Learning-Based Energy-Efficient Data Collection by Unmanned Vehicles in Smart Cities

Bo Zhang, Chi Harold Liu, *Senior Member, IEEE*, Jian Tang, Zhiyuan Xu, Jian Ma, and Wendong Wang

***Abstract*—Mobile crowdsourcing (MCS) is now an important source of information for smart cities, especially with the help of unmanned aerial vehicles (UAVs) and driverless cars. They are equipped with different kinds of high-precision sensors, and can be scheduled/controlled completely during data collection, which will make MCS system more robust. However, they are limited to energy constraint, especially for long-term, long-distance sensing tasks, and cities are almost too crowded to set stationary charging station. Towards this end, in this paper we propose to leverage emerging deep reinforcement learning (DRL) techniques for enabling model-free unmanned vehicles control, and present a novel and highly effective control framework, called "DRL-RVC." It utilizes the powerful convolutional neural network for feature extraction of the necessary information (including sample distribution, traffic flow, etc.), then makes decisions under the guidance of the deep Q network. That is, UAVs will cruise in the city without control and collect most required data in the sensing region, while mobile unmanned charging station will reach the charging point in the shortest possible time. Finally, we validate and evaluate the proposed framework via extensive simulations based on a real dataset in Rome. Extensive simulation results well justify the effectiveness and robustness of our approach.**

***Index Terms*—Data crowdsourcing, energy-efficiency, smart city.**

I. Introduction

MOBILE crowdsourcing (MCS) has been widely used in various domains, especially in smart cities, with the help of unmanned vehicles, such as unmanned aerial vehicles (UAVs) and driverless cars. They are equipped with different kinds of high-precision sensors in relatively ample space (if compared with smart devices), that enable them to join an MCS system, in applications of long-/medium-range navigation and landing [1], two-dimensional (2-D) or three-dimensional (3-D) scientific mapping, feature detection for agricultural crops, surveillance for wildlife, emergence rescue, etc.

Fig. 1. Considered MCS scenario for unmanned vehicles, where UAV cruise around the sensing region to collect data with certain charging point to recharge the battery. Mobile unmanned charging station is scheduled to reach the charging point, to prepare to service the UAV.

However, due the limited battery capacity, UAVs are constrained with their sensing range and lifetime, and it is also not viable to increase battery system size as its weight will become another limiting factor [2]. In order to perform a long-term, long-distance task, charging stations are quite necessary and very helpful for UAVs in an MCS system. Nevertheless, cities are usually too crowded to set stationary stations on the road. However, it is possible to use driverless cars, as mobile unmanned charging stations. They are scheduled to the charging point, waiting for UAVs to fly over, charge, and continue sensing tasks. On the other hand, as time progresses, collected sensing data, e.g., environmental parameters, are constantly changing, such as traffic flow conditions, and thus, MCS system operators have to manually control these vehicles during data collection, which will consume much manpower and material resources.

In this paper, our research is motivated by an application scenario as shown in Fig. 1, where some areas of a city are associated with sampling requirements with different priorities. A sensing task is to collect samples under certain constraints, including but are not limited to, time period, data quantity, etc. We consider two kinds of mobile, unmanned agents: UAVs and mobile charging stations. UAVs can fly freely in the region for

Manuscript received September 1, 2017; revised November 11, 2017; accepted November 23, 2017. Date of publication December 14, 2017; date of current version April 3, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61772072 and in part by China Postdoctoral Science Foundation funded project under Grant No.2017M610828. B. Zhang and C. H. Liu contributed equally to this work. Paper no. TII-17-2061.R1. *(Corresponding author: Chi Harold Liu.)*

B. Zhang, J. Ma, and W. Wang are with the Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zbo@bupt.edu.cn; majian@mwsn.com.cn; wdwang@bupt.edu.cn).

C. H. Liu is with the Beijing Institute of Technology, Beijing 100081, China, and the Department of Computer Information and Security, Sejong University, Seoul 143747, South Korea (e-mail: chiliu@bit.edu.cn).

J. Tang and Z. Xu are with the Department of Computer Science and Engineering, Syracuse University, Syracuse, NY 13244 USA (e-mail: jtang02@syu.edu; zxu105@syu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2783439

data collections, while the latter are scheduled to the charging point and prepare to charge UAVs in case they are running out of battery. We also assume that UAVs start at an initial, preset location, fly around to collect data, and finally "return back" to the initial location (we call this as "UAV collection"). Note that unlike human participant in an MCS system, it is important to collect the UAVs for future sensing tasks, rather than leaving them somewhere unknown in the map. Mobile charging stations are scheduled to the charging point in the shortest possible time. Different from UAVs, since mobile charging stations move on the ground, they will be continuously influenced by the traffic flow, and improper scheduling will result in late arrival.

