Deep-Reinforcement-Learning-Based Drone Base Station Deployment for Wireless Communication Services

Getaneh Berie Tarekegn, *Student Member, IEEE*, Rong-Terng Juang, *Member, IEEE*, Hsin-Piao Lin, *Member, IEEE*, Yirga Yayeh Munaye, Li-Chun Wang, *Fellow, IEEE*, and Mekuanint Agegnehu Bitew

***Abstract*—Over the last few years, drone base station (DBS) technology has been recognized as a promising solution to the problem of network design for wireless communication systems, due to its highly flexible deployment and dynamic mobility features. This article focuses on the 3-D mobility control of the DBS to boost transmission coverage and network connectivity. We propose a dynamic and scalable control strategy for drone mobility using deep reinforcement learning (DRL). The design goal is to maximize communication coverage and network connectivity for multiple real-time users over a time horizon. The proposed method functions according to the received signals of mobile users, without the information of user locations. It is divided into two hierarchical stages. First, a time-series convolutional neural network (CNN)-based link quality estimation model is used to determine the link quality at each timeslot. Second, a deep *Q*-learning algorithm is applied to control the movement of the DBS in hotspot areas to meet user requirements. Simulation results show that the proposed method achieves significant network performance in terms of both communication coverage and network throughput in a dynamic environment, compared with the *Q*-learning algorithm.**

***Index Terms*—Channel estimation, communication coverage, convolutional neural network (CNN), deep reinforcement learning (DRL), drone base station (DBS) mobility control, network connectivity.**

Manuscript received 12 March 2022; revised 9 May 2022; accepted 6 June 2022. Date of publication 13 June 2022; date of current version 24 October 2022. This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Grant 110-2221-E-027-033-MY2 and Grant 109-2222-E-035-003-MY2. *(Corresponding author: Getaneh Berie Tarekegn.)*

Getaneh Berie Tarekegn is with the Department of Electrical Engineering and Computer Science, National Taipei University of Technology, Taipei 10608, Taiwan (e-mail: gechb21@gmail.com).

Rong-Terng Juang is with the Department of Electronic Engineering, Feng Chia University, Taichung 40724, Taiwan (e-mail: rtjuang@mail.fcu.edu.tw).

Hsin-Piao Lin is with the Department of Electronic Engineering, National Taipei University of Technology, Taipei 10608, Taiwan (e-mail: hplin@ntut.edu.tw).

Yirga Yayeh Munaye and Mekuanint Agegnehu Bitew are with the Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar University, Bahir Dar 26, Ethiopia (e-mail: byyirga@gmail.com; memekuanint@gmail.com).

Li-Chun Wang is with the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 300-10, Taiwan (e-mail: lichun@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/JIOT.2022.3182633

I. Introduction

WIRELESS communication systems play an important role in daily life, such as the Internet of Things (IoT), context-aware advertising, and smart cities. The aim of these systems is to support various communication applications with heterogeneous requirements of Quality of Service (QoS), energy efficiency, and ultrahigh reliability [1]. In the current telecommunication sector, terrestrial base stations suffer from shortages of coverage and traffic capacity due to rapid population growth and insufficient infrastructure [2]. Furthermore, installing terrestrial base stations in temporary hotspots and disaster areas is not feasible and cost effective. On the other hand, unmanned aerial vehicles (UAVs), also known as drones, are used as aerial wireless communication platforms and provide a promising solution to the above problems [3]. The drone base station (DBS) has advantageous features, such as agility, high-probability Line-of-Sight (LoS) connections, maneuverability, and cost-effectiveness. According to [4], the integration of drones into 5G cellular systems is beneficial in most areas, including mobile communications, vertical industries, etc. Therefore, drone-aided communications are expected to become effective options in future 5G and beyond 5G heterogeneous wireless communication systems [4], [5]. This development is particularly significant when the conventional terrestrial communication infrastructure is insufficient to meet the demand for high-quality wireless services (e.g., special events and traffic offloading) in emergency communications and natural disaster scenarios, such as earthquakes, flooding, and tsunamis.

The drone's location can be dynamically adjusted over time to create favorable communication channels with ground mobile users (gMUs). However, the optimization of the 3-D trajectory of DBSs is a critical and challenging issue that is still unresolved and has not yet been fully investigated. The reason for this challenge is that air-to-ground communication link depends on not only the locations of users as existing cellular communication systems but also the communication environment, user distribution, and drone location [6].

Recently, several studies have examined the deployment of UAVs for wireless communication systems to provide effective area coverage and network connectivity [7]–[12], [18]–[22]. The differences between this article and the abovementioned

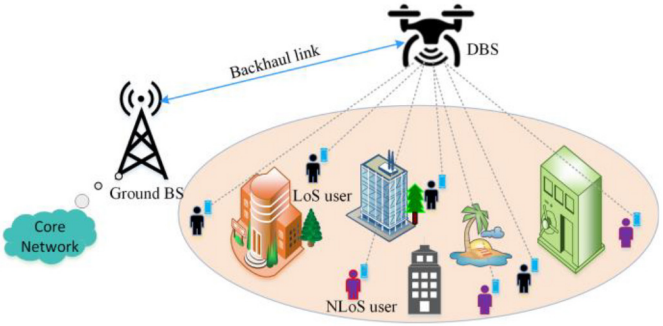

Fig. 1. Conceptual graphic of the problem to be solved, i.e., how to make the optimal decision regarding the movement of the DBS to serve LoS and NLoS users in urban environments.

works are summarized as follows: 1) unlike the static drone deployment problems investigated in [10]–[12], an autonomous control strategy for 3-D mobility of the DBS is studied here; 2) none of the previous studies have carefully addressed air-to-ground channel modeling in DBS wireless communications; and 3) many previous studies assumed that drone-aided communications were fully aware of the propagation environment. However, channel parameters and user location are limited for real-time deployment in a dynamic and complex environment through conventional learning approaches, which is the main focus of this article.

Motivated by this issue, this article focuses on developing a DBS mobility control strategy for *maximizing communication coverage and network connectivity* (DMC^3) in a drone-based wireless communications system,[1] which is shown in Fig. 1. To accomplish our goal, two steps are carried out. First, an online-learning-based link quality estimation model is developed to extract propagation characteristics from a dynamic environment using a time-series convolutional neural network (CNN). The preprocessed user channel information (including amplitude A_s and phase θ_s signals) are used as inputs to the learning model, and the corresponding link quality levels, i.e., the modulation and coding scheme (MCS) indices are used as target values. After extensive training, the well-trained link quality estimation model was obtained, which can be used to determine the link quality of the user's over time. The detailed learning model structure will be discussed in Section V-A. Morover, the proposed CNN-based link quality estimation model can achieve superior performance compared to [7] and [16]. In dynamic radio propagation environments, shallow learning methods [7], [16] are more susceptible to gradient disappearance, lack of adaptability and learning instability, particularly for large data samples. Second, we apply a deep reinforcement learning (DRL) algorithm to drive an autonomous drone mobility control strategy that meets the objective function in (3). The proposed method performs a continuous control task, where the DBS tracks or follows real-time users based on the received signals without requiring information on user locations to offer reliable wireless service.

[1]In this paper, the DBS mobility control strategy controls only the drone's position. In future work, we will take the speed and other parameters into account.

The novelty and *contributions* of this work can be summarized as follows.

1) In practice, the information of the user locations are not available to the network operator due to privacy concerns. Rather than using the user location [22], this article considers dynamic multiuser tracking based on observing the received radio signals of ground users in a target area.
2) Many existing works in the literature evaluated the link quality based on theoretical channel capacity using path-loss estimations [8]–[12], [17]–[22]. This article provides an accurate estimation of the link quality by considering path loss, slow fading, and fast fading.
3) We design an optimized drone mobility control framework by integrating time-series CNN and DRL algorithms to enhance the wireless communication service.
4) To verify the proposed method in practice, this article conducts air-to-ground channel measurements, rather than numerical simulations. The performance of the proposed framework is evaluated using experimental user channel data in a real environment.

The remainder of this article is organized as follows. In Section II, we discuss related works. Section III describes the system model and problem formulation of drone-aided wireless communication. Some preliminary CNN and DRL algorithms are introduced in Section IV. Then, Section V presents the proposed framework. Subsequently, Section VI reports the experimental results. Finally, Section VII makes some concluding remarks.

II. Related Works

Different studies have investigated air-to-ground channel modeling [6]–[9] and DBS deployment issues [10]–[12] to enhance the performance of cellular networks. In [6], the hybrid k-means and CNN algorithm are leveraged to accurately classify the link quality of ground users, and to further support the user selection for UAV-BS downlink. Wang *et al.* [7] proposed a k-means clustering algorithm to estimate the current air-to-ground communication link quality. Both Wu *et al.* [6] and Wang *et al.* [7] utilized the received signal strength of the users to extract and analyze the characteristics of the air-to-ground 3-D wireless channel. The relationship between the flight altitude of the DBS and its optimum user coverage was investigated by Al-Hourani *et al.* [8] and Mozaffari *et al.* [9]. Similarly, Alzenad *et al.* [10] and Chen *et al.* [11] studied a 3-D positioning problem of DBSs to maximize user coverage under different QoS constraints. On the other hand, Munaye *et al.* [12] adapted deep learning-based UAV deployment to increase the throughput of communication services. However, they assumed that user locations and UAV altitudes were fixed.

A model-free reinforcement learning (RL)-based scheme [13] has been demonstrated to be a successful approach for developing an autonomous drone mobility control strategy to overcome the limitations of the conventional optimization method. The DBS can learn the best trajectory policy by continuously interacting with the environment

without an explicit model via RL algorithms [14]. In fact, some prior works have considered RL-based drone-aided communication to enhance user throughput [15]–[22]. For instance, Abd-Elmagid *et al.* [15] applied an RL-based algorithm for UAV-assisted communications to achieve the minimum weighted sum of age-of-information. Luo *et al.* [16] utilized a two-stage learning method to optimize the horizontal position of drones. In the same context, Liu *et al.* [17] proposed a three-stage approach to control the dynamic movement of DBSs to maximize the mean opinion score. First, the *k*-means algorithm was applied to partition the ground users and assign drones to each cluster. Next, the authors employed a *Q*-learning approach to regulate the motion of the drone position. Finally, *Q*-learning-based user movement prediction was proposed to predict the coordinates of ground users at each *TS*. Additionally, Liu *et al.* [18] adopted a *Q*-learning algorithm to optimize the drone position to maximize user throughput and rate requirements. The GPS coordinates of real-time users were gathered from social media to model future user mobility. In [19], DRL was proposed for the energy-efficient control of drones by taking into account energy consumption and communication coverage at a fixed altitude. However, user movements are ignored in the proposed system model. Likewise, Li *et al.* [20] investigated a similar system model to that in [19] to maximize the sum rate of ground users. Shi *et al.* [21] investigated the 3-D DBS trajectory planning and scheduling through traditional optimization technique. Similarly, Shi *et al.* [22] developed a multidrone cell trajectory and resource allocation to maximize the accumulative network throughput for high-mobility users using DRL algorithms. The authors in [17]–[22] modeled the air-to-ground channel with mathematical channel modeling rather than real measurements. However, in dynamic wireless environments, the quality of a channel is depends on the user's mobility, the location of the DBS and the environment itself.

