner, $Q(s, a^*; \xi, \psi)$ equals to $\mathcal{V}(s; \xi)$ for the optimal action a^*, i.e., $a^* = \text{argmax}_{a \in \mathcal{A}} Q(s, a; \xi, \psi) = \text{argmax}_{a \in \mathcal{A}} \mathcal{D}(s, a; \psi)$. However, in (12), the advantage stream changes as fast as the change in predicted optimal action's advantage, i.e., $\max_{a' \in \mathcal{A}} \mathcal{D}(s, a'; \psi)$, leading to instability in estimating the Q-values. Hence, we can replace the max operator by the mean to stabilize the estimation of Q-function as follows [20]:

$$Q(s, a; \xi, \psi) = \mathcal{V}(s; \xi) + \left(\mathcal{D}(s, a; \psi) - \frac{1}{|\mathcal{A}|} \sum_{a'} \mathcal{D}(s, a'; \psi) \right). \tag{13}$$

To address the overestimation problem of Q-learning, we adopt the double deep Q-learning [19]. Specifically, the D3QL employs two dueling neural network-based estimators, which are Q-network for action selection, i.e., $Q(s, a; \xi, \psi)$, and target Q-network for action evaluation, i.e., $\hat{Q}(s, a; \xi^-, \psi^-)$. Thus, the target Q-value at step t is defined by:

$$G_t = r_t + \zeta \hat{Q}\left(s_{t+1}, \text{argmax}_a Q(s_{t+1}, a; \xi_t, \psi_t); \xi_t^-, \psi_t^-\right). \tag{14}$$

For convenience, let ϕ_t and ϕ_t^- denote the parameters of Q-network and those of target Q-network at time t, respectively. Since the goal of Q-network training is to minimize the gap between the estimated Q-value and target Q-value, we can

define the loss function at time t as follows:

$$\mathcal{L}_t(\phi_t) = \mathbb{E}_{(s, a, r, s')} \left[\left(G_t - Q(s, a; \phi_t) \right)^2 \right], \tag{15}$$

where (s, a, r, s') represents a data point in memory $\mathbf{B}$ used to train the Q-network.

Gradient Descent (GD) algorithm is widely used to minimize the loss function for deep learning algorithms because of its simplicity in implementation and ability to find the global minima [28]. In particular, GD calculates the cost function at time t as follows:

$$J_t(\phi_t) = \frac{1}{|\mathbf{B}|} \sum_{(s, a, r, s') \in \mathbf{B}} \mathcal{L}_t(\phi_t). \tag{16}$$

Then, the parameters of Q-network are updated by

$$\phi_{t+1} = \phi_t - \alpha_t \nabla_{\phi_t} J_t(\phi_t), \tag{17}$$

where α_t is a step size at time t determining how much the parameters are updated, and $\nabla_{\phi_t}(.)$ is the gradient of the cost function with respect to the parameters of Q-network ϕ. For each update, GD requires computing the loss and gradient for all data points in the memory $\mathbf{B}$, leading to a very slow process if the data size is large, especially for the considered problem. To that end, we adopt stochastic gradient descent (SGD) to speed up the parameter update process while guaranteeing the learning convergence [29]. Specifically, SGD only needs to calculate the gradient and cost of a mini-batch sampled uniformly at random from $\mathbf{B}$ for each time step. Hence, the computational complexity of this algorithm is significantly decreased. It is worth noting that the target Q-values G_t in the loss function (15) appear similar to the labels used for supervised learning. However, instead of being fixed before the training, G_t changes as fast as the changes in the Q-target network's parameters ϕ_t^-. Therefore, the target Q-network's parameters are only updated by cloning from the Q-network's parameters at every I steps to stabilize the training process, as shown in Fig. 4.
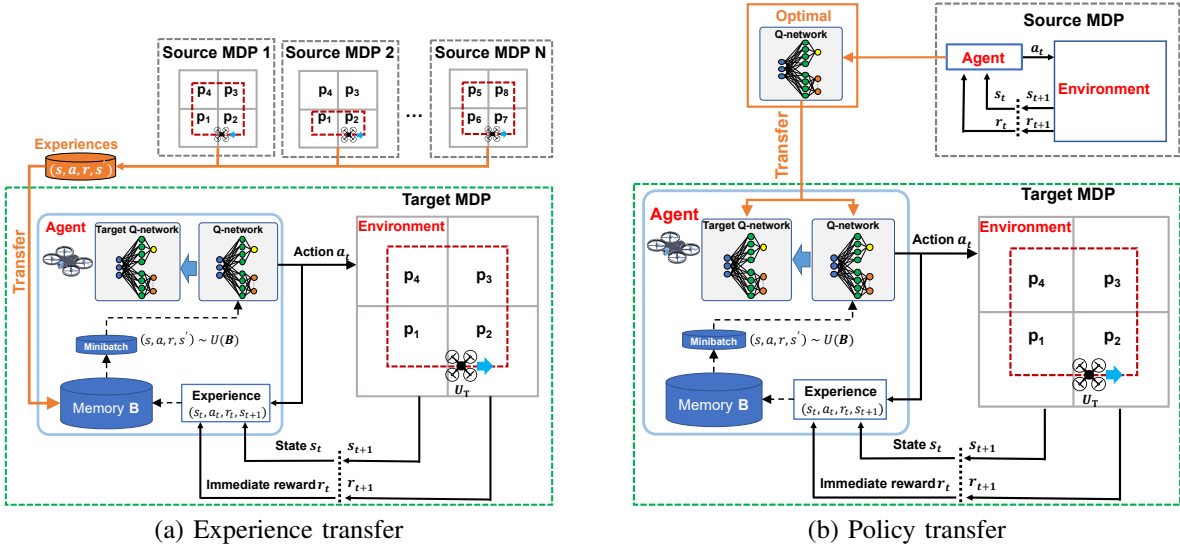
(a) Experience transfer   (b) Policy transfer

Fig. 5: The proposed D3QL-TL based models: (a) experience transfer and (b) policy transfer.