Our goal is that given the sensing region, tasks, and location of charging point, we aim to proposed an efficient algorithm that can provide the best route for both UAVs and mobile charging stations. Recent research output of deep reinforcement learning (DRL [3]) offers a promising technique for enabling effective experience-driven model-free control. Even though DRL has made tremendous successes on game-playing that usually has a limited action space (e.g., moving up/down/left/right), it has not yet been investigated how it can be applied to the area of optimal resource allocation in complex networks, which usually have sophisticated states and huge continuous action spaces. We believe that DRL is promising for control unmanned agents in our application. This is because of the following reasons.

1) It has advantages over other dynamic system control techniques such as model-based predictive control in that the former is model-free and does not rely on accurate and mathematically solvable system models (e.g., queuing models), thereby enhancing its applicability in complex network with random and unpredictable behaviors.
2) It is able to deal with highly dynamic time-variant environments such as time-varying system states and demands.
3) It is capable of handling a sophisticated state space and more advantageous over traditional reinforcement learning (RL).

Towards this end, in this paper, we develop a novel and highly effective DRL-based, model-free framework for unmanned vehicle control. We summarize our contributions as follows.

1) We present a highly effective and practical DRL-based, experience-driven unmanned vehicle control framework, called "DRL-RVC," for long-term, long-distance sensing tasks in an MCS system.
2) We propose a DRL-based loop-back control algorithm for UAVs to cruise within the sensing region to collect the most higher priority data samples.
3) We propose a DRL-based fast-path decision algorithm for mobile unmanned charging stations to be scheduled from origin to the charging point, while fully considering traffic flow conditions to optimize the time or space cost.
4) The effectiveness and flexibility of the proposed strategy have been extensively evaluated by real trace-driven simulations.

The rest of the paper is organized as follows. Section II reviews the related research activities. Section III establishes the scenario definition and formal models of our systems. Section IV introduces necessary background of DRL and the definition of our considered problem. Section V introduces our DRL-based unmanned vehicle control solution. Section VI extensively evaluates the performance of the proposed strategy by real trace-driven simulations, and finally Section VII concludes the paper.

II. Related Work

MCS evolves originally from the concept of participatory sensing, and then followed by many proposals. Zhang *et al.* [4], [5] designed a novel community-centric framework for community activity prediction based on big data analysis, and proposed an approach to extract community activity patterns by analyzing the big data collected from both the physical world and virtual social space. A few participants selection and status prediction methods for energy-aware mobile crowd sensing were proposed in [6]–[9]. Cardone *et al.* [10] initialized a project, called "ParticipAct Living Lab testbed," as an ongoing experiment at the University of Bologna involving 300 students for one year in crowdsourcing campaigns that can passively access smartphone sensors and also require active user collaboration. Pankratius *et al.* [11] discussed an application of crowdsourcing in space weather monitoring, called "the Mahali project." Mahali used GPS signals that penetrate the ionosphere for science rather than positioning. A large number of ground-based sensors will be able to feed data through mobile devices into a cloud-based processing environment, enabling a tomographic analysis of the global ionosphere at unprecedented resolution and coverage. Maharjan *et al.* [12] design an optimal incentive mechanism for multimedia data collection. This is archived by model the interactions between the crowdsourcer and the contributors as a utility maximization problem, and optimize the utility over reward to the participants. Energy efficiency has recently been investigated for MCS. Nath [13] improved device battery lifetimes by inferring sensor readings by sensors with lower energy consumption or temporal continuous readings, since the values of various context attributes can be highly correlated. Furthermore, a software-defined machine-to-machine (SD-M2M) communication framework, which enables fine-grained resource allocation and reliable QoS guarantees via data-control separation and service-infrastructure decoupling, was proposed for the domain of smart energy management in [14]–[17]. It provides unprecedented flexibility, scalability, and programmability for managing a large number of heterogeneous M2M devices, and can effectively enhance the data acquisition and exchange capability in a variety of smart energy management applications.

DRL has recently attracted extensive attention from both industry and academia. The goal of RL is to learn good policies for sequential decision problems, by optimizing a cumulative future reward signal. Q-learning [18] is one of the most popular RL algorithms, but known to sometimes learn unrealistically high action values because it includes a maximization step over estimated action values, which tends to prefer overestimated to underestimated values.