From the abovementioned research contributions, we recognize the following *challenges* that must be addressed in the design of 3-D aerial base station deployment.

1) *Air-to-Ground Channel Modeling:* Instead of learning from the environment over time, mathematical air-to-ground channel modeling was used in [17]–[22].
2) *Mobility of Users and DBSs:* Luo *et al.* [16], Liu *et al.* [19], and Li *et al.* [20] focused mainly on solving the drone deployment problem by assuming either the flight altitudes of drones or the ground users were static. In system deployment, the authors did not take into account that both users and DBSs are mobile.
3) *Learning Strategy:* Due to a large number of system state spaces, the decision space is large. Therefore, the conventional *Q*-learning algorithm [17], [18] faces the curse of dimensionality.

To sum up, this article aims to develop a learning-based drone mobility control strategy by considering the abovementioned issues to improve drone-aided wireless communication services. To the best of our knowledge, this is the first work to propose a drone mobility control framework by jointly considering the propagation movement, mobility of DBS, and online-learning-based link quality estimation of ground users.

TABLE I
LIST OF NOTATIONS

Notation	Explanation
H_a	Hotspot area
K	Number of users at each timeslot
TS	Timeslot
L_d	(x_d, y_d, h_d)
(x_d, y_d)	Latitude and longitude of drone d
h_d	Flight altitude of drone d
$m_{u,d}$	The MCS index of user u
m_{th}	Channel quality threshold
$\Gamma_{u,d}$	Association between user u and drone d
C_d	Coverage score of drone d
$\boldsymbol{S}$	System state space
$\boldsymbol{A}$	The space of all eligible actions
$\boldsymbol{P}$	Transition probability
$\boldsymbol{R}$	System reward
θ_s	Phase signal
A_s	Amplitude signal

III. System Model and Problem Formulation

In this section, we first introduce the system model for drone-aided wireless communication. Then, we formulate the optimized objective function of the DMC strategy.

A. System Model

We consider a drone-aided communication system as shown in Fig. 1, where a single drone d is deployed as an aerial base station to provide continuous wireless services for a set $K = \{1, 2, \ldots, u\}$ of u ground users distributed over a target hotspot area H_a.[2] The drone is connected to the core network via a ground gateway node. We assume that the DBS is aware of its own location through GPS and that each user $u \epsilon K$ is movable and distributed in a random manner in the target area. In addition, drone d can be located in 3-D space $L_d = (x_d, y_d, h_d)$, where (x_d, y_d) is the latitude and longitude coordinates of drone d and h_d is the flight altitude of drone d. For ease of reference, important notations used throughout this article are summarized in Table I. In the initial stage, the DBS is deployed at the center of the target H_a. Then, the DBS adjusts its 3-D position at each *TS* under a mobility control policy to maximize the communication coverage score and user throughput. To provide stable wireless communication and high data rates, the drone hovers at different altitudes based on the movement of ground users. In our proposed system model, we limit the minimum, $h_d^{\min}$ and maximum, $h_d^{\max}$ flight altitude of drone d.

[2]For our future work, we will consider and investigate the multiple DBS deployment problem to provide continuous wireless services in a scalable environment.

1) Air-to-Ground Channel Modeling: The ground users receive different radio propagation signals from the DBS in a dynamic environment. Typically, ground users can obtain LoS or approximate LoS (OLoS) communications in which the signal reaches the user directly and NLoS communications in which the signal is strongly reflected or diffracted due to obstacles. Accordingly, in the proposed drone mobility control problem formulation, we consider the air-to-ground channel characteristics and various environmental factors to attain on-demand wireless services. We chose users who received radio signals (i.e., A_s and θ_s) from the deployed DBS to analyze the air-to-ground wireless channel features.

2) User Mobility Model: In real time, users can move randomly and are distributed in the target H_a so that the DBS must travel/hover based on user traffic and movement to provide continuous wireless services. In our system model, we choose the random walk model, also called the Markovian mobility model [23], which can reflect the real movement of users. Therefore, each user can move randomly in the right, left, backward, and forward directions. Additionally, the user's speed is allotted as a pedestrian in the range $0 \leq u_{\text{speed}} \leq V_{\max,}$ which represents a user's maximum speed.

B. Problem Formulation

Our optimized objective is to find a mobility control strategy that specifies how the DBS moves in each time slot to maximize communication coverage with an acceptable network throughput of the gMUs in a continuous manner. The goal of the proposed DRL-DMC^3 is twofold. First, we defined the coverage score C_d of the DBS with respect to the gMUs. Let $\Gamma_{u,d} \in \{0, 1\}$ denote that the association between DBS d and user u is given by (1). If $\Gamma_{u,d} = 1$, user u is served by DBS d; otherwise, user u is not served by DBS d

$$\Gamma_{u,d} = \begin{cases} 1, & \text{the } u\text{th user is connected with drone } d \\ 0, & \text{the } u\text{th user is not connected with drone } d. \end{cases} \tag{1}$$

Accordingly, the coverage score C_d of the drone-aided communication system can be expressed as

$$C_d = \left(\prod /\mathrm{K}\right) \times 100\%, \sum_{u=1}^{\mathrm{K}} \Gamma_{u,d}, \Gamma_{u,d} \in 1 \ \forall t \tag{2}$$

where Π is the number of connected users in deployed drone d at each *TS* t. Second, we ensure the quality level of network connectivity or throughput of each user u at time t. We assume that each user $u \epsilon K$ received different channel link quality levels at each *TS* t from the aerial base station.[3] The proposed system increased overall system throughput to an acceptable level by maximizing the predicted link quality level of each user $m_{u,d}$, i.e., the MCS index. Accordingly, to associate with the MCS index, the received A_s information of the user is mapped to the modulation order, while the θ_s information is mapped to the coding rate. Then, the drone mobility control problem objective function (*DRL-DMC*^{3}) can be mathematically formulated as

[3]Assume the DBS knows the total number of ground users K in the target hotspot area at each timeslot.

DRL-DMC^3*:*

$$\arg\max_{L_d} \sum_{u=1}^{\mathrm{K}} \Gamma_{u,d}(L_d) \times m_{u,d}(L_d) \tag{3}$$

$$\text{s.t.} \quad \Gamma_{u,d} \in \{0, 1\} \ \forall u \in K, \forall t \tag{3a}$$

$$P(C_d | u \in K) \cong 100\% \tag{3b}$$

$$m_{u,d} \geq m_{th} \ \forall u \in K \ \forall m \in M \ \forall \mathrm{t} \tag{3c}$$

$$M = 0, 1, 2, \ldots, m \tag{3d}$$

$$h_d^{\min} \leq h_d \leq h_d^{\max} \ \forall t \tag{3e}$$

$$0 \leq u_{\text{speed}} \leq V_{\max} \tag{3f}$$

where (3a) denotes the communication association status of the user u either served by the drone (i.e., $\Gamma_{u,d} = 1$) or not (i.e., $\Gamma_{u,d} = 0$); constraint (3b) ensures that drone d covers as many users as possible in the target H_a; equation (3c) ensures the channel quality of each user is greater than or equal to a threshold value m_{th}; equation (3d) ensures the MCS index of each user u is chosen from the given MCS set (see [24, p. 2] Table I for details), where M denotes the total number of MCS indices; and (3e) and (3f) indicate the altitude constraint of the drone and the users' speed limit, respectively.

IV. Theoretical Preliminaries

In this section, we provide a brief overview of the time-series CNN and DRL algorithms before presenting the proposed method.

A. General Convolutional Neural Network Framework

The CNN network was recently introduced as a deep learning approach [25] that achieves great improvement in recognition and classification for time-series and image data [26]. It is a hierarchical feedforward artificial neural network that consists of an input layer, several hidden layers, and an output layer. Convolutional layers, pooling layers, and fully connected (FC) layers are included in the hidden layers. A CNN architecture consists of two main components, namely, the feature extractor and classification stages [25]. The convolutional layers can be used to generate high-level features from the input feature map using an $M \times M$ convolutional kernel, called the output feature map. The pooling layers are optional layers that aim to reduce the dimensionality of a feature map to make the network more robust. Max pooling and average pooling are the most common pooling operations [25]. After a series of convolutions and pooling operations is performed, FC layers serve as the classifier of a neural network and output class scores.

Thus, in this article, a time-series CNN-based LQE model is proposed that estimates each user's link quality over time by predicting its MCS index in a dynamic wireless environment. The proposed time-series CNN-based LQE model will be explained in Section V-A.

B. General Reinforcement Learning Framework

RL [13] is a dynamic learning method that can automatically learn from the current situation and adapt the

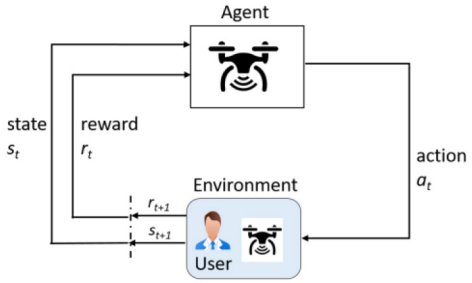

Fig. 2. Illustration of agent–environment interaction in MDP.

environment's changes to solve decision-making problems, e.g., real-time user throughput maximization and resource allocation issues in aerial base station networks [14]. RL is the only machine learning method that can be learned without a dataset. However, the RL agent generates data when it interacts with the environment. The proposed DMC process can be modeled as a Markov decision process (MDP) in an ergodic way and therefore can be processed by the DRL algorithm. MDP is defined as a fourfold ($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{P}$, $\boldsymbol{R}$), where $\boldsymbol{S}$ is the system state space, $\boldsymbol{A}$ is the system action space, $\boldsymbol{R}$ is the reward function, and $\boldsymbol{P}$ is the state transition probabilities that specifies the dynamics. At each timeslot t, the agent or the learner observes the system state s_t from the target environment and then executes a system action a_t from $\boldsymbol{A}$ according to a policy $\pi(s_t, a)$. The agent will obtain a numerical reward signal r_t and transits to a new system state s_{t+1} with $\boldsymbol{P}(s_{t+1}|s_t, a_t)$. The goal of MDP is to determine an optimal policy $\pi^* = \arg\max_{a\in A} Q^*(s_t, a)$ that maximizes the long-term reward. Fig. 2 shows that the RL agent continuously interacts with the environment to learn from past experiences. In the RL algorithm, a *policy* is a function that maps the RL agent observed system states to the legitimate actions to maximize the accumulated reward, which can be denoted as $\pi(\mathrm{s}) \in \boldsymbol{A}$ $\forall \mathrm{s} \in \boldsymbol{S}$. Additionally, the *value function* provides a prediction of the future reward that evaluates the goodness and badness of a particular action in a given system state.