B. Optimal Operation Policy with Transfer Learning

Although D3QL can effectively address the shortcomings of Q-learning, it still poses some drawbacks inherited from conventional DRL when addressing scenarios with high sample complexity, as the considered problem in this work where the surrounding environment of the UAV is unknown in advance. First, it often takes a lot of time to train DNN, e.g., DQN's training time is up to 38 days for each Atari game [18]. If the environment dynamics or the trajectory of the UAV changes, the DNN may need to be retrained from scratch, yielding a high computational complexity. Consequently, it is unable to deploy on a UAV that has very limited energy and computing resources. Second, since the UAV should only return for replacing its battery when it is close to the station or its energy level is low, it needs sufficient experiences in this region, especially when flying over the station. However, as the UAV flies over its fixed trajectory, experiences obtained from this region are often very small compared with all obtained over the entire considered area. Therefore, the UAV may not have adequate information to learn an optimal policy. To address these challenges, we develop a novel framework leveraging transfer learning techniques, namely Deep Dueling Double Q-learning with Transfer Learning (D3QL-TL).

1) Transfer Learning in Reinforcement Learning: Transfer learning is a method of leveraging knowledge obtained when performing a source task in a source domain to enhance the learning process of target tasks in target domains [17], [30]–[32]. Typically, a domain contains labeled or unlabeled data given before the considered training process starts. However, data in RL is obtained via interactions between the agent (i.e., the UAV) and its surrounding environment. As a result, both the domain and task can be represented by an MDP. Thus, we can define transfer learning in RL as in Definition 1 [31].

Definition 1. *Transfer learning in RL: Suppose the source and target MDPs are defined. Transfer learning (TL) in RL intends to leverage the knowledge $\mathcal{K}_S$ obtained from the source MDP,*

i.e., the policy, the environment dynamics, and the data, as a supplement to the target MDP's information $\mathcal{K}_T$ to efficiently learn the target optimal policy π_T^* as follows:

$$\pi_T^* = \operatorname*{argmax}_{\pi_T} \mathbb{E}_{s \sim \mathcal{S}_T, a \sim \pi_T} \left[Q^{\pi_T}(s, a) \right], \qquad (18)$$

where π_T is a target MDP's policy approximated by an estimator, e.g., a table or a DNN, that is trained on both $\mathcal{K}_S$ and $\mathcal{K}_T$.

Note that transfer learning for DRL may look like supervised learning since they both use existing data, but they are very different. In particular, all DRL data used to train a DNN are unlabeled and on-the-fly data generated by interactions between an agent (e.g., the UAV) and its surrounding environment. Although the source data are collected in advance for the target UAV, they are just observations of the source UAV about the source environment, which do not have any label to indicate which action the UAV should take. Thus, the target UAV still needs learning algorithms (e.g., D3QL) to learn the optimal policy gradually. Our proposed TL can help the UAV utilize the knowledge from the source domain to avoid bad decisions at the beginning of the learning process when the UAV is exploring the environment by taking a random action, thereby improving the learning rate and learning quality.

To measure the effectiveness of TL, we can use three metrics, including jump-start, asymptotic performance, and time-to-threshold [17]. In particular, jump-start measures how much the UAV's performance at the beginning of the learning process can be improved by applying TL, while the asymptotic performance measures this improvement at the end of the learning process. The third metric, i.e., time-to-threshold, measures how fast TL can help the UAV to achieve a predefined performance level compared with the scenario without TL. It is worth highlighting that TL cannot guarantee improvement in the learning curve. It may even negatively impact the learning in the target MDP if the transfer knowledge is not carefully chosen. Thus, in the following sections, we propose a transfer

Algorithm 2 The D3QL-TL

1: Establish memory $\mathbf{B}$ and ϵ.
2: Establish Q-network Q with random parameters ϕ, and target Q-network $\hat{Q}$ with parameters $\phi^- = \phi$.
3: **if** Experience transfer **then**
4:     Copy the experience set of source MDP $\mathcal{M}_S$ to $\mathbf{B}$.
5: **else if** Policy transfer **then**
6:     Re-initialize Q and $\hat{Q}$ with the parameter of source Q-network's parameters ϕ_S.
7: **else if** Hybrid transfer **then**
8:     Re-initialize Q and $\hat{Q}$ with the parameter of source Q-network's parameters ϕ_S.
9:     Copy the experience set of source MDP $\mathcal{M}_S$ to $\mathbf{B}$.
10: **end if**
11: **for** *step = 1 to T* **do**
12:     Choose action a_t according to the ϵ-greedy policy.
13:     Execute a_t, observe reward r_t and next state s_{t+1}.
14:     Save experience (s_t, a_t, r_t, s_{t+1}) in $\mathbf{B}$.
15:     Sample mini-batch of experiences randomly from $\mathbf{B}$, i.e., $(s, a, r, s') \sim U(\mathbf{B})$.
16:     Calculate Q-value $Q(s_k, a_k; \phi)$ and target Q-value G_t by (11) and (14), respectively.
17:     Take a gradient descent step with respect to the parameters of Q-network.
18:     Decrease the value of ϵ.
19:     Set $\hat{Q} = Q$ at every I steps.
20: **end for**

learning framework that can reduce the learning time and learning quality for D3QL.

2) Deep Dueling Double Q-learning with Transfer Learning: The details of D3QL-TL are presented in Algorithm 2. In particular, as illustrated in Fig. 5, we consider a UAV U_S working in an IoT data collection environment formulated by the MDP framework $\mathcal{M}_S$. Then, the knowledge of U_S can be leveraged to help a new UAV U_T effectively learn the optimal policy for working in another environment formulated by the MDP framework $\mathcal{M}_T$. In practice, $\mathcal{M}_T$ can be the same or different with $\mathcal{M}_S$. Moreover, the transferring knowledge can be in the form of the policy and/or experiences of the source UAV, i.e., U_S.