A pioneering work [3] proposed DQN, that can learn successful policies directly from high-dimensional sensory inputs. Particularly, they introduced two new techniques, experience replay and target network, to improve learning stability. Van Hasselt *et al.* [19] proposed double Q learning as a specific adaptation

to the DQN. Schaul *et al.* [20] proposed to use prioritized experience replay in DQN, so as to replay important transitions more frequently, and therefore learn more efficiently.

Due to the powerful feature construction capability, convolutional neural network (CNN) have achieved the state-of-the-art performance on a wide variety of computer vision tasks. Many CNN-based methods have been introduced for image classification, object detection, image content analysis, etc. For example, Huang *et al.* [21] design a dense convolutional network architecture for image classification, which introduces direct connections between any two layers with the same feature-map size. By integrating the properties of identity mappings, deep supervision, and diversified depth, the feature can be reused throughout the networks. Recently, region-based convolutional neural networks (R-CNNs) are widely used for object detection which combines selective search region proposals and convolutional network based postclassification.

CNN-based algorithms are also used to perceive environment information for visual navigation, such as [22] trains a deep neural network (DNN) for visually perceiving the direction of an hiking trail from a single image. Mnih *et al.* [23] present a deep learning model based on CNN and Q-learning, which can obtain much better scores when playing Atari games, by using only raw pixels as the model input. Oh *et al.* [24] introduce three classes of cognition-inspired tasks in Minecraft, the models are trained and evaluated on disjoint sets of maps and can be used in unseen (interpolation and extrapolation) maps.

Based on the above-mentioned analysis, how to apply novel deep learning models and algorithms in MCS to control UAV and mobile unmanned charging station together for city-level data collection has been sparsely investigated and serves as the focus of our paper.

III. System Model

This section presents our considered scenario and formal model for our MCS system. We consider our scenario in a 2D region $\mathcal{L}$, as shown in Fig. 1. Our system is composed of an origin, a mobile unmanned charging station o, a UAV m and a charging point, and a set of sensing tasks. Sensing region $\mathcal{L}$ is divided into a set of l subregions, denoted by $\mathcal{L} \triangleq \{l = 1, 2, \ldots, L\}$, where each of them is associated with certain data sample priority denoted by p. Each task lasts for t time slots/decision epochs. At very beginning, a certain task (which is associated with the data distribution and priority requirements in the sensing region $\mathcal{L}$) is published. Required data samples in different subregions have different priorities, as shown in Fig. 1. During the data collection phase, UAV m should collect higher priority data samples as many as possible with limited energy.

For UAV m, we assume that it can fly to an adjacent subregion in one time slot from the current subregion, due to its limited speed. It is also initialed with certain energy constraint at the start point, denoted by e_m. Without loss of generality, here we assume that e_m is translated into the number of subregions, or the number of time slots, a UAV can fly, given fixed per subregion energy consumption.

For the mobile unmanned charging station o, let traffic flow condition of a particular subregion is denoted by f, and each subregion l is certainly associated with different traffic flows. Station o also has limited energy, denoted by e_o. In our scenario, we explicitly consider the traffic congestion impact when scheduling the mobile station from origin to the preset charging point, that different energy consumption will be used if the station o passes certain subregion.

To facilitate the learning based control, we consider the system state space, denoted by $\mathcal{S} = \{s = 1, 2, 3, \ldots\}$, includes the data sample priority of each subregion (denoted by $\mathcal{P} = \{p = 1, 2, 3, \ldots, P\}$), the information of each subregion that whether or not it was visited before, the exact location of charging point the mobile charging station should be scheduled to (denoted by c) and the number of steps that UAV and mobile station have moved. When the states are determined before a time slot, our system will make a decision, the action to perform (i.e., the direction in which UAV and mobile station needs to move), denoted by $\mathcal{A} = \{a = 1, 2, 3, \ldots, A\}$. After the action is taken, they will receive a reward (either positive or negative), denoted by $\mathcal{R} = \{r = 1, 2, 3, \ldots, R\}$ and the system status will be updated accordingly.

Table I shows the list of important notations used in this paper.