Q-learning is a well-known RL algorithm [13]. In Q-learning, a Q-table or look-up table is used to store Q values for all combinations of $\langle s, a\rangle$ pairs. To maximize the long-term accumulative reward, the agent needs to select the best action $a \in \mathbf{A}$ for a given state $s \in \mathbf{S}$ under policy π^*. To find the optimal policy, the Q-values of the $\langle s, a\rangle$ pairs in the look-up table are updated based on the experiences of the agent using the Bellman equation as expressed by [27, p. 1297]

$$Q^*(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r(s_t, a_t) + \gamma \arg\max_{a'\in A} Q(s_{t+1}, a')\right]. \tag{4}$$

Note that $\alpha \in (0, 1]$ refers to the learning rate, $r(s, a)$ is the received reward, and a' is the system action that is subject to the maximum state–action value at s_{t+1}. $\gamma \in [0, 1]$ is the discount factor for weighting future rewards. If γ is close to 0, the RL agent will focus on immediate or short-term rewards, whereas if γ approaches 1, the RL agent will focus on long-term rewards. Therefore, by referring to the look-up table, the agent can decide its optimal action based on any state to obtain an accurate decision-making strategy. However, the application of conventional Q-learning algorithms is limited due to the curse-of-dimensionality problems [14]. When we consider large-scale environments, finding the optimal policy using Q-learning is challenging, as: 1) the RL agent may not be able to explore several $\langle s, a\rangle$ pairs because of the large look-up table, resulting in reduced learning performance and algorithm efficiency and 2) the storage size of the Q-table is unrealistic.

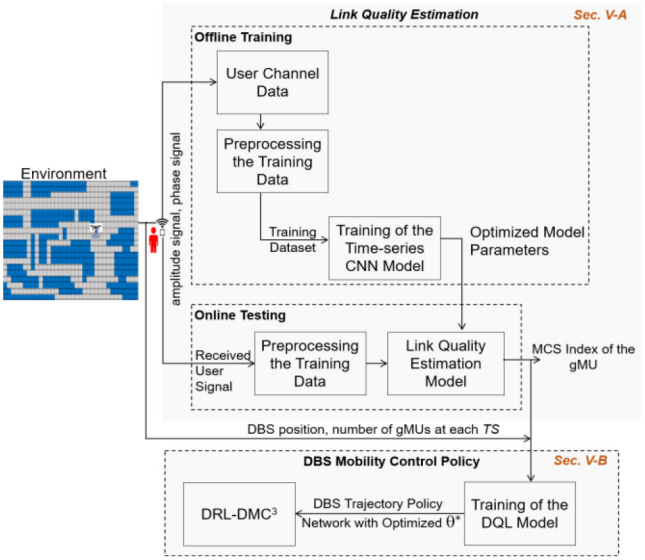

Fig. 3. Overall system architecture of the proposed DRL-DMC^3.

To address these two issues, deep neural networks (DNNs) add intelligence to the traditional Q-learning method. Consequently, in this article, we propose a DRL specifically deep Q-learning (DQL) algorithm to derive an approximate value of $Q^{*(}s_t, a; \theta)$ using a DNN instead of a Q-table. $Q^{*(}s_t, a, \theta)$ represents the maximum expected long-term reward for $\langle s, a\rangle$ pairs, and the optimal policy is $\pi^* = \arg\max_{a\in A} Q^{*(}s_t, a; \theta)$ for a given state $s \in \mathbf{S}$. This algorithm accelerates the learning process and improves RL performance [14]. We will explain the DRL-based DMC mechanism in detail in Section V-B.

V. Proposed Solution

This section describes the methodology of the DBS mobility control strategy in a drone-aided wireless communications system, which is shown in Fig. 3. Specifically, the proposed method consists of two parts, namely, the construction of a learning-based link quality estimation model and a DRL-based drone mobility control strategy.

A. Time-Series CNN-Based Link Quality Estimation Model

In the current long-term evolution (LTE) communication systems, the user channel quality level is measured using the channel quality indicator (CQI), which only considers the signal strength. The range of CQI values is between 0 and

Fig. 4. Field measurement setup at the NTUT campus.

15 and is determined mainly by the signal-to-interference-plus-noise ratio (SINR) [28]. The SINR parameter may be adequate for regulating the channel quality for fixed base station communications due to their static properties. However, it is difficult to identify the A2G channel modeling of the flying base station based on the SINR parameter only to increase the overall system efficiency due to the mobility features [29]. Therefore, in this article, we adopt a time-series CNN algorithm to build an accurate LQE model that identifies the link quality of the gMUs over time using the received amplitude and phase signals. The proposed time-series CNN-based link quality estimator process consists of the following steps.

1) User Data Collection: The experiments were conducted at the National Taipei University of Technology (NTUT) campus and surrounding roads to obtain user channel data, as shown in Fig. 4. Additionally, the target H_a includes many buildings and trees. During user channel data collection, we consider two kinds of user mobility to reflect the real scenario. The first is pedestrians (users) moving at a relatively slow speed walk (<1.4 m/s), while the second is a fast speed walk (between 1.4 m/s and 2.5 m/s). In addition, the DBS flight altitude is limited to between 30 and 50 m. In this article, the radio signal is the OFDM signal. We transmit the OFDM signal at the 860-MHz frequency band and a bandwidth of 3.2 MHz. We also used universal software radio peripheral (USRP) B210 from National Instruments to convert the OFDM baseband signal to a random forest (RF) signal. Table II summarizes the experimental parameters that were used in these experiments.

For this work, we use the pilot symbol to measure channel data in the OFDM communication system using the following:

$$x(t) = Ae^{j(\omega_0 t+\theta)} \tag{5}$$

where $x(t)$ represents pilot data, ω_0 is the radian frequency, and A and θ are the amplitude and phase in radians, respectively. Each OFDM signal has 273 complex original pilot symbols as shown in the following:

$$H_{\text{Pilot}} = \left[A_1 e^{-i\theta_1}, A_2 e^{-i\theta_2}, A_3 e^{-i\theta_3}, \ldots, A_{273} e^{-i\theta_{273}}\right]. \tag{6}$$

The receiver receives a complex number that contains the real (Re) and imaginary (Im) values of the individual OFDM

TABLE II
EXPERIMENTAL PARAMETERS

Parameter	Value
Channel Bandwidth	3.2 MHz
Carrier Frequency	860 MHz
Transmitter Power	30 dBm
Antenna	dipole
Modulation	OFDM/QPSK
FFT Size	2048
Subcarrier $\Delta\mathbf{f}$	1.5 kHz
Pilot Subcarrier	273
Guard Band Subcarrier	143

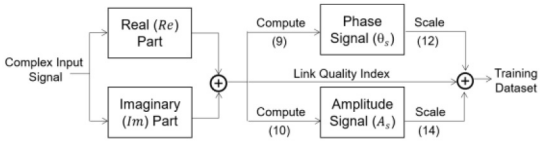

Fig. 5. Block diagram of the proposed data preprocessing method.

signal pilot. Since the training process is performed by a personal computer, the value must be converted so that a computer can process it. According to the Yula formula (7), each signal can be written as a complex number as follows:

$$A_n e^{-i\theta_n} = A_n \cos\theta_n - iA_n \sin\theta_n. \tag{7}$$

Among various signal parameters, we choose the amplitude and phase signals to implement the online-learning-based LQE model. Therefore, we need to compute the phase signal (θ_s) and amplitude signal (A_s) of each user from the complex pilot symbol. First, we separate the Re and *Im* parts from the complex pilot symbol. Then, we compute θ_s (8) and A_s (9) using Euler's formula as follows:

$$\theta_s = \arctan\left(\frac{\sin\theta_n}{\cos\theta_n}\right) \tag{8}$$

$$A_s = \sqrt{Re^2 + Im^2}. \tag{9}$$

The amplitude and phase signals can be denoted in the matrix form as

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_N \end{bmatrix} \tag{10}$$

where $\mathbf{R} \in R^{N\times F}$, N denotes the total record of the user channel data, F is *the number of features*, and $\mathbf{r}_i$ is a row vector containing A_s and θ_s in the ith row. In this case, each $\mathbf{r}_i$ is assigned to one of α classes that can be represented as $f : \mathbf{R} \to \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_\alpha\}$. In our case, $C_\alpha = M$.

2) Data Preprocessing: Fig. 5 depicts a block diagram of the data preprocessing stage of the proposed method to make the dataset appropriate for training machine formats, which represents the concatenation of the θ_s and A_s channel matrices. Accordingly, the min–max normalization technique is used to speed up the training process and enhance the performance of

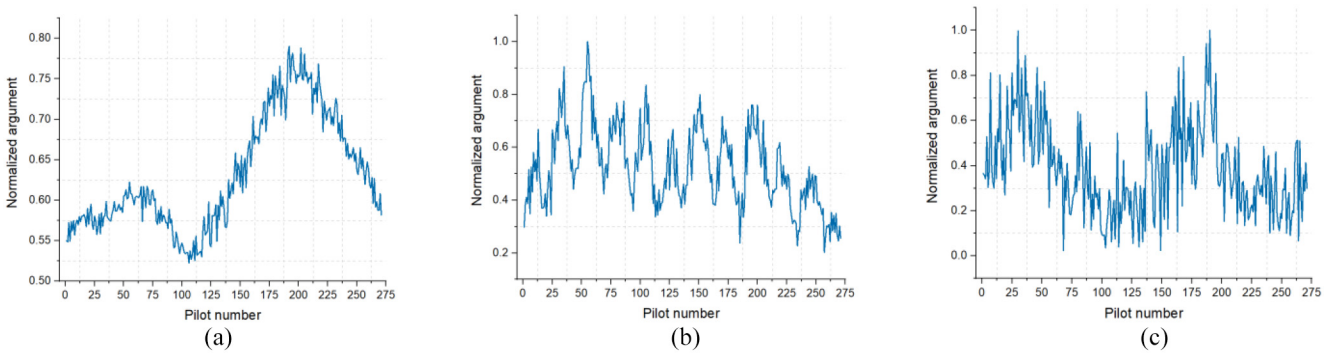

Fig. 6. Normalized amplitude changes in the OFDM signal in 273 pilots. (a) LoS. (b) OLoS. (c) NLoS.