To that end, D3QL-TL defines three types of knowledge transferring as follows:

- *Experience Transfer (ET)*: This approach aims to leverage a set of experiences E_S, in which each element is a experience tuple $< s, a, r, s' >$, obtained in the source MDP, i.e., $\mathcal{M}_S$, to improve the learning process of the target UAV, i.e., U_T, working in the target MDP, i.e., $\mathcal{M}_T$. Specifically, E_S is first copied to the memory buffer of the target UAV. Then, these transferred experiences and target UAV's new experiences are used to train the Q-network. In this manner, the target UAV can quickly get adequate information, and thereby significantly improving the learning speed. In addition, the quality of the experiences also affects the learning process. For example, an experience does not have much value if it is easy to be obtained by the target UAV. In contrast,

an experience is considered to be valuable if it is hard to obtain and highly impacts the system performance. For example, experiences obtained when the UAV is near the station may have high values because they not only contain information about environment dynamics (i.e., packet arrival probabilities) but also may reveal value information about the right time to take the battery replacement action.

- *Policy Transfer (PT)*: This approach directly transfers the policy of a source UAV to a target UAV. In particular, U_T starts the learning process with the policy of U_S, which is represented by the Q-network of U_S, called the source Q-network. Hence, the U_T's Q-network is initialized by the source Q-network parameters ϕ_S. Then, the U_T's Q-network is trained with the new experiences of U_T obtained in the target MDP $\mathcal{M}_T$. Thus, starting with the source policy can help U_T to avoid random decisions caused by the randomness of action selection at the beginning of the learning process, e.g., inappropriately choosing the battery replacement action.

- *Hybrid*: This approach aims to leverage the benefits of both experience and policy transfer types. Particularly, the hybrid scheme can improve not only the jump-start but also the asymptotic performance.

Note that the efficiency of each transferring technique depends on the relationship between the source and the target MDPs. For example, if the source and target MDPs are very similar, policy transfer may yield a better result in terms of convergence rate than that of the experience transfer technique. In contrast, when the source and target MDP are not similar, e.g., differences in environment dynamics, the experience transfer should be a better choice.

C. Optimality and Computational Complexity Analysis

In RL, if a linear function approximates the value function, the learning process can be guaranteed to converge to the optimal policy [36]. Whereas, if a nonlinear function (e.g., a neural network) is used instead, it may not converge to the optimal one [36]. In our proposed learning algorithms, we adopt two innovative techniques (i.e., the dueling architecture and double Q-learning) and three transfer learning approaches to stabilize the learning process and improve the learning quality, thereby improving the convergence rate. Thus, even though the optimality of our deep reinforcement learning algorithm, i.e., D3QL-TL, could not be proven theoretically, the intensive simulation results show that D3QL-TL obtains a stable and superior performance compared to those of other approaches.

We now discuss the complexities of our proposed algorithms (i.e., the D3QL and D3QL-TL), which mainly depend on the training process of the Q-network. Note that the Q-network training processes of D3QL and D3QL-TL are the same, so their complexities are identical. In the Q-network, there is one input layer L_i, one hidden layer L_h, and two output layers L_v and L_a corresponding to the state value and advantage streams, respectively. It is worth noting that training a DNN is typically carried out with matrix multiplication. Thus, the
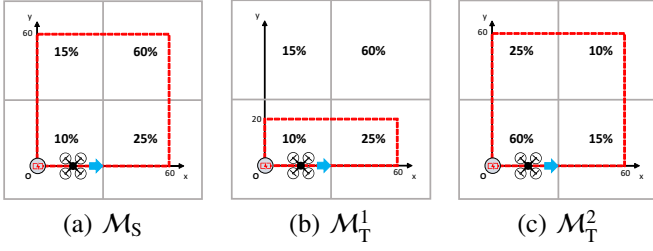
(a) $\mathcal{M}_S$ (b) $\mathcal{M}_T^1$ (c) $\mathcal{M}_T^2$

Fig. 6: (a) Source MDP, (b) the first target MDP, and (c) the second target MDP.

complexity of feeding a mini batch with size S_b to the Q-network is $O\Big(S_b\big(|L_i||L_h| + |L_h||L_v| + |L_h||L_v|\big)\Big)$, where $|.|$ is the layer's size, i.e., a number of neurons in a layer. Given the training process takes T iterations, the complexity of the proposed algorithms are $O\Big(S_b\big(|L_i||L_h| + |L_h||L_v| + |L_h||L_v|\big)\Big)$.

In general, more computing resource is required to train a DNN, especially the one with a complex architecture. However, the deep Q-network in our proposed algorithms is only contains four layers in which only one is fully connected (i.e., the hidden layer). In addition, the proposed transfer methods can significantly reduce the complexity of reinforcement learning, as can be seen through intensive simulation results in the next section.

V. PERFORMANCE EVALUATION

A. Parameter Setting

We first evaluate our proposed approaches in an IoT system, in which a UAV flies over a predefined trajectory in a considered area to collect IoT data, as in [13]–[15]. This area is divided into four zones such that the UAV's travel distance in each zone equals $60m$, as illustrated in Fig. 6(a). The station is located at the origin, i.e., $(0,0)$. The probabilities of packet arrival in these zones are given by a vector $\mathbf{p} = [p_1, p_2, p_3, p_4]$, e.g., p_1 is the packet arrival probability when the UAV is flying over zone 1 and so on. Since the UAV collects data while flying, it often flies at a low speed (e.g., 5 m/s) to maintain the reliability of the data collection process [14]. Therefore, we consider that the UAV has three speeds: $1, 3,$ and 5 (m/s). The UAV consumes 2, 3, and 4 (energy units/time slot) when flying at $1m/s$, $3m/s$, and $5m/s$, respectively. Note that our proposed MDP framework and learning algorithms can help the UAV learn the optimal policy according to its observation (i.e., its current position and energy level) regardless of the UAV's specifications, i.e., its available speeds and corresponding energy consumption. Here, the energy unit is used to quantify an amount of energy (as in [34], [35]), and energy consumption values are only to demonstrate how much energy the UAV uses at each time slot for its operation. In practice, UAVs manufactured by different brands may have different specifications, but the proposed algorithm still can obtain the optimal policy for the UAV. The parameters of the reward function are provided in Table. I.