TABLE I
List of Important Notations and Their Descriptions

Notation	Explanation
m	Data collector (UAV)
o	Mobile unmanned charging station
e	Energy constraint of UAV and mobile charging stations
$\mathcal{L}$	Entire sensing region
$\mathcal{T}$	Entire time slots for a sensing task
z	Current position of UAV and mobile charging station
$\mathcal{S}$	System state space
$\mathcal{A}$	A set of actions to take
$\mathcal{R}$	A set of reward after the action
$\mathcal{P}$	Priority of subregions in a sensing region
g	Location of origin where agents start
c	Present location of charging point
f	Current traffic flow condition

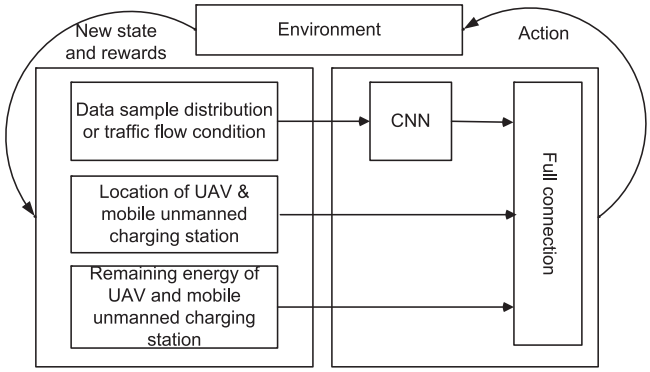

Fig. 2. Overall system flow of our proposed solution.

A. Overall System Flow

During the training process, as shown in Fig. 2, our proposed approach runs iteratively at each decision epoch. Before

each decision epoch, we use CNN to extract the feature of current map (by using the data sample distribution or traffic flow condition as the input). Then, the extracted feature, location of UAV or mobile unmanned charging station, and the steps they have flied/moved are used as the inputs of full connection layer for action decision. Once the action is produced, we obtain the next system state and the reward for the next round of iteration.

B. Feature Extractions by CNN

Due to the powerful feature construction capability, as a class of attractive deep models for automated feature construction, CNNs have achieved the state-of-the-art performance on a wide variety of computer vision tasks. Many CNN-based methods have been introduced for image classification [25], object detection [26], image content analysis [27], etc. These CNNs can automatically generate massive features from the input images by convolution, reduce the amount of parameters and control overfitting by pooling operation.

In MCS, we usually use n-step Markov decision process (MDP) based algorithms to predict participants' trajectories, and better results are expected if considering more steps. In this paper, we use a DRL-based algorithm to predict the best action for UAVs to collect data, which can also be considering a kind of trajectory prediction method. The most important input for our model is a 2-D map (that has been discretized as a digital image), the state of current sensing region, and the model will obtain the region relevance directly. However, CNN has shown excellent performance in many computer vision and machine learning problems, especially in image related tasks. When CNN is used for calculating scores of different actions, the status of nearby locations are considered, then our model can predict a better action, and training process is accelerated.

Thus, we consider to utilize CNNs for unmanned vehicle action recognition while collecting data in MCS. A simple approach in this direction is to treat traffic flow and the distribution of required samples as static images, and apply CNNs to recognize actions at the individual time slot level. However, such approach does not consider the motion information encoded in multiple contiguous time slots. Since the division of sensing region and the traffic flow or priority of data samples in each subregion are all very important in our approach, it is not applicable to use max pooling method that will cost the loss of necessary information. An additional advantage of the CNN-based models is that the recognition phase is very efficient due to their feed-forward nature. Thus, we show that by applying CNNs in our DRL-based framework, the network can accelerate the convergence of the proposed algorithm. In our scenario, CNNs are used to extract the correlation information of the adjacent subregions, which can increase the convergence rate during model training.

IV. Background

In this section, we introduce necessary background of DRL and the definition of our considered problem.

A. DRL Background

RL [28] is not deep learning, but we can consider it as a kind of machine learning techniques. RL is usually used for sequential decision making, deals with learning a policy for an agent that interacts an unknown environment. For one model training step, after observing the current environment state, the agent takes an action given by the current policy, observes the reward and the new environment state. The agent saves these data and use them to optimize the policy. After sufficient amount of training steps, the agent will find a policy to solve the problem. RL-based algorithms are more and more popular and many researchers have found that RL is good at "playing games," such as Go game. An RL-based algorithm needs to answer two basic questions, how to evaluate a policy and how to find an optimal strategy for the problem.

In MCS, key challenges to be tackled, such as how to select the best participants for data contributions, how to protect the privacy of participants during data collection, and how to distribute the incentive for encourage participants, are all policy-based problems, that can potentially fully leverage the strengths of RL-based algorithms.

Compared with other algorithms, such as MDP, the benefits of RL-based approaches are that the trained model can solve extremely complex problems and give a solution within a very short time, but the training process is time consuming and computationally inefficient.