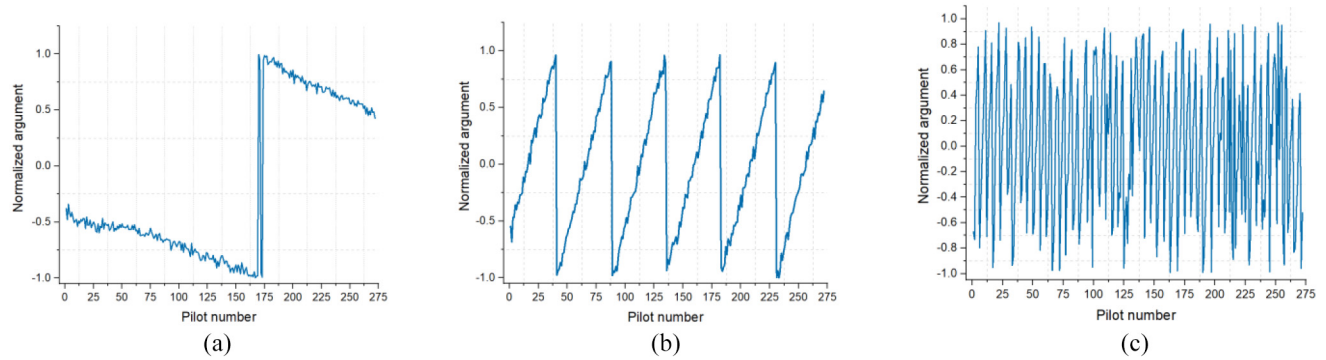

Fig. 7. Normalized phase changes in the OFDM signal in 273 pilots. (a) LoS. (b) OLoS. (c) NLoS.

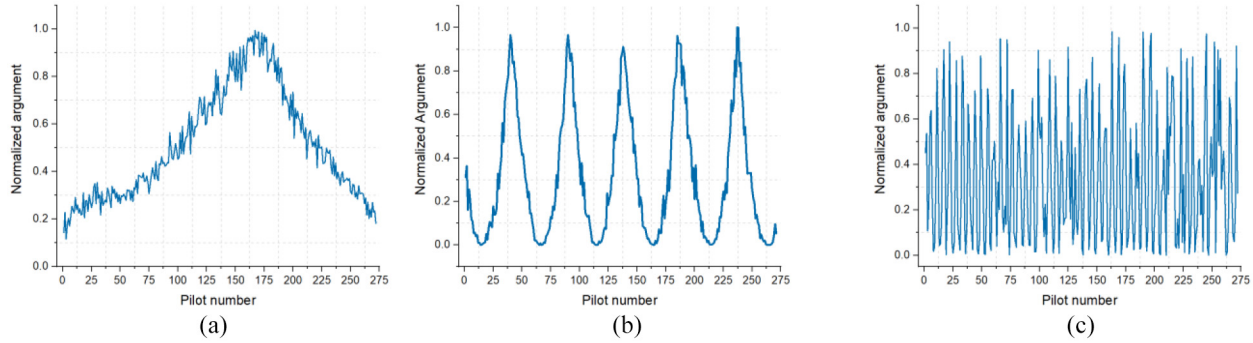

Fig. 8. Continuous normalized phase changes in the OFDM signal in different channel states during the pilot. (a) LoS. (b) OLoS. (c) NLoS.

the neural networks [30]. Consequently, θ_s is rectified between $[-1, 1]$ using (11), which is adapted from a general range transformation normalization method (12) [31]. However, A_s does not include negative values; therefore, we normalized the data over the interval [0, 1] using (13)

$$\widetilde{r_i} = 2\frac{r_i - r_{\min}}{r_{\max} - r_{\min}} - 1 \ \forall r_i \in \mathbf{R} \tag{11}$$

$$\widetilde{r_i} = (b-a)\frac{r_i - r_{\min}}{r_{\max} - r_{\min}} + a \ \forall r_i \in \mathbf{R} \tag{12}$$

$$\widetilde{r_i} = \frac{r_i - r_{\min}}{r_{\max} - r_{\min}} \ \forall r_i \in \mathbf{R} \tag{13}$$

where $\widetilde{r_i}$ represent the normalized feature vectors ($\widetilde{r_i} \leq i \leq N$) and $r_{\max}$ and $r_{\min}$ are the maximum and minimum values, respectively, among the 273 θ_s and A_s OFDM signals.

Figs. 6 and 7 show the normalized amplitude and phase changes in the pilot OFDM signal in different channel states, respectively. The x-coordinate represents the pilot number, and the y-coordinate is the normalized argument. However, a cliff-like tendency also exists in the preprocessed θ_s. To avoid the influence of this cliff tendency on the estimation model learning, we square the obtained phase argument so that the cliff at the junction of -1 and 1 can be changed to a continuous value, as shown in Fig. 8. Finally, we concatenate the preprocessed θ_s and A_s via time synchronization to create an appropriate training dataset for the neural network, as described in Algorithm 1.

The MCS (i.e., CQI) index is a metric used to measure the link quality for certain channel conditions. To determine the link quality of each gMU, the received A_s and θ_s information must be mapped to the MCS index. Typically, the

Algorithm 1 Preprocessing User Channel Data

Input: User channel data.

1: Collect historical channel data of users using (6):

$$\mathrm{H_{Pilot}} = \left[A_1 e^{-i\theta_1}, A_2 e^{-i\theta_2}, \ A_3 e^{-i\theta_3}, \ldots, A_{273} e^{-i\theta_{273}}\right];$$

2: Separate the real (*Re*) and imaginary (*Im*) part from the complex number;

3: Compute phase signal θ_s using (8) and amplitude signal A_s using (9), respectively:

$$\theta_s = \arctan\left(\frac{\sin\theta_\mathrm{n}}{\cos\theta_\mathrm{n}}\right), A_\mathrm{s} = \sqrt{Re^2 + Im^2};$$

4: Preprocess the θ_s and A_s to create an appropriate structured training dataset format for machine learning using (11) and (13), respectively:

$$\widetilde{r_i} = 2\frac{r_i - r_{min}}{r_{\max} - r_{min}} - 1 \ \forall r_i \in \mathrm{R};$$
$$\widetilde{r_i} = \frac{r_i - r_{min}}{r_{\max} - r_{min}} \ \forall r_i \in \mathrm{R},$$

5: Concatenate the A_s and θ_s channel matrix;

Output: Preprocessed A_s and θ_s.

A_s information is mapped to the modulation order, while the θ_s information is mapped to the coding rate. Accordingly, the A_s information is labeled in the modulation order using the SNR values, whereas we apply the *k*-means algorithm to cluster θ_s data into the coding rate. Finally, we map the MCS index based on the labeled modulation order and coding rate to build the LQE model.

3) Offline Training of the LQE Model: We denote the sequence of F input data vectors as $X = (r_1, r_2, \ldots, r_F)$ and the corresponding channel quality level or MCS index as y. The preprocessed channel information is therefore fed as input into the time-series CNN algorithm, and the corresponding link quality level is used as the target value to train the proposed LQE model. Therefore, each user MCS index is mapped to the corresponding input features, i.e., (X, y), where X is the feature matrix and y is the output vector, which can be expressed as

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1F} \\ x_{21} & x_{22} & \cdots & x_{2F} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NF} \end{bmatrix} \text{ and } y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ C\alpha \end{bmatrix} \quad (14)$$

where F is the number of features in the training dataset and N is the total number of time-domain samples.

A schematic of an online-learning-based link quality estimation model is shown in Fig. 9. The proposed time-series CNN network consists of four convolutional layers and one FC layer. To find the best hyperparameters of the CNN network, such as convolutional layers, batch sizes, epochs, and activation functions, we used the trial and error method [32]. The convolutional layer extracts silent and discriminative features from the input feature matrix, X, via a convolutional kernel (filter) to differentiate link quality levels. Let w be a convolution kernel window that is applied across X. The proposed model

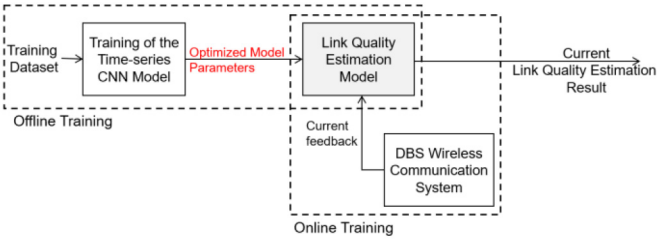

Fig. 9. Schematic of the online-learning-based LQE model in DBS wireless communication.

uses a 5 × 5 convolution kernel size w to achieve good channel estimation performance with low complexity. After applying w to X, the output feature map $X*$ can be obtained as follows:

$$X* = f(X \otimes w + b) \quad (15)$$

where $f(.)$ is the activation function, and b is a bias matrix. Then, we employed a nonlinear rectified linear unit (ReLU) activation function to provide nonlinear behavior

$$f_{\mathrm{ReLU}}(\Lambda) = \begin{cases} \Lambda, & \text{if } \Lambda > 0 \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where Λ is the argument of the function. A max-pooling operation with a window pooling shape of 2 × 1 is then applied using (17) to capture the best features in $X*$ and make the network more robust. Additionally, this operation reduces the number of network parameters, thus significantly reducing the training computation time

$$X* = \max(X*). \quad (17)$$

In addition, we use the minibatch hyperparameter to speed up the training process instead of feeding all of the training samples. After executing a series of convolutional, nonlinear, and pooling layers, the layer is transformed into a 1-D array vector and connected to the FC layer. In this layer, every input feature is connected to the target outputs of the network by a learnable weight. The FC layer has M output neurons which is the same as the number of MCS indices. Hence, we used a softmax nonlinear activation function for the final output neurons of the network to predict the probability value of each MCS index as follows:

$$y^{(j)} = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_i}} \quad (18)$$

where $y^{(j)}$ is the output of the jth neuron in the output layer, j is the index of the output neurons, and M is the total number of MCS indices. This activation function maps the output in the range of [0, 1]. To update and optimize the network parameters of the proposed LQE model, we employed the cross-entropy loss function that is defined as

$$L(\theta_c) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log y'_{ij} \quad (19)$$

where θ_c represents the model parameters of the CNN in the training time, N is the number of all samples for classification, M is the number of MCS indices, y_{ij} is the actual label

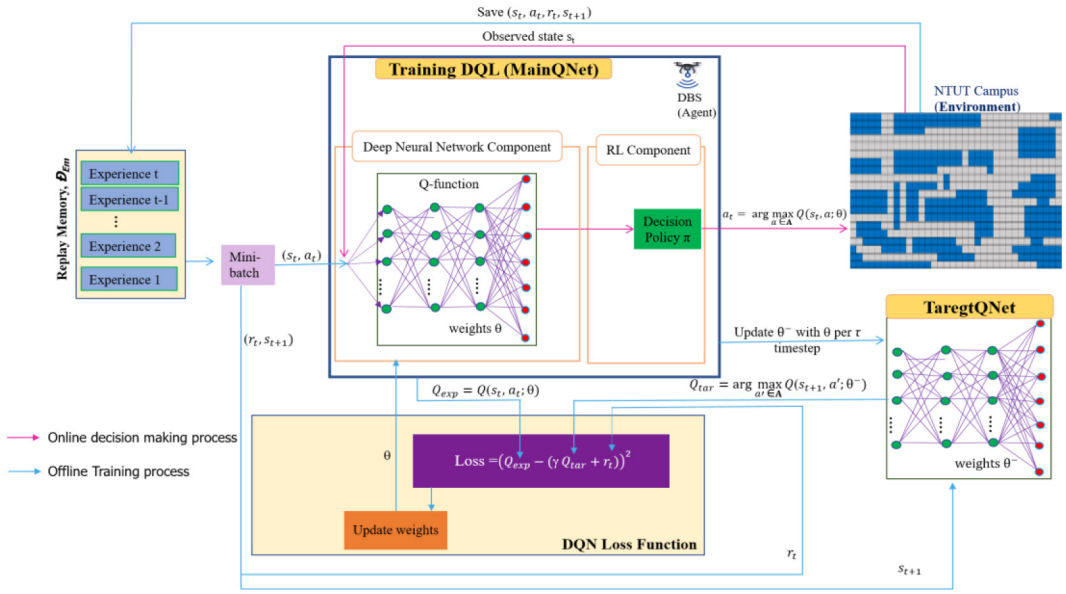

Fig. 10. Illustration of the proposed DRL agent design for DMC strategy.