The settings for our proposed algorithms are set as follows. For the ϵ-greedy policy, ϵ is first set at 1, then gradually de-
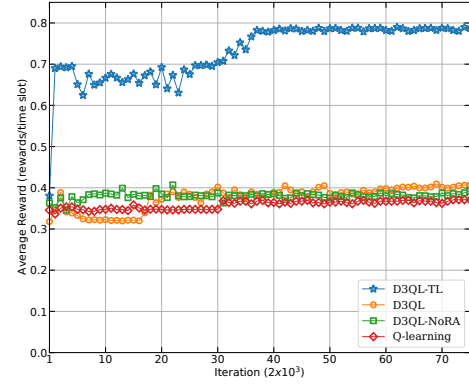
(a) Convergence rate of proposed algorithms
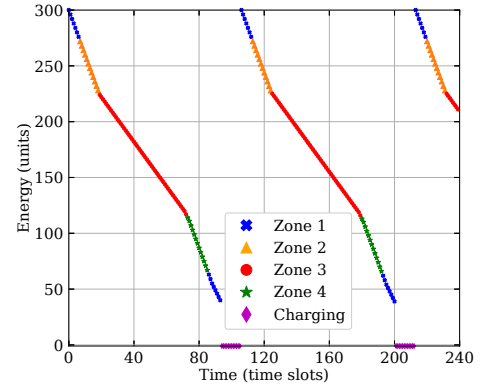
(b) Policy obtained by D3QL-TL

Fig. 7: Convergence rate and policy.

creased to 0.01. For all the proposed algorithms, the discount factor is set at 0.9. In Q-learning, the learning rate β is 0.1. The architectures of Q-network and target Q-network are illustrated in Fig. 3. We use typical hyperparameters for training DNN, e.g., the learning rate and the frequency update of $\hat{Q}$ are set to 10^{-4} and 10^4, respectively, as those in [18], [33].

For the experience transfer approach in D3QL-TL, experiences are selected to be transferred based on their valuable information. Recall that if the UAV performs the battery replacement action when it is far from the station, it always receives a very small reward compared with those of other actions, as in (4). Therefore, the UAV's experiences in this area can imply that it should not take the battery replacement action. In contrast, experiences obtained when the UAV is near the station can contain both information, i.e., when it not worth to return for charging (e.g., current energy level is high) and when it is worth to take the battery replacement action (e.g., current energy level is low). Thus, we choose experiences obtained when the UAV flies near the station to be transferred.

We study a scheme where the UAV does not have complete information about the surrounding environment in advance, e.g., the data arrival probabilities and the energy replenishment process. Hence, we compare our approach with three other deterministic policies, i.e., the UAV always flies at (1) lowest speed ($1m/s$), (2) middle speed ($3m/s$), and (3) highest speed ($5m/s$). In addition, the Q-learning and D3QL are selected

Parameters	Ω	w_1	w_2	c_1	c_2	c_3	c_4	c_5	E	$\mathbf{p}$	t_b	v_r
Value	1	1	0.3226	5	0.5	0.5	0.022	0.2	300	$[0.1, 0.25, 0.6, 0.15]$	10	1

TABLE I: Simulation parameters.

(a) Average reward   (b) Average throughput   (c) Average energy consumption

Fig. 8: Vary battery replacement time.

(a) Average reward   (b) Average throughput   (c) Average energy consumption
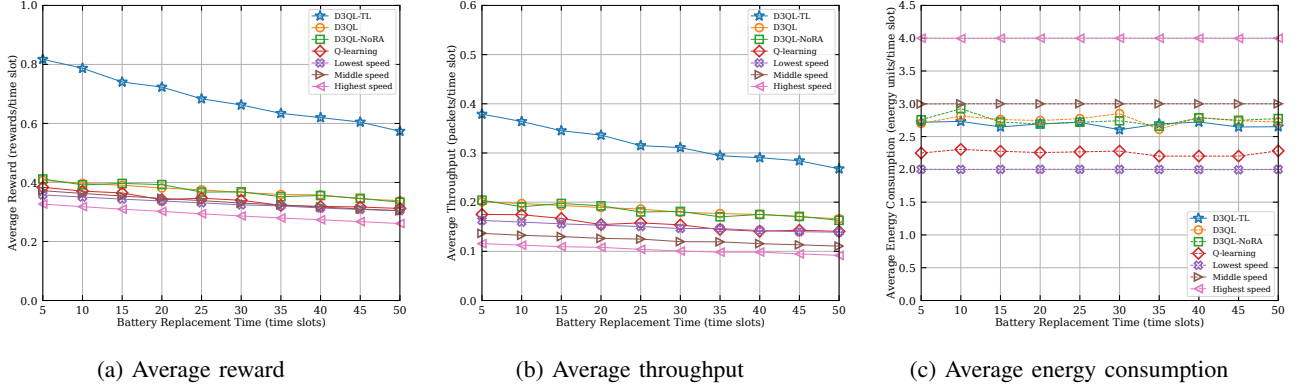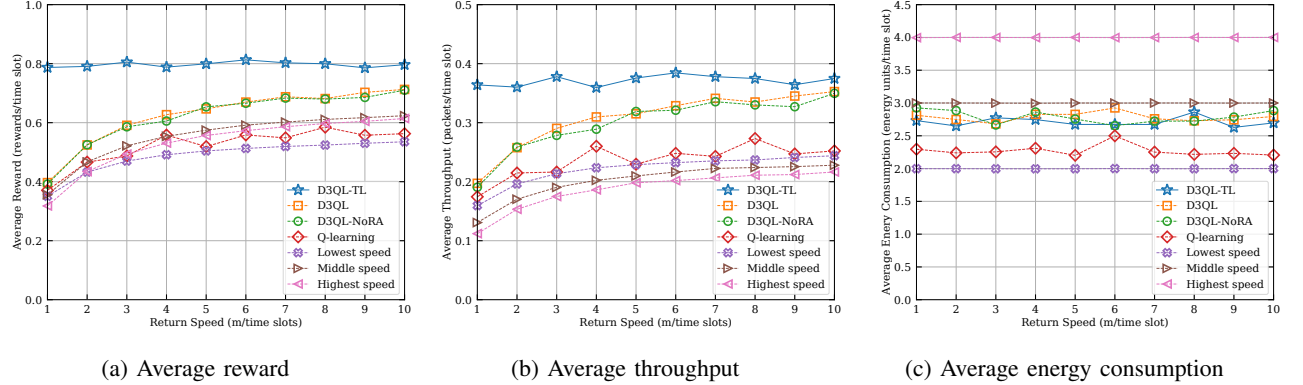
Fig. 9: Vary the UAV's return speed.