To learn a policy to control a system: when the states $s \in S$ is given in environment X, make a decision, provide the actions $a \in A$ for the system, by maximizing the expected sum of returns according to a reward function $r(s, a)$. At each decision epoch t, the agent observes state s_t, takes an action a_t, and receives a reward r_t. The objective is to maximize the expected sum of returns and find a policy $\pi(s)$ mapping a state to an action (deterministic) or a probability distribution over actions (stochastic) with the objective of the discounted cumulative reward $R_0 = \sum_{t=0}^{T} \gamma^t(s_t, a_t)$, where $r(\cdot)$ is the reward function and $\gamma \in [0, 1]$ is the discount factor. The expected return R can be optimized using a variety of model-free and model-based algorithms.

1) Model-Free and Model-Based RL: When modeling the environment, there are two type of methods for standard RL.

a) Model-based RL [29]: It is to learn a real world model by trial-and-error interactions with the real world and then producing optimal actions and values on the learned model. At the beginning of a task, the dynamic distribution $\Pr(s_t + 1|s_t, a_t)$ is known, or the distribution can be approximated with certain learned model $\Pr(s_t + 1|s_t, a_t)$. The agents, i.e., UAV and mobile unmanned charging station in our case, either know all relevant probabilistic information about state transitions and rewards, and know how this information depends on past events, or estimates this information through observations to determine actions and values, by a suitable version of classical dynamic programming or a similar optimization technique. The advantage of a model-based algorithm is that less exploration is needed, so that it can immediately propagate that information throughout the state space, whenever some information is gained from an exploratory move.

b) Model-free RL: It is to learn optimal actions and values by trial-and-error interactions with the real world without building an explicit model. At the beginning of a task, the dynamic distribution $\Pr(s_t+1|s_t, a_t)$ is not known, and the agents attempt to construct policy and evaluation functions directly, without the ability to predict transitions or immediate rewards resulting from its performed actions with no explicit memory, other than the memory of the current and the next state; for example, the value function or Q-function learning with function approximation [30] in physical control cases. Policy gradient methods provide a simple, direct approach to RL, which can succeed on high-dimensional problems, but potentially requires a large number of samples [31]. Off-policy algorithms that use value or Q-function approximations can achieve better data efficiency in principle [32].

2) DRL: DRL [3] was proposed in learning to play Atari games, archived by Q-learning and the end-to-end system control that are based on high-dimensional sensory inputs. During the training procedure, they utilize a DNN called DQN to derive the correlation between each state–action pair (s_t, a_t) of the system under control and its value function $Q(s_t, a_t)$. Thus, the action-value function that describes the expected return after taking an action a_t in state s_t and thereafter following policy π is defined as follows:

$$Q(s_t, a_t) = \mathbb{E}\big[R_t|s_t, a_t\big] \tag{1}$$

where $R_t = \sum_{k=t}^{T} \gamma^k r(s_t, a_t)$. A commonly used off-policy algorithm takes the greedy policy $\pi(s_t) = \arg\max_{a_t} Q(s_t, a_t)$. Furthermore, DQN can be trained by minimizing the loss function:

$$L(\theta^Q) = \mathbb{E}\big[y_t - Q(s_t, a_t)|\theta^Q\big] \tag{2}$$

where θ^Q is the weight vector of the DQN and y_t is the target value, which can be estimated by

$$y_t = r(s_t, a_t) + \gamma Q\Big(s_{t+1}, \pi(s_{t+1}|\theta^\pi)\Big|\theta^Q\Big). \tag{3}$$

Although using a DNN to serve as the function approximator in RL has been proposed before, a nonlinear function approximator (such as neural network) is known to be unstable or even to diverge. In this regard, two effective techniques were introduced [3] to improve system stability, i.e., experience relay and target network. Unlike traditional RL, a DRL agent updates the DNN with a mini-batch from an experience replay buffer that stores state transition samples collected during learning. Compared with using only immediately collected samples (such as original Q-learning), random sampling from the experience reply buffer allows the DRL agent to break the correlation between sequentially generated samples, and learn from a more independently and identically distributed past experiences, which is required by most of training algorithms, such as stochastic gradient descent. Therefore, experience replay can smooth out learning and avoid oscillations or divergence. In addition, a DRL agent utilizes a separate target network (with the same structure as the DQN) to estimate target values for training the DQN, whose parameters, however, are slowly updated with the DQN weights during training and are held fixed between individual updates.

B. Problem Definition

We describe the unmanned vehicle control problem in this section. As time progresses, since the sensing requirements are always changing in continuous tasks, in order to let mobile agents (i.e., UAV and mobile unmanned charging station in our case) learn to find the best route automatically, it is necessary to force them to know much about the environmental states. Here, we try to make the problem statement as general as possible. Consider a sensing region (a map) is just $n \times n$ dimensions, and one charging station is set. Meanwhile, the origin of UAV and station is always placed at the bottom left of the map and the charging point is somewhere random on the map.