Algorithm 2 Time-Series CNN-Based Link Quality Estimation

Input: Training dataset

1: Initialize the learning parameters of the CNN;
2: **For** *epoch*: = 1 to T **do**
3: Train the LQE model;
4: Update the network weights w and b of the CNN by minimizing the loss function using (19);
5: **end for**

Output: Trained LQE model with optimized w and b.

Algorithm 3 Online Link Quality Estimation Application

1: Load the **Algorithm 2** model;
2: **For** each *TS* t **do**
3: each user receives real-time radio signals from the deployed DBS;
4: estimate the link quality of the ground users;
5: $t = t + 1$;
6: **End for**

value and y'_{ij} is the estimated MCS index. Consequently, we obtain well-trained LQE model with optimized weights using Algorithm 2.

4) Online Link Quality Estimation Application: Once the offline training is completed, the well-trained LQE model is deployed at the DBS to estimate the link quality of ground users over time as described in Algorithm 3. The estimated link quality of ground users is then fed as an input into the proposed DQL algorithm to implement an optimized drone mobility control strategy for drone-aided wireless communications.

B. DRL-Based DBS Mobility Control Strategy

We propose a DRL-based DMC policy to provide better wireless communication services to all ground users based on the received radio signals rather than user locations. The DQL comprises the DNN and RL modules as illustrated in Fig. 10. It uses DNN to drive the approximate Q-values, i.e., determined $Q^*(s, a)$ by DNN instead of a look-up table, and inputs them into RL for decision-making based on the DNN results. In our proposed system, the DRL agent is a DBS whose system state space, system action space, and reward function are defined as follows.

1) System Action Space: The actions represent the flying direction of the DBS. The DBS updates its locations under the mobility of gMUs to provide better communication services. In this article, the drone can fly in discretized directions and nine actions were defined to optimize the objective function.[4] Hence, the action space of the proposed model can be expressed as follows:

$$\boldsymbol{A} = \{left, right, up, down, left\text{-}up, right\text{-}up \\ left\text{-}down, right\text{-}down, no\ movement\}. \quad (20)$$

The drone takes action $a_t \in \boldsymbol{A}$ based on its learning experiences and policy π at any *TS* t.

2) System State Space: The DBS is modeled as a DRL agent. The system states are the 3-D location of the drone, the user link quality, and the current number of users available in a target H_a. Hence, at each TS, the DBS observes the system state s_t from an unknown environment to take an optimal action as follows:

$$s_t = \{(L_d, K, m_{u,d}) \\ L_d \in (x_d, y_d, h_d), K \in \{0, 1, \ldots, u\}, m_{u,d} \in \{1, 2, \ldots, 6\}\} \quad (21)$$

[4]In the future, we will consider the mobility of DBS to be constrained to 360 degrees of angles instead of nine directions.

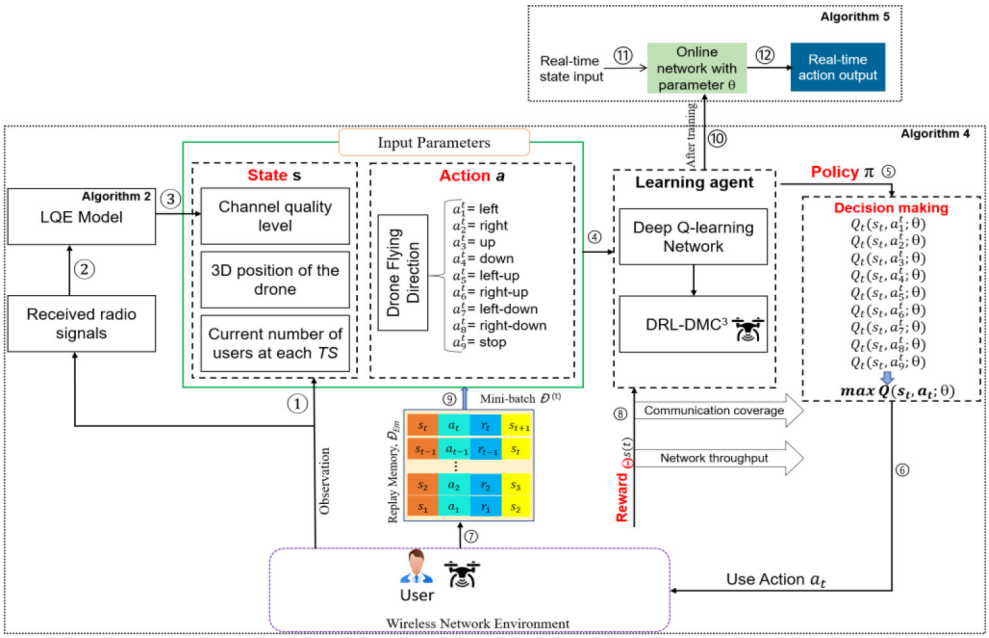

Fig. 11. Proposed DRL-DMC^3 model training process.

where L_d is the 3-D position of the DBS and K is the current number of users at each *TS* in the target H_a. $m_{u,d}$ is the MCS index of user u, and we defined a threshold state value m_{th} (in our case, $m_{th} \geq 6$) to determine whether the user communication channel is good or bad. The input of the proposed DNN in a DQL algorithm (DNN^Q) is the $\langle s, a\rangle$ pairs, while the output is a scalar approximated Q-value for all possible actions, $Q(s, a; \theta)|a \in \boldsymbol{A}$, where θ is a vector containing the network parameter (i.e., weights) of the DNN^Q.

3) Reward Function: The reward is a function of the system state and system action that determines the performance of the DMC strategy that should consider both the communication area coverage and network throughput. During the training phase, the DRL agent will receive a corresponding reward from the reward function in each TS. The DRL agent then updates its policy π based on the received reward to optimize the DMC strategy network. This network can select the best system actions at different system states to offer a high reward. Therefore, the reward function is intended to optimize the objective function problem in (3). Let $\Theta_{s(t)}$ denote the value of the objective function in DRL-DMC^3, which can be defined as

$$\Theta_{s(t)} = \left(\{m_{u,d}\}, \{C_d\}\right) \tag{22}$$

where $(\{m_{u,d}\}$ and $\{C_d\}$ are given by the system state s_t according to the definition in (21). Once an action is taken, the state transitions from the current state s_t to the next state s_{t+1}. Then, based on feedback from the environment,$\Theta_{s(t)}$ and $\Theta_{s(t+1)}$, we define a reward of the $\langle s_t, a_t\rangle$ pairs to assess how well the intended goal is achieved as follows:

$$\begin{aligned} r_t &= r_{(s(t),a(t),s(t+1))} = -1, \text{ if } \Theta_{s(t+1)} < \Theta_{s(t)} \\ r_t &= r_{(s(t),a(t),s(t+1))} = 1, \text{ if } \Theta_{s(t+1)} > \Theta_{s(t)} \\ r_t &= r_{(s(t),a(t),\ s(t+1))} = 0, \text{ if } \Theta_{s(t+1)} = \Theta_{s(t)}. \end{aligned} \tag{23}$$

TABLE III
DRL AGENT DESIGN

Environmental Variables	System Equivalent
Agent	DBS
State $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_t}\}$	$\{(L_d$, K , $m_{u,d})\}$
Action $\boldsymbol{A}$	{left, right, up, down, left-up, right-up, left-down, right-down, no movement}
Reward r	($\{m_{u,d}\}$ and $\{C_d\}$ that satisfy in (3c) and (3b), respectively)
$m_{u,d}$	MCS index of user u
K	Number of users at each *TS* at a target H_a
L_d	3D location of DBS (x_d, y_d, h_d)

Table III summarizes the DRL agent design environmental variables.

4) Training Procedure: The structure of the training process of the proposed DRL-DMC^3 framework is presented in Fig. 11 and consists of the input parameters, online DQL and offline memory experience reply ($\mathcal{D}_{Em}$) phases. The training process of the proposed DRL-DMC^3 method follows Algorithm 4. The complexity of a DRL algorithm is O($Te^{\max}$), where T is the number of training time steps per episode and $e^{\max}$ is the total number of episodes. In particular, initially, we perform the initialization in lines 1–4. The proposed algorithm requires initializing a system action space A, learning rate β, discount factor γ, and the initial position of the DBS. For the first time, we do not know the location and distribution of users. Therefore, we assume that the drone is initially deployed at the center of target H_a. This assumption helps achieve faster network convergence and optimization.

Then, the training process begins from line 5, where the drone first obtains its observation from the environment using (22) ① and the drone controlled by DRL agent ⑤. Moreover, the link quality of each user ③ is computed using a LQE

Algorithm 4 DRL-Based Solution for Autonomous DMC Strategy (Training Phase)

Input: System action space ***A;*** mini-batch size M_b; discount factor γ; learning rate β; exploration coefficient ϵ.