as the baseline methods to demonstrate the effectiveness of our proposed TL techniques since they are widely used algorithms and the most recent advanced techniques in RL, respectively. Moreover, to investigate impacts of the battery replacement action on the system performance, we consider an approach, namely D3QL-NoRA, where the D3QL will still be implemented on the UAV, but the UAV will not select the battery replacement action.

B. Simulation Results

In the simulation, we first evaluate the performance of our proposed learning algorithm, i.e., D3QL-TL, by examining the convergence rate and the obtained policy. Then, we evaluate the system performance when varying some important parameters (e.g., battery replacement time, UAV's energy capacity, return speed, and packet arrival probabilities) to assess their influences on the system performance. For the D3QL-TL, the experience transfer type is chosen because it can leverage the experiences obtained during the learning phases of other algorithms, i.e., D3QL and Q-learning. Finally, to gain more

insights into the effectiveness of three transferring types in D3QL-TL, we compare their performance in different scenarios, i.e., changing the UAV trajectory and the probabilities of receiving a packet.

1) Convergence and Policy: In Fig. 7 (a), we compare the convergence of proposed algorithms in terms of average rewards. At the beginning of the learning processes, the average rewards of proposed approaches are close to each other, approximately 0.35. However, only after 4,000 iterations, D3QL-TL's average reward is nearly 170% greater than those of other approaches. Then, D3QL-TL almost converges to the optimal policy after 7.5×10^4 iterations, and its average reward becomes stable at around 0.78, which is more than 190% greater than those of other learning algorithms. Interestingly, D3QL and D3QL-NoRA converge to policies that achieve similar average rewards. This suggests that D3QL is unable to take advantage of the battery replacement action. In other words, D3QL cannot effectively handle this complicated decision-making situation. This result implies the outperformance of our proposed algorithm, i.e., D3QL-TL, compared with other methods when addressing the extremely complex problem as
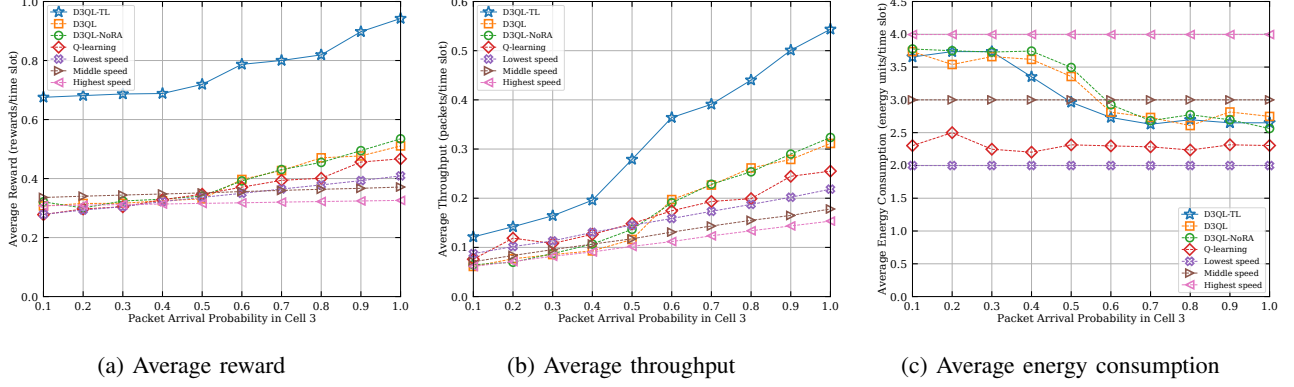
(a) Average reward   (b) Average throughput   (c) Average energy consumption

Fig. 10: Vary the packet arrival probability of zone 3.

(a) Average reward   (b) Average throughput   (c) Average energy consumption
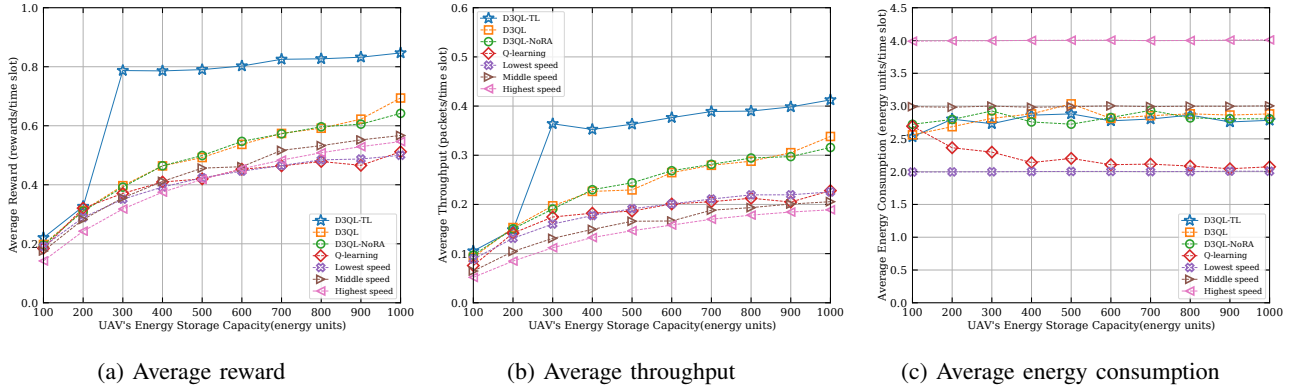
Fig. 11: Vary the UAV's energy capacity.

the one considered in this paper.