Our problem is likely to use the standard RL model that is able to solve the shortest path problem when provided with sufficient state space variables. For instance, by using classical Q-tables, it determines a future optimal score by analyzing an MDP. But the underlying traffic condition is always changing, then classical Q-tables are completely useless. Furthermore, if the sensing region is much bigger, or we divide the map into more subregions, the required storage space for classical Q-tables will be too large to handle efficiently. Therefore, it is necessary to use DRL that trains a network instead of using a look-up table to solve our problem.

1) Cruising Route Learning for UAVs With Charging Point: Since UAVs are flying in the sky during data collections, they are not subjected to the ground traffic, but limited by their power supply. Thus, we hope to fly the UAV to always take the action that will collect higher priority data samples as many as possible, within this limited battery. To solve this problem, on the one hand, we use the traffic flow information to generate the environmental maps for our problem, and we assume the value of the traffic flow as the priority of data samples. On the other hand, we also random generate some experimental maps as supplements.

2) Traffic-Aware Fast Route Learning for Mobile Unmanned Charging Station: Different from UAVs, mobile unmanned charging stations, such as driverless cars, are strongly influenced by the ground traffic flow, and also limited by their power supply. Therefore, our goal is to schedule the agent to always take the action that will bypass congested subregions and reach the preset charging point as soon as possible.

V. Proposed DRL-Based Control Solution

In this section, we present the proposed solution for DRL-based unmanned vehicle control, to solve our problem described above. In order to utilize the DRL techniques (no matter which method/model to use), we first need to define the state space, action space, and reward function.

1) State space: It consists of four components, information of each subregion that if it is visited before, the exact location of UAV and mobile charging station, and the steps they have flied/moved. Formally, the state vector $\mathbf{s} = \{(p_k, v_k, t_k, c_k)|k = 1, 2, \ldots, K\}$.

2) Action space: An action is defined as the solution to the unmanned vehicle control problem, i.e., the direction in which the agents need to fly/move. Formally, the action vector $\mathbf{a} = \{a_1, a_2, \ldots, a_k, \ldots, a_K\}$.
3) Reward: It is the objective of the unmanned vehicle control problem, which is calculated in the awareness of the priority of each subregion. We will provide punishment to UAVs, if not returning to the origin or reaching the mobile recharging station before running out of battery. Formally, $\mathbf{r} = \{r_1, r_2, \ldots, r_k, \ldots, r_K\}$.

Note that the design of state space, action space, and reward are critical to the success of a DRL method. Our design well captures data collection states and the key components of the unmanned vehicle control problem without including useless/redundant information. The core of the proposed control framework is that an agent runs a DRL algorithm (see Algorithms 1 and 2) to find the best action at each decision epoch, takes the action to observes the next state, and collect a transitional sample.

The unmanned vehicle control problem is obviously a continuous control problem. As described previously, the basic DQN-based DRL in [3] does not work well. We suspect that this is due to the following two reasons: 1) the DRL-based framework in [32] does not clearly specify how to explore. A simple random noise based method or the exploration methods proposed for physical control problems (as mentioned in [32]) do not work well for the unmanned vehicle control problem here; and 2) DRL-based algorithm utilizes a simple uniform sampling method for experience replay, that ignores the significance of transition samples in the replay buffer.

To address these two issues, we propose two new techniques to optimize DRL-based algorithm particularly for unmanned vehicle controls, including exploration that leverages an optimal unmanned vehicle control solution as the baseline during exploration; and DRL-based prioritized experience replay that can employs a new method for specifying significance of samples with careful consideration in networks. Exploration is an essential and important process for training a DRL agent, because an inexperienced agent needs to see sufficient transitional samples to gain experience and eventually learn a good (hopefully optimal) policy. DRL-based algorithm generates an action for exploration by adding a random noise to the action returned by the current network.

For exploration, we propose a new randomized algorithm that guides the exploration process with a base unmanned vehicle control solution. Specifically, with ϵ probability, the DRL agent derives an random action, and with $(1-\epsilon)$ probability, it derives an action as $a + \epsilon \cdot N$, where a is the output of actor network $\pi(\cdot)$ and ϵ is an adjustable parameter. ϵ can tradeoff exploration and exploitation, by determining the probability of adding a random noise to the action, rather than taking the derived action from the actor network. ϵ decays with decision epoch t, which means with more learning, more derived (rather than random) actions will be taken. Parameter N is a uniformly distributed random noise. The proposed control framework is not restricted to any specific base unmanned vehicle control solution for abase, which can be obtained in many different ways.