/* *Initialization* */

1: Initialize a $\mathcal{D}_{Em}$ that stores past drone experiences;
2: Initialize the parameters of DNN^{Q} Q with random weights θ;
3: Initialize the target Q-network $\hat{Q}$ with random weights $\theta^{-} = \theta$;
4: Initially, deploy the DBS at the center of target H_a;

/* *Parameter updating to train the DRL* */

5: **For** episode: =1 to e^{max} **do**
6: Set t: = 1 and the DBS perceive the initial system state s_1 after initializing the environment;
7: Update DBS location (x_d, y_d, h_d);
8: Input the received radio signals into **Algorithm 2** to estimate the user's $m_{u,d}$;
9: **For** t: = 1 to T **do**
10: The drone chooses action a_t at system state s_t based on the ε-greedy approach from Q(s_t, a; θ) $a \in \boldsymbol{A}$ using (24):
11: Generate a random number $\rho \in [0,1]$
12: **if** $\rho > \epsilon$ **then**
13: choose $a_t \leftarrow \arg\max_{a \in A} Q(s_t, a; \theta)$; /* *exploitation* */
14: **else**
15: choose $a_t \leftarrow$ randomly from $\boldsymbol{A}$; /* *exploration* */
16: **end if**
17: Execute the selected action a_t;
18: The DBS observes the next system state s_{t+1} and obtains the immediate reward $_{s(t)}$ according to (3). Then, execute (23) to balance the long-term rewards;
19: **if** DBS flies beyond target H_a **then**
20: cancel the corresponding action and update s_{t+1} accordingly;
21: **end if**
22: Store E_t in the offline experience memory $\mathcal{D}_{Em}$ where $E_t = (s_t, a_t, r_t, s_{t+1})$;
23: Randomly sample a mini-batch of M_b transitions from a local memory $\mathcal{D}_{Em}$;
24: Calculate the target Q-value using (26);
25: The DRL agent updates the weights θ of the trained DNN^{Q} of Q(.) by minimizing the loss function using (25);
26: **if** t modulo τ == 0 /* *update TargetQNet* θ^{-} *every* τ *steps* **then**
27: Set $\theta^{-} = \theta$;
28: **end if**
29: t $\leftarrow$ t + 1;
30: **end for**
31: episode $\leftarrow$ episode + 1;
32: **end for**
33: Store the trained Q-network;

Output: Optimal drone mobility control policy π^{*}.

model by feeding the amplitude and phase radio signals as input ②, as explained in Section V-A3. The DQL algorithm takes the input in the state–action tuple $\langle s_t, a_t \rangle$ ④ to train the DMC strategy. Then, based on the achieved observation, the drone selects the proper action a_t ⑥ using policy π to make a decision. In the action selection mechanism, we adapt an epsilon (ε)-greedy approach to balance the exploitation of past experiences with the exploration of its target environment to obtain better rewards (lines 10–17). When the network starts learning, the DRL agent does not have confidence in the estimated value of the Q-function because it may not have visited some of the $\langle s, a \rangle$ pairs in the target H_a. Thus, the DRL agent needs to explore the environment to some degree to avoid being trapped in nonoptimal policies [13]. Therefore, the action selection probability of the RL agent is expressed as

$$a_t = \begin{cases} \arg\max_{a \in \mathbf{A}} Q(s_t, a; \theta), & \text{with probability } 1 - \varepsilon \\ \text{random } a \in \mathbf{A}, & \text{with probability } \varepsilon \end{cases} \quad (24)$$

where the ε parameter refers to the probability of choosing to explore. When $\epsilon = 1$, the DRL agent explores 100% of the environment rather than exploiting it. The DRL agent chooses either exploration or exploitation based on the randomly generated number ρ between 0 and 1. As stated in line 12, if $\rho > \epsilon$, then the agent will choose exploitation as the next state, which means that the DRL agent selects an action from the Q-table with the highest Q-value for the current system state, $a_t = \arg\max_{a \in A} Q(s_t, a; \theta)$. Otherwise, the next action is chosen randomly from action space $\boldsymbol{A}$ with probability ϵ to explore more actions to enhance long-term rewards (line 15). This decision is important to allow the drone to explore and discover new states to develop an optimal DMC strategy.

Moreover, the DMC strategy can explore and learn the environment using a high exploration coefficient (ε close to 1) at the beginning of training. Additionally, the DBS will remain at its current location if the next location is obtained beyond the target H_a (lines 18–21). In this article, we set the initial $_i$ and final exploration $_f$ probabilities to 0.95 and 0.01, respectively. After executing a_t, we obtained the reward $\Theta_{s(t)}$ ⑧ and the next system state s_{t+1} (line 18). To make the proposed network more stable, an experience reply memory technique and the target DQN-network (TargetQNet) are utilized for better learning convergence [33], [34]. After obtaining a new training experience s_t, a_t, r_t, and s_{t+1}, the DRL agent is stored in an *offline experience reply memory*, $\mathcal{D}_{Em}$ ⑦, which can store the experiences in a first-in-first-out (FIFO) queue fashion (line 22). In the experience replay memory, the DRL agent experiences at each *TS* (T steps at each episode) is saved as $E_t = (s_t, a_t, r_t, s_{t+1})$ in the database $\mathcal{D}_t = (E_1, E2, \ldots, E_t)$ when executing the DQL algorithm. Therefore, the system saves the assessment of previous $\langle s, a \rangle$ pairs and their resulting costs to an experience replay memory in each *TS*. Once sufficient experiences have been collected in $\mathcal{D}_{Em}$, we sample a minibatch of M_b transactions that is extracted randomly from the memory pool $\mathcal{D}_{Em}$ ⑨ to train the DQL network from past experiences rather than training with a single experience observed from the environment (line 23). TargetQNet has the same structure as the main or trained DQL network (MainQNet), which is used to evaluate the target value (y^{Tar}) for the online DNN^{Q} training. TargetQNet copies the evaluated MainQNet weights in every τ steps (in our case experiments, = 75) for learning stability. The TargetQNet weights θ^{-} are used to evaluate the action selected by the MainQNet weights, θ. To obtain the optimal DMC strategy, DNN^{Q} updates the weights θ for each tuple within the sampled minibatch to minimize the prediction error $Q(s_t, a, \theta)$ in (25) (lines 24–32) using a gradient descent method. The loss function L(.) can be defined as (25) using the Bellman equation [34]

$$L(\theta) = E\left[\left(y^{\text{Tar}} - Q_{\text{exp}}\right)^2\right] \quad (25)$$

Algorithm 5 DMC Strategy (Testing Phase)

Input: Trained Q-network; system state s_t.

1: Load the trained Q-network from **Algorithm 4**;
2: $t = 0$;
3: **for** each *TS* t **do**
4: The DBS select an action a_t for system state s_t that maximizes $\Theta_{s(t)}$;
5: update the system state; **/*** *until terminate* ***/***
6: $t = t + 1$;
7: **end for**

Output: Maximize communication coverage and user throughput;

where the target value y^{Tar} can be estimated by [14]

$$y^{\text{Tar}} = r(s_t, a_t) + \gamma Q_{tar} \tag{26}$$

where $Q_{tar} = \arg\max_{a' \in A} Q(s_{t+1}, a'; \theta^-)$, $Q_{\text{exp}} = Q(s_t, a_t; \theta)$, θ and θ^- are the training and target DNN^Q network parameters, respectively, and y^{Tar} is the TargetQNet output (line 24). In our implementation, we set the minibatch length $M_b = 32$, the local memory size $Ð_{Em} = 15\,000$ and the discount factor $\gamma = 0.9$.

Online DRL-DMC^3 Application: After training, the DQL algorithm obtains a well-trained DMC strategy. During the application phase, the DBS observes the environmental state s_t at each *TS*. Afterward, the DMC strategy chooses an action to reposition the DBS location to maximize $\Theta_{s(t)}$. Algorithm 5 demonstrates the real-time drone-aided communication system to maximize communication coverage while guaranteeing overall throughput of the gMUs to an acceptable level. Moreover, Fig. 11 illustrates the relationship between Algorithms 4 and 5. In this article, we build the DNN^Q with three FC layers, which are set to 300, 250, and 150 neurons. The DNN^Q input is the system state s_t and its output is the estimated Q-values that correspond to all actions at the given s_t. The DBS selects the action with the highest Q-value in accordance with (24) to maximize its reward function (26). The optimal hyperparameters of the proposed DNN^Q algorithm are described in Table V.

VI. Performance Analysis

In this section, the implementation details, including the necessary parameter settings, are presented. Then, the performance of the proposed drone mobility control strategy is evaluated.

A. Implementation Details

Our experiments were carried out on an Ubuntu machine using the Python 3.7 programming language and Keras library with a TensorFlow backend deep learning framework. In the experiments, a random walk model is used to model the user movement. Two different types of user movements are considered, namely, slow and fast speed walking. Additionally, the minimum and maximum allowed DBS altitudes are set between 30 and 50 m, i.e., $30 \leq h_d \leq 50$. The main objective of this work is to develop a drone mobility control strategy for maximizing the communication coverage with an acceptable level of user throughput using CNN and DRL algorithms. The proposed time-series CNN-based link quality estimation model was trained with four convolutional layers, with 32, 64, 128, and 256 neurons and one FC layer. A dropout layer is also added after each pooling to prevent overfitting during the training process, which ensures that the model can be generalized. Adamax is used as the optimizer to adjust the learning rate $L_r = 0.01$ for each epoch and the minibatch size $M_b = 512$. Table IV presents the experimentally validated optimal hyperparameters for training the proposed time-series CNN-based link quality estimation model.

TABLE IV
Network Parameters of the Deep Learning Network

Category	Value
Filter size w	5 * 5
Pooling size	[2, 1]
No. of convolutional layers	4 [32, 64, 128, 256]
Fully connected layer	1
Batch size	512
Optimizer	Adamax
Weight decay	$L2$ regularization ($L2$=0.0005)
Activation function	ReLU, Softmax (last layer)
Optimizer	Adam (L_r = 0.01)
Epoch	100
Loss	cross-entropy

TABLE V
DQL Parameter Settings

Hyperparameters	Value
No. of DNN^Q FC layers	3
No. of neurons in the hidden layer	L_1=300, L_2=250, L_3=150
MCS index threshold (m_{th})	6
Optimizer	Adamax ($\beta = 10^{-3}$)
Activation function	ReLU, Softmax (last layer)
Discount factor γ	0.94
Mini-batch size M_b	32
Exploration coefficient ε	$\varepsilon_i = 0.95$, $\varepsilon_f = 0.01$
Replay memory size $Ð_{Em}$	15000
Number of training episodes e^{max}	1000
Number of trainings per episode T	1000

On the other hand, we trained DNN^Q with three FC hidden layers. The ReLU activation function is adopted for the first two hidden layers of the DNN^Q to map the $\langle s, a \rangle$ pairs to their corresponding Q-values. The softmax nonlinear activation function was used in the last DNN^Q layer to provide the probabilities of all action feedback values. Additionally, the Adamax optimizer is employed to minimize the loss function (25) and to converge the network with a learning rate of $\beta = 10^{-3}$. We trained the DNN^Q over 1000 iterations. Table V reports the optimal training parameters of the DQL algorithm.

The effectiveness of our proposed method is assessed using the following two indicators.

1) System Sum Throughput: The link quality of the gMUs may change over a time horizon due to suboptimal DBS

positions. The MCS index is a good solution for mapping the received user's channel data to the corresponding user throughput in a dynamic wireless environment. The proposed system evaluates the overall throughput of the serving gMUs by mapping the predicted MCS index based on the received radio signals. Accordingly, the average network throughput (ANT) of the DBS wireless communication in each timeslot t can be calculated using the following:

$$\text{ANT} = \sum_{t=1}^{T} \sum_{u=1}^{K} f_{\text{throughput}}(m_{u,d}) \tag{27}$$

where ANT is the average network throughput of all serving ground users and $f_{\text{throughput}}$ is a function that maps the user MCS index to the corresponding data rate at each timeslot t.

2) Communication Coverage Score: At each timeslot, the system coverage score is evaluated using (2).