Next, we show the policy obtained by D3QL-TL after 1.5×10^5 iterations in Fig. 7 (b). In particular, a point indicates the energy level of the UAV at the beginning of a time slot. The slope of a straight line between two points indicates the selected action, e.g., the steeper this line is, the higher speed is selected. Generally, the lowest speed is selected in a zone that has a high probability of successfully collecting a packet. In contrast, the highest speed is selected in a zone that has a low probability of receiving a packet, as shown in Fig. 7 (b). Given $\mathbf{p} = [0.1, 0.25, 0.6, 0.15]$, as in Table. I, the D3QL-TL selects the lowest speed in zone 3, and the highest speed in zone 1, 2, and 4. More interestingly, the UAV experiences the middle and high speeds in zone 1. In particular, it travels at the highest speed until its energy decreases to 55 energy units at time slot 88, then the middle speed is selected. When its energy drops to 40 energy units at time slot 93, equivalent to 13.3% energy level, the UAV takes the battery replacement action. Note that Fig. 7 (b) also reveals information of the location where the UAV should take the battery replacement action by energy replenishment time, i.e., t_e. Particularly, given $t_e = 12$ time slots, the location that the UAV decides to return only $1m$ away from the station. This result demonstrates the impacts of location and energy level on the optimal policy of the UAV.

2) Performance Evaluation: In this section, we perform simulations to evaluate our proposed algorithms in terms of average reward, throughput, and system energy consumption. The parameters are set to be the same as those in V-B1. The policies of proposed learning algorithms, including Q-learning, D3QL, and D3QL-TL, are obtained after 1.5×10^5 iterations.

In Fig. 8, we vary the battery replacement time, i.e., t_b. Clearly, the average reward and throughput of all policies decrease as the battery replacement time increases from 5 to 50 time slots. This is stemmed from the fact that given a fixed duration, the less time the UAV needs to replace the battery, the more time it can spend collecting data. As a result, the data collection efficiency of the system reduces. It can be observed that D3QL-TL can significantly outperform other approaches in terms of average reward and throughput, while it still obtains a reasonable energy consumption per time slot. In particular, the average reward and throughput achieved by D3QL-TL are up to 200% and 185% greater than those of the second-best policy, i.e., D3QL, respectively.

Next, we vary the return speed v_r of the UAV to observe the system's performance in terms of average reward, throughput, and energy consumption. Figs. 9 (a) and (b) show that as the return speed v_r increases from $1(m/s)$ to $10(m/s)$, all policies have upward trends in terms of the average reward and throughput, except that of the D3QL-TL. The reason is that

(a) The first scenario
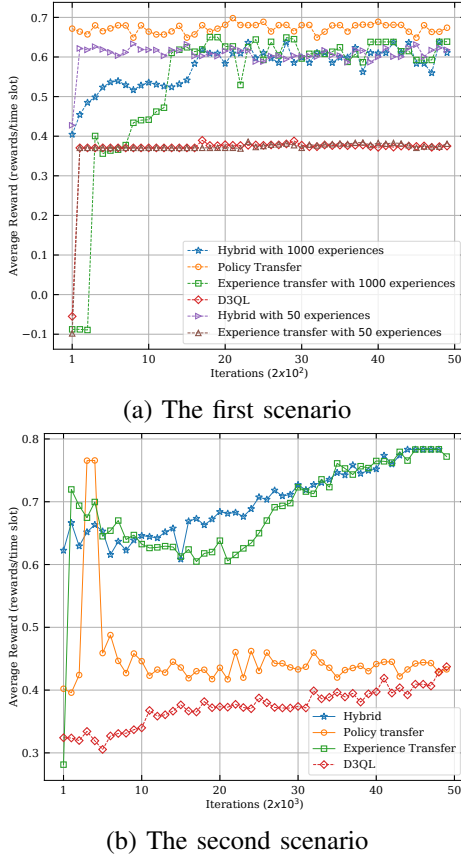

(b) The second scenario

Fig. 12: Convergence of the proposed TL schemes.

the increase of v_r leads to the decrease of time for returning to the station, i.e., t_f, and thus the UAV has more time to collect data in a fixed duration. More interestingly, when the return speed is low (e.g., lower than $3(m/s)$), the lowest speed policy obtains a higher average reward than that of the highest speed policy, as observed in Fig. 9 (a). Nevertheless, the lowest speed policy obtains the lowest performance (i.e., the average reward is lowest) when the return speed is large. This emanates from the fact that given a fixed serving time, the energy consumption of the highest speed is higher than that of the lowest speed. Therefore, the UAV has to replace its battery more frequently if it flies at the highest speed rather than if it flies at the lowest speed. Consequently, as the return speed increases, the highest speed policy eventually performs better than that of the lowest speed policy. Unlike other policies, D3QL-TL achieves a stable average reward, approximately 0.8, that is always much higher than those of other policies, as shown in Fig. 9 (a). This is because the UAV equipped with D3QL-TL can learn an excellent policy, e.g., taking battery replacement action when the UAV is close to the station, making it more adaptable to the changes of the return speed.