Algorithm 1: Proposed solution for cruising route learning for UAVs with charging point.

while True **do**
  Get the current state s, the remaining energy level e_m, the current position z_m;
  Generate a random number n;
  if $n < \epsilon$ **then**
    Generate a random action a;
  else
    Obtain action $a + \epsilon \cdot N$ from network;
  end if
  Reduce the value of ϵ
  Get the following state s', the remaining energy level e'_m, the next position z'_m, obtained reward r, if we execute action a:
  if action a makes UAVs out of bounds **then**
    Let UAVs stay in situ
    Reduce the value of r with a smaller value
  end if
  if e'_m is lower than the threshold for the first time **then**
    if z'_m is not c **then**
      Reduce r with a bigger value, reset parameters, update $\mathcal{P}$ with a new map
    end if
  end if
  if e'_m is lower than the threshold for the second time **then**
    if z'_m is not g **then**
      Reduce r with a bigger value, reset parameters, update $\mathcal{P}$ with a new map
    end if
  end if
  if z'_m has not been visited before **then**
    Increase the value of r with the priority of z'_m
  end if
  Obtain the maximum Q-value v using s', e'_m and z'_m
  Save s, e_m, z_m, s', e'_m, z'_m, r, and v to the replay buffer
  if the length of replay buffer is larger than the maximum size **then**
    Drop the first record
  end if
  if the length of replay buffer is larger than the batch size W **then**
    Randomly extract W records
    if e'_m is larger than the threshold in a record **then**
      Increase the value of r with $\gamma \cdot v$
    end if
    Train network
  end if
  Execute action a
  if z'_m has not been visited before **then**
    set the priority of z'_m to 0
  end if
  if e'_m is lower than the threshold **then**
    Reset parameters, update $\mathcal{P}$ with a new map
  end if
  Save network checkpoint every 1000 steps
end while

Algorithm 2: Proposed solution for traffic-aware fast route learning for mobile unmanned charging station.

while True **do**
  Get the current state s, the remaining energy level e_m and the next position z_m
  Generate a random number n
  if $n < \epsilon$ **then**
    Generate a random action a
  else
    Obtain action $a + \epsilon \cdot N$ from network
  end if
  Reduce the value of ϵ
  Get the next state s', the remaining energy level e'_m and the next position z'_m, obtained reward r, if we execute action a:
  if action a makes m out of bounds **then**
    Let m stay in situ
    Reduce the value of r with a smaller value
  end if
  if e'_m is lower than the threshold **then**
    if z'_m is not c **then**
      Reduce r with a bigger value, reset parameters, update f with a new map
    end if
  end if
  Reduce the value of r with the traffic flow value of z'_m
  Obtain the maximum Q-value v using s', e'_m and z'_m
  Save s, e_m, z_m, s', e'_m, z'_m, r, and v to the replay buffer
  if the length of replay buffer is larger than the maximum size **then**
    Drop the first record
  end if
  if the length of replay buffer is larger than the batch size W **then**
    Randomly extract W records
    if e'_m is larger than the threshold in a record **then**
      Increase the value of r with $\gamma \cdot v$
    end if
    Train network
  end if
  Execute the above action
  if e'_m is lower than the threshold **then**
    Reset parameters, update f with a new map
  end if
  Save network checkpoint every 1000 steps
end while

A. Solution for UAV Cruising Route Learning With Charging Point

Here, we formally present the proposed solution for UAV cruising route learning with charging point, called "DRL-RVC," as shown in Algorithm 1. At the beginning, we randomly initializes all the weights of DQN and the UAV is replaced in the lower-left corner of the map (the origin). Before each decision epoch, we generate a random number n, and if its value is smaller than ϵ, we make a random action for UAV, or use the DQN output, calculated with the current system state s (remaining sample distribution), the remaining energy level e_m, and the current position of UAV z_m. Then, we observe the next system state s', the remaining energy level e'_m, the next position of UAV z'_m, obtained reward r, if we execute the above action a, as follows.

1) If action a leads the UAV out of boundary, we will let UAV stay within the region and reduce the r value with a penalty.
2) If the remaining energy level e'_m is lower than the threshold for the first time and UAV does not reach the charging point c, we will restart this trial and reduce the value of r with a penalty.
3) If the remaining energy level e'_m is lower than the threshold for the second time and UAV does not return back to the origin g, we will restart this trial and reduce the r value with a higher penalty.
4) Otherwise, if the next position z'_m is not visited before, we will increase the r value with the priority of z'_m.

Then, we will save the transition (s, e_m, z_m, a, s', e'_m, z'_m, r, DQN output, whether or not trial is over) in a mini-batch. If we have accumulated enough mini-batches, we will randomly extract certain transitions from the saved mini-batches to train the network. At the end of this trail, we will execute the above-mentioned action for the UAV, update and save all the necessary parameters.