B. Results and Analysis

1) Link Quality Estimation Model Performance: To verify the effectiveness of the proposed link quality estimation method for the drone mobility control strategy, a total of 1.5 million user channel samples was collected. 80% of the data was randomly selected to train the proposed LQE model, and the remaining 20% was used to evaluate the proposed system. Before being input to the proposed time-series CNN algorithm, the collected data were preprocessed into the appropriate structure using Algorithm 1. The time-series CNN algorithm is then used to estimate the air-to-ground link quality of the gMUs based on the MCS index. The CNN algorithm was trained using various hyperparameters, such as the number of convolutional layers, kernel size, batch size, activation function, optimizer, and the number of epochs to create a well-developed LQE model. The optimal learning hyperparameters of the CNN-based link quality estimation are listed in Table IV. To investigate the effects of the numbers of convolutional and FC layers, we perform four different experiments. These are: 1) LQEM-I (3 Conv and 1 FC); 2) LQEM-II (3 Conv and 2 FC); 3) LQEM-III (4 Conv and 1 FC); and 4) LQEM-IV (4 Conv and 2 FC). According to the learning accuracy, the optimal network structure has been chosen as the proposed well-trained link quality estimation model to estimate each user's link quality over time.

Figs. 12 and 13 illustrate the proposed model (i.e., LQEM-*III*) training and validation cross-entropy loss and accuracy with respect to the number of epochs, respectively

$$\text{CrossEntropy} = -\sum_{i}^{B} \sum_{j=1}^{C_\alpha} LQ_{actij} \log(LQ_{estij}) \tag{28}$$

where B and C_α are the total number of testing samples and MCS indices, respectively, and LQ_{acti} and LQ_{esti} are the actual and softmax probabilities for the jth class in the ith row testing sample, respectively. As shown in Fig. 12, both the training and validation losses tend to decrease as the number of epochs increases. This observation indicates that the proposed CNN network converges quickly and accurately to estimate the MCS index of ground users.

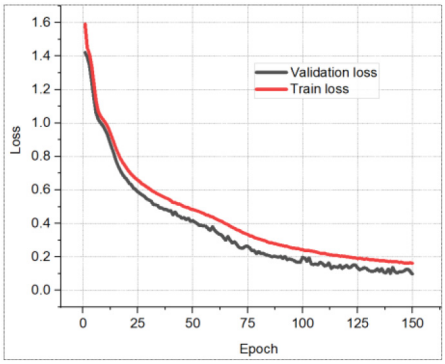

Fig. 12. Performance losses of the time-series CNN LQE model as a function of the number of epochs.

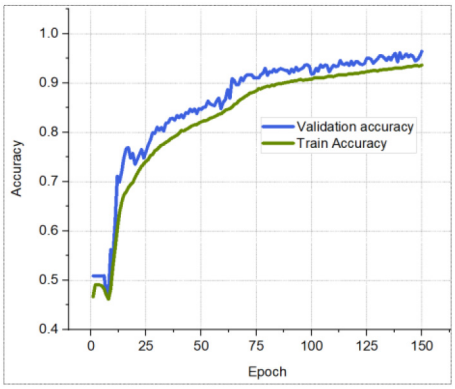

Fig. 13. Time-series CNN-based LQE model accuracy.

TABLE VI
ONLINE-LEARNING-BASED LINK QUALITY ESTIMATION ACCURACY

Model	Accuracy
LQEM-I	92.03%
LQEM-II	93.74%
LQEM-III	**98.69%**
LQEM-IV	98.26%

Table VI shows the link quality estimation performance of deployed drone-aided wireless communication in different network structures. The performance of the proposed link quality estimation method is evaluated using an accuracy metric. The validation accuracies of the time-series CNN-based LQEM-I, LQEM-II, LQEM-III, and LQEM-VI are 92.03%, 93.74%, 98.69%, and 98.26%, respectively. LQEM-I and LQEM-II offer the lower channel estimation accuracy because shallow architectures require more neurons per layer to extract discriminative propagation characteristics from a dynamic environment. However, this requirement will lead to high computational time. Conversely, LQEM-III and LQEM-IV produce better channel estimation accuracy than LQEM-I and LQEM-II. Due to having more convolutional layers, LQEM-III and LQEM-IV extract more significant features of the 3-D air-to-ground wireless channel characteristics in a dynamic and complex environment. However, the proposed LQEM-III classifier outperforms LQEM-IV. For this reason, LQEM-III was selected as the link quality estimation model

TABLE VII
LINK QUALITY ESTIMATION PERFORMANCE COMPARISON BETWEEN THE PROPOSED METHOD AND PREVIOUS WORKS

Learning Algorithm	Accuracy
CNN [6]	91.92
k-means [7]	86.17
RF [16]	87.03%
Time-Series CNN	98.69%

for drone-aided wireless communication to determine the link quality of ground users over time.

In this article, we used real measurement data to build the channel estimation model. In many previous studies, the propagation channels were derived by mathematical channel modeling rather than real measurements. We compared the performance of the proposed system with previous works [6], [7], and [16] to ensure its reliability. However, the environment, the user's channel data, and the assumptions of the model are different. To make the comparison results more representative, we use the proposed user's channel data to compare the proposed link quality estimation model with the previous published methods [6], [7], [16]; the results are presented in Table VII. Luo *et al.* [16] utilized a RF algorithm to estimate the link qualities from the DBSs to the users. RF is a flexible machine learning algorithm that estimates link quality by generating a decision tree. The link quality estimation performance of the proposed LQE, [6], [7], and [16] are 98.69%, 91.92%, 86.17%, and 87.03%, respectively. The method presented in [7] has the lowest results since it has poor scalability for large data sets. The proposed LQE method outperforms the methods presented in [6], [7], and [16] because of the following reasons. First, convolutional layers of the CNN algorithm can effectively capture local dependencies and extract separable features of air-to-ground channel from user channel data. However, the shallow learning methods [7], [16] are more susceptible to gradient disappearance, lack of adaptability, and learning instability, particularly for large data samples. Second, the CNN algorithm uses pooling layers to reduce the dimensionality of a feature map while retaining meaningful feature information that makes the network more robust in a scalable wireless environment. Besides, our proposed system utilized the user's received amplitude and phase signals rather than received signal strength [6], [7], and signal-to-noise ratio [16] to extract discriminative propagation characteristics from a dynamic environment and time-varying channel conditions.

2) Performance of the Drone Mobility Control Strategy: In this section, the performance of the DMC strategy for maximizing communication coverage while maintaining the overall throughput of the ground users to an acceptable level is evaluated according to Algorithm 4. The proposed DRL-DMC^3 performance can be improved by choosing appropriate hyperparameters in a trial-and-error manner [32], as specified in Table V. The proposed DNN$^{\mathcal{Q}}$ has three FC layers. The performance of the DNN$^{\mathcal{Q}}$ algorithm can be improved by assigning an appropriate number of neurons to each layer, enabling the identification of a better $Q(.)$ and policy π. Accordingly, the first, second and third DNN$^{\mathcal{Q}}$ network structures have 300, 250, and 150 neurons, respectively. Fig. 14 depicts the effects of different optimizers, such as the Adam, Adamax, AdaDelta, SGD, and RMSprop optimizers, on the convergence of the proposed DQL-based drone mobility control strategy. The results show that the Adamax optimizer outperforms the others, resulting in faster network convergence and a better reward. However, the SGD optimizer provides the worst accuracy overall, which means that it is not appropriate for our proposed DQL network. The learning rate cannot be too large or too small because a large learning rate causes fluctuations while a small learning rate results in a low training speed. In our experiments, we chose a learning rate of $\beta = 10^{-3}$ to adjust the updated speed of the learnable weights in the DQL algorithm.

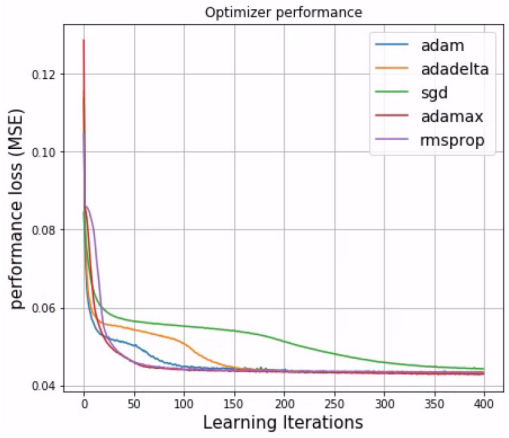

Fig. 14. DQL convergence performances under different optimizers.

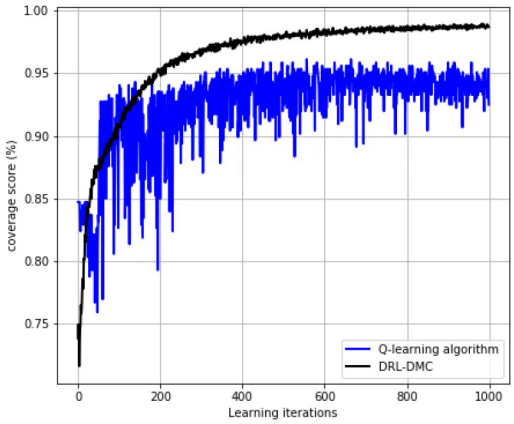

Fig. 15. Communication coverage score.

Fig. 15 shows the coverage score obtained by DRL-DMC^3 and by conventional Q-learning algorithms over time. According to our experimental results, the coverage score increases with increasing learning iterations. Particularly, when the number of timeslots is increased sufficiently, the proposed DMC strategy can achieve a very high coverage score of 97.3%. However, at the beginning of the first iteration, the coverage score of the proposed method can reach only 70.92%. This limit is due to the DBS initially being deployed at the center of the target H_a. Therefore, the DRL agent that the DBS

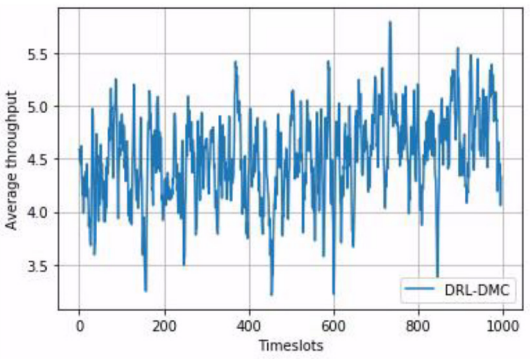

Fig. 16. Average throughput performance in the movement scenario.

cannot move to optimal positions within the limited number of training epochs. Furthermore, the DBS has not yet covered all states at the beginning, and it will take time to reach unexplored states by executing an action on the current state to fully understand the system environment. For comparison, we also show the performance of the Q-learning algorithm employed in [17] and [18]. Compared to DRL-DMC^3, the conventional Q-learning algorithm provides less user coverage. Additionally, estimation of the expected reward using a Q-learning approach has a high variance problem due to learning instability.