We then vary the packet arrival probability p_3 of zone 3 while those of other zones are unchanged, as provided in Table I, and observe the performance of our proposed approaches. Figs. 10 (a) and (b) clearly show the increase of average rewards and throughputs for all policies when p_3 increases from 0.1 to 1.0. Interestingly, as shown in

Fig. 10 (a), when p_3 is small, e.g., lower than 0.4, the lowest speed policy obtains lower rewards than those of the highest speed. However, when p_3 becomes larger, the lowest speed achieves a higher average reward than that of the highest speed. This implies that the UAV should fly at the lowest speed if the packet arrival probability is high, and vice versa. Fig. 10 (c) demonstrates that our proposed algorithm, i.e., D3QL-TL, can learn the environment's dynamic, e.g., the packet arrival probability. In particular, when the probability of receiving a packet is low, e.g., less than 0.4, the UAV's average energy consumption is high, approximately 3.65 energy units/time slot, indicating that the highest speed is selected more frequently than the lowest speed. In contrast, when this probability becomes higher, e.g., larger than 0.5, the UAV's energy consumption decreases to around 2.6, implying that the lowest speed is the most frequently selected speed. To that end, D3QL-TL can leverage this knowledge to consistently obtain the best performance compared with other policies.

Finally, in Fig. 11, we vary the UAV's energy storage capacity E to study its impact on the system performance. In particular, when E is varied from 100 to 1000 energy units, the average rewards and throughputs of all policies increase, as shown in Figs. 11 (a) and (b), respectively. It is worth highlighting that if E is small, e.g., less than 500, the performance of the lowest speed is better than that of the highest speed, as illustrated in Fig. 11 (a). However, the highest speed outperforms the lowest speed when E is larger than 500. This due to the fact that when E is small, the UAV's battery has to be replaced more frequently, leading to a downgrade of the UAV's data collection efficiency. Thus, the UAV must conserve more energy by flying at the lowest speed. By balancing between energy usage efficiency and data collection efficiency, our proposed D3QL-TL approach can always achieve the highest performance compared to other policies.

3) Transfer Learning Strategies: In this section, we evaluate and compare the effectiveness of three TL types in D3QL-TL (i.e., experience transfer (ET), policy transfer (PT), and hybrid transfer) in different scenarios, as shown in Figs. 6 (b) and (c). In particular, the source MDP, i.e., $\mathcal{M}_S$ is defined as the MDP described in Section V-A, and the simulation parameters are also provided in Table. I. Then, the optimal policy obtained by D3QL-TL after 1.5×10^5 iterations and the UAV's experiences gathered in the source MDP are leveraged to reduce the learning time and learning quality. To gain an insight of when and how much these transfer learning techniques can improve the learning process of D3QL, we define two scenarios as follows:

- In the first scenario (illustrated in Fig. 6 (b)), the target MDP (i.e., $\mathcal{M}_T^1$) is the same as $\mathcal{M}_S$ except the trajectory. The UAV only flies over two zones, i.e., zone 1 and 2. In each zone, it travels $80m$.
- In the second scenario (illustrated in Fig. 6 (c)), the difference between target MDP (i.e., $\mathcal{M}_T^2$) and $\mathcal{M}_S$ is the probabilities of receiving a packet, i.e., $\mathbf{p} = [0.6, 0.15, 0.1, 0.25]$.

We first compare the convergence rate of several transfer learning schemes of D3QL-TL and D3QL in the first scenario in Fig. 12 (a). To investigate how experiences impact on

the learning process of D3QL-TL, we select two sizes of experience sets, which are 50 and 100. As shown in Fig. 12 (a), after 5×10^4 learning iterations, the average rewards obtained by all types of D3QL-TL can achieve up to 179% greater than that of D3QL, except the ET with the size of 50. For the ET group, when the experience size is small, e.g., 50, transfer learning does not improve the learning process of D3QL since the average reward of ET is similar to that of D3QL. However, when this size is large enough, e.g., 10^3, the asymptotic performance of D3QL-TL is 171% greater than that of D3QL in terms of average reward. Interestingly, for the hybrid approach, when the experience size decreases, the D3QL-TL's performance increases. Especially, the PT, which is equivalent to the hybrid with zero experience size, consistently outperforms other transfer approaches in all the metrics, i.e., jump-start, time-to-threshold, and asymptotic performance. Specifically, only after 10^3 iterations, the PT obtains the optimal policy, and its average reward is stable at around 0.68. Thus, the PT can help the UAV to reduce the learning time up to 50% compared with those of other approaches. These results demonstrate that if the environment's dynamics in the target MDP, e.g., probabilities of receiving a packet, are similar to those in the source MDP, PT is the best choice. The reason is that the change of trajectory makes source experiences less efficient than that of the source policy. It is worth noting that only schemes with PT can improve the system performance at the beginning of the learning process because this policy can help the UAV choose valuable actions in this period, e.g., selecting battery replacement action when it is near the station and its energy level is low.

In Fig. 12 (b), we show the results of the second scenario where the packet arrival probabilities are different from that of the source MDP. In this scenario, we set the experience size to 10^3 for the ET and hybrid schemes. Again, it can be observed that only schemes with PT can improve the initial performance, e.g., during the first 2×10^3 iterations. Unlike the first scenario, the PT yields the worst performance among the transfer learning schemes, and its asymptotic performance is almost zero, meaning that there is no improvement in terms of average reward at the end of the learning process. In contrast, the ET and hybrid schemes achieve similar asymptotic performance, approximately 172%. However, hybrid's jump-start metric, i.e., 192%, is significantly higher than that of ET, i.e., 86%. These results suggest that when the environment dynamics change, the hybrid scheme should be selected.

Lessons learned: The above results demonstrate that applying TL to DRL is not straightforward. If it is not carefully implemented, TL may not be able to improve the performance of DRL, e.g., the ET with 50 experiences in the first scenario and the PT in the second scenario. In the first case, the change in UAV's trajectory means that the UAV still works in the same environment, but it follows another path. Therefore, the PT can quickly achieve the best policy because it can directly improve the performance of the UAV instead of gradually transferring the source knowledge in the ET. On the other hand, the probability of successfully collecting a packet changes in the second scenario, meaning that the UAV is placed in a different environment. In this context, the hybrid transfer technique can obtain the highest performance since it can leverage both the policy and the important experiences in the source MDP.