Fig. 16 demonstrates the average throughput obtained by the DQL algorithm at different timeslots. The proposed DRL-DMC^3 overall user throughput changes with time in a dynamic wireless channel environment. Moreover, the average throughput increases linearly with an increasing number of timeslots. This change occurs because the proposed method can iteratively learn the system environment from mistakes by executing actions to improve each gMU throughput. The DBS was initially assumed to be deployed at the center of target H_a. Therefore, the overall system throughput is slightly lower at the beginning. Furthermore, due to user mobility and user distribution in the target area, average throughput performance in some timeslots degrades, causing the DBS position to become suboptimal for the tradeoff between the coverage and quality of communication. Consequently, the proposed method repositions the DBS location based on real-time radio signals until each ground user throughput exceeds the threshold value mth. To this end, the integration of an online-learning-based LQE model and a DQL algorithm improves the effectiveness of communication services in a dynamic and complex environment.

VII. Conclusion and Future Work

In this article, we proposed an autonomous 3-D DBS deployment approach to provide better communication services to all ground users in a drone-aided wireless communications system. To address this issue, we proposed a two-step learning method to maximize the communication coverage score while guaranteeing that user throughputs are at acceptable level. First, a time-series CNN-based link quality estimation model was developed to properly determine the 3-D air-to-ground wireless communication links. Second, a DRL-based drone mobility control strategy was developed to control the movement of the DBS based on the received signals to achieve the objective function. We evaluated our DRL-DMC^3 method with real air-to-ground channel measurements. The simulation results showed that the proposed system can achieve promising and reasonable communication performance in terms of both communication coverage and network throughput. As future work, we plan to develop a reconfigurable intelligent surface (IRS)-assisted DBS wireless communications system to significantly improve the channel fading effects to further enhance the overall user throughput and communication coverage.

TABLE VIII
List of Acronyms

Acronym	Description
A2G	Air-to-ground
CNN	Convolutional neural network
CQI	Channel quality indicator
DBS	Drone base station
DMC	Drone base station mobility control
DNN	Deep neural network
DRL	Deep reinforcement learning
FC	Fully connected
LQE	Link quality estimation
OFDM	Orthogonal frequency-division multiplexing
RF	Random forest
RL	Reinforcement learning
gMU	Ground mobile user
MCS	Modulation and coding scheme
QoS	Quality of service
UAV	Unmanned aerial vehicle
USRP	Universal software radio peripheral

Appendix
List of Acronyms

See Table VIII.

References

[1] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.

[2] Q. Wu, L. Liu, and R. Zhang, "Fundamental trade-offs in communication and trajectory design for UAV-enabled wireless network," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 36–44, Feb. 2019.

[3] H. Wang, H. Zhao, W. Wu, J. Xiong, D. Ma, and J. Wei, "Deployment algorithms of flying base stations: 5G and beyond with UAVs," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10009–10027, Dec. 2019.

[4] C. T. Cicek, H. Gultekin, B. Tavli, and H. Yanikomeroglu, "UAV base station location optimization for next generation wireless networks: Overview and future research directions," in *Proc. 1st Int. Conf. Unmanned Veh. Syst. Oman (UVS)*, Feb. 2019, pp. 1–6.

[5] Y. Zhong, T. Q. S. Quek, and X. Ge, "Heterogeneous cellular networks with spatio-temporal traffic: Delay analysis and scheduling," *IEEE J. Sel. Areas Commun*, vol. 35, no. 6, pp. 1373–1386, Jun. 2017.

[6] C. P. Wu *et al.*, "Learning-based downlink user selection algorithm for UAV-BS communication network," in *Proc. IEEE Annu. Consum. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2020, pp. 1–5.

[7] J. Wang, Y. Li, A. Adege, and L. Wang, "Machine learning based rapid 3D channel modeling for UAV communication," in *Proc. IEEE Annu. Consum. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2019, pp. 1–5.
[8] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.
[9] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.
[10] M. Alzenad, A. El-Keyi, and H. Yanikomeroglu, "3-D placement of an unmanned aerial vehicle base station for maximum coverage of users with different QoS requirements," *IEEE Wireless Commun. Lett.*, vol. 7, no.1, pp. 38–41, Feb. 2018.
[11] Y. Chen, N. Li, C. Wang, W. Xie, and J. Xv, "A 3D placement of unmanned aerial vehicle base station based on multi-population genetic algorithm for maximizing users with different QoS requirements," in *Proc. IEEE Int. Conf. Commun. Technol. (ICCT)*, Chongqing, China, Oct. 2018, pp. 967–972.
[12] Y. Y. Munaye, H.-P. Lin, A. B. Adege, and G. B. Tarekegn, "UAV positioning for throughput maximization using deep learning approaches," *Sensors*, vol. 19, no. 12, p. 2775, Jun. 2019.
[13] R. S. Sutton and A. G. Barto *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018, pp. 1–775.
[14] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.
[15] 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 *Proc. IEEE Conf. Global Commun. (GLOBECOM)*, Waikoloa, HI, USA, Oct. 2019, pp. 1–6.
[16] X. Luo, Y. Zhang, Z. He, G. Yang, and Z. Ji, "A two-step environment-learning-based method for optimal UAV deployment," *IEEE Access*, vol. 7, pp. 149328–149340, 2019.
[17] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-UAV networks: Deployment and movement design," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8036–8049, Aug. 2019.
[18] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.
[19] 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 J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.
[20] X. Li, Q. Wang, J. Liu, and W. Zhang, "Trajectory design and generalization for UAV enabled networks: A deep reinforcement learning approach," in *Proc. IEEE Conf. Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, Jun. 2020, pp. 1–6.
[21] W. Shi *et al.*, "Multi-drone 3D trajectory planning and scheduling in drone-assisted radio access networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8145–8158, Aug. 2019.
[22] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen, "Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical DRL approach," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9800–9813, Jun. 2021.
[23] X. Ge, J. Ye, Y. Yang, and Q. Li, "User mobility evaluation for 5G small cell networks based on individual mobility model," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 528–541, Mar. 2016.
[24] J. Fan, Q. Yin, G. Y. Li, B. Peng, and X. Zhu, "MCS selection for throughput improvement in downlink LTE systems," in *Proc. IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, Lahaina, HI, USA, Jul. 2011, pp. 1–5.
[25] I. J. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 2016, pp. 1–775.
[26] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 1–67, 3rd Quart., 2019.
[27] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.
[28] S. K. Pulliyakode, S. Kalyani, and K. Narendran, "Rate prediction and selection in LTE systems using modified source encoding techniques," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 416–429, Jan. 2016.
[29] H.-A. Hou, L.-C. Wang, and H.-P. Lin, "Micro-doppler shift and its estimation in rotary-wing UAV Sub-6 GHz communications," *IEEE Wireless Commun. Lett.*, vol. 10, no. 10, pp. 2185–2189, Oct. 2021.
[30] G. B. Tarekegn, R.-T. Juang, H.-P. Lin, A. B. Adege, and Y. Y. Munaye, "D$_{FOPS}$: Deep learning-based fingerprinting outdoor positioning scheme in hybrid networks," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3717–3729, Mar. 2021.
[31] P. J. M. Ali, and R. H. Faraj, "Data normalization and standardization: A technical report," *Mach. Learn.*, vol. 1, no. 1, pp. 1–6, Jan. 2014.
[32] G. Panchal and M. Panchal, "Review on methods of selecting number of hidden nodes in artificial neural network," *Int. J. Comput. Sci. Mobile Comput.*, vol. 3, no. 11, pp. 455–464, Nov. 2014.
[33] Z. Qin, Z. Liu, G. Han, C. Lin, L. Guo, and L. Xie, "Distributed UAV-BSs trajectory optimization for user-level fair communication service with multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12290–12301, Dec. 2021, doi: 10.1109/TVT.2021.3117792.
[34] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

Getaneh Berie Tarekegn (Student Member, IEEE) received the B.Sc. degree in computer science from the University of Gondar, Gondar, Ethiopia, in 2012, and the M.Sc. degree in electrical engineering and computer science from the National Taipei University of Technology, Taipei, Taiwan, in 2019, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science.

His research interests include the application of deep learning in mobile and wireless networks, machine-learning-based positioning, tracking of IoT devices in a wireless environment, and dynamic drone-base station deployment.

Rong-Terng Juang (Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2002 and 2007, respectively.

Since 2020, he has been an Assistant Professor with the Department of Electronic Engineering, Feng Chia University, Taichung, Taiwan. His research interests include cell planning, mobile localization, and resource management in wireless cellular networks.

Hsin-Piao Lin (Member, IEEE) received the B.S. degree in communication engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1986, and the M.S. and Ph.D. degrees from the Electrical Engineering Research Laboratory, The University of Texas at Austin, Austin, TX, USA, in 1992 and 1997, respectively.

Since 1997, he has been an Assistant Professor with the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, where he is the Founding Professor and has been the Director of the Multiple Antenna System Research and Development Center since 2015. He has authored more than 100 articles and holds six patents. His research interests include cognitive radio, Wi-Fi/LTE network communications, radio propagation modeling, smart antenna systems, and applications of machine/deep learning-assisted wireless communications.

Yirga Yayeh Munaye received the B.Sc. degree in information technology from Bahir Dar University, Bahir Dar, Ethiopia, in 2009, the M.Sc. degree in information science from Addis Ababa University, Addis Ababa, Ethiopia, in 2014, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, National Taipei University of Technology, Taipei, Taiwan, in 2021.

Since 2021, he has been an Assistant Professor with the Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar University. His research interests include the application of deep learning in wireless communications, ad hoc networks, interference management in unmanned aerial vehicle (UAV) communication, and dynamic UAV-base station deployment.

Li-Chun Wang (Fellow, IEEE) received the Ph.D. degree from Georgia Institute of Technology, Atlanta, GA, USA, in 1996.

From 1996 to 2000, he was employed by AT&T Laboratories, Florham Park, NJ, USA, where he was a Senior Technical Staff Member with the Department of Wireless Communications Research. In 2000, he joined the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan, where he is jointly appointed in the Department of Computer Science and Information Engineering. He holds 19 U.S. patents, has authored or coauthored over 200 journal articles and conference papers, and co-edited *Key Technologies for 5G Wireless Systems* (Cambridge Univ. Press, 2017). His current research interests include software-defined mobile networks, heterogeneous networks, and data-driven intelligent wireless communications.

Dr. Wang was a recipient of the Distinguished Research Award of the National Science Council, Taiwan, in 2012 and was a co-recipient of the IEEE Communications Society Asia–Pacific Board Best Award in 2015, the Y. Z. Hsu Scientific Paper Award in 2013, and the IEEE Jack Neubauer Best Paper Award in 1997.

Mekuanint Agegnehu Bitew received the bachelor's degree from Mekelle University, Mekelle, Ethiopia, in 2005, the master's degree from Addis Ababa University, Addis Ababa, Ethiopia, in 2007, and the Ph.D. degree from the College of Electrical and Computer Science, National Taipei University of Technology, Taipei, Taiwan, in 2016.

He has been a Postdoctoral Researcher with the National Taipei University of Technology for two years. Since 2018, he has been an Assistant Professor with the Faculty of Computing, Bahir Dar Institute of Technology, Bahir Dar University, Bahir Dar, Ethiopia. His research interests include WSNs, ad hoc networks, wireless communication, optical networks, and AI.