VI. CONCLUSIONS

In this paper, we develop a novel Deep Dueling Double Q-learning with Transfer Learning algorithm (D3QL-TL) that jointly optimizes the flying speed and energy replenishment activities for the UAV to maximize the data collection performance of a UAV-assisted IoT system. The proposed algorithm effectively addresses not only the dynamic and uncertainty of the system but also the high dimensional state and action spaces of the underlying MDP problem with hundreds of thousands of states. In addition, the proposed TL techniques (i.e., experience transfer, policy transfer, and hybrid transfer) allow UAVs to "share" and "transfer" their learned knowledge, resulting in a decrease of learning time and an improvement of learning quality. The simulation results show that our proposed solution can significantly improve the system performance (i.e., data collection and energy usage efficiency) and has a remarkably lower computational complexity compared with other conventional approaches.

REFERENCES

[1] N. H. Chu, D. T. Hoang, D. N. Nguyen, N. V. Huynh and E. Dutkiewicz, "Fast or slow: An autonomous speed control approach for UAV-assisted IoT data collection networks," in *Proceedings of 2021 IEEE Wireless Communications and Networking Conference (WCNC),* 2021, pp. 1-6.

[2] Cisco Annual Internet Report (2018-2023) White Paper, [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] J. Wang, C. Jiang, Z. Wei, C. Pan, H. Zhang, and Y. Ren, "Joint UAV hovering altitude and power control for space-air-ground IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1741-1753, Apr. 2019.

[4] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1274-1285, Jun. 2020.

[5] Facebook, "Connecting the world from the sky," *Facebook Technical Report,* 2014.

[6] https://loon.com/

[7] X. Zeng, F. Ma, T. Chen, X. Chen and X. Wang, "Age-optimal UAV trajectory planning for information gathering with energy constraints," in *Proceedings of 2020 IEEE/CIC International Conference on Communications in China (ICCC),* 2020, pp. 881-886.

[8] O. Bouhamed, H. Ghazzai, H. Besbes and Y. Massoud, "A UAV-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling," *IEEE Access,* vol. 8, pp. 110446-110460, Jun. 2020.

[9] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet of Things Journal,* vol. 8, no. 5, pp. 3786-3800, Mar. 2021.

[10] S. Fu et al., "Energy-efficient UAV-enabled data collection via wireless charging: A reinforcement learning approach," *IEEE Internet of Things Journal,* vol. 8, no. 12, pp. 10209-10219, Jun. 2021.

[11] X. Xu, H. Zhao, H. Yao and S. Wang, "A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT," *IEEE Internet of Things Journal,* vol. 8, no. 4, pp. 2431-2443, Feb. 2021.

[12] F. Shan, J. Luo, R. Xiong, W. Wu, and J. Li, "Looking before crossing: An optimal algorithm to minimize UAV energy by speed scheduling with a practical flight energy model," in *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM),* IEEE, 2020, pp. 1758-1767.

[13] J. Gong, T. Chang, C. Shen and X. Chen, "Flight time minimization of UAV for data collection over wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1942-1954, Sep. 2018.

[14] Q. Pan, X. Wen, Z. Lu, L. Li, and W. Jing, "Dynamic speed control of unmanned aerial vehicles for data collection under internet of things," *Sensors*, vol. 18, no. 11, Nov. 2018.

[15] X. Lin, G. Su, B. Chen, H. Wang and M. Dai, "Striking a balance between system throughput and energy efficiency for UAV-IoT systems," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10519-10533, Dec. 2019.

[16] K. Li, W. Ni, E. Tovar and A. Jamalipour, "On-board deep Q-network for UAV-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology,* vol. 68, no. 12, pp. 12215-12226, Dec. 2019.

[17] M. E. Taylor and P. Stone. "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, no. 7, pp. 1633-1685, 2009.

[18] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[19] H. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning." in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, AAAI Press, 2016, pp. 2094-2100.

[20] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1995-2003.

[21] H. Khodr, N. Kouzayha, M. Abdallah, J. Costantine and Z. Dawy, "Energy efficient IoT sensor with RF wake-up and addressing capability," *IEEE Sensors Letters,* vol. 1, no. 6, pp. 1-4, Dec. 2017.

[22] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6614-6627, Dec. 2018.

[23] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks." in *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM),* IEEE, 2019, pp. 1-6.

[24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279-292, 1992.

[25] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133-3174, May 2019.

[26] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School*, Hillsdale, NJ, 1993.

[27] S. Halkjear and O. Winther, "The effect of correlated input data on the dynamics of learning." in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, 1996, pp. 169-175.

[28] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research,* 2019, pp. 1675-1685.

[29] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400-407, 1951.

[30] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345-1359, Oct. 2009.

[31] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey." arXiv:2009.07888 [cs.LG], Sep. 2020.

[32] C. T. Nguyen *et al.*, "Transfer learning for future wireless networks: A comprehensive survey." arXiv:2102.07572v1 [cs.LG], Feb. 2021.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[34] C. H. Liu, Z. Chen, J. Tang, J. Xu and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 9, pp. 2059-2070, Sept. 2018.

[35] N. Van Huynh, D. T. Hoang, D. N. Nguyen and E. Dutkiewicz, "DeepFake: Deep dueling-based deception strategy to defeat reactive jammers," *IEEE Transactions on Wireless Communications,* vol. 20, no. 10, pp. 6898-6914, Oct. 2021.

[36] V. A. Papavassiliou and S. Russell. "Convergence of reinforcement learning with general function approximators," in *Proceedings of IJCAI,* 1999, pp. 748-755.