B. Solution for Traffic-Aware Fast Route Learning for Mobile Unmanned Charging Station

Next, we formally present the proposed solution for traffic-aware fast route learning for mobile unmanned charging station, as shown in Algorithm 2. This algorithm is nearly identical with Algorithm 1 with slight differences summarized as follows: 1) the system state is calculated corresponding to the traffic flow condition, and 2) mobile unmanned charging station does not need to return back to the origin g.

VI. Performance Evaluation

In this section, we first describe our experiment settings including the used data set, and then we present results and discussions. Our experiments are performed with Tensorflow 1.3 [33] and Python 3.5 in a Ubuntu 16.04.3 server with two NVIDIA TITAN Xp graphics cards.

A. Experiment Settings

We evaluate the proposed scheme by simulations using the real dataset of mobility traces of taxis in Rome, Italy [34]. It contains GPS coordinates of approximately 320 taxis collected over 30 days. Each trajectory is marked by a sequence of time-stamped GPS points that contain taxi driver ID, time stamp (i.e., date and time), and taxi position (i.e., latitude and longitude). We adopt the following procedures to set up our simulation.

(a) (b) (c)

Fig. 3. Used dataset of mobility traces of taxi cabs in Rome, Italy. (a) All taxi trajectories in Rome. (b) Map and trajectories of taxies in one of the selected subregions. (c) Traffic flow condition of subregion as shown in (b) and also the movement trajectory obtained from the model.

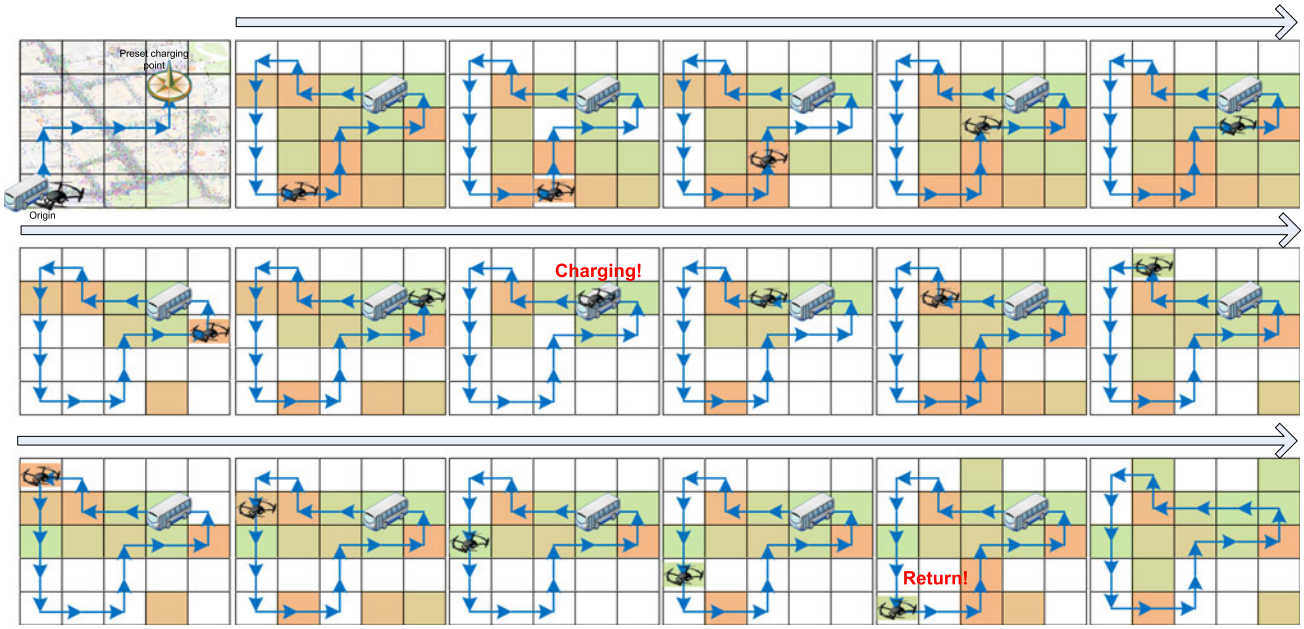

Fig. 4. Movement trajectory of UAV from origin, flying around the sensing region to collect data, and then before battery is exhausted, it moves to the charging point to recharge the battery by a mobile unmanned charging station, then continues to collect data and returns back to the origin.

1) As all traces were spread in different parts of Rome, more than 20 million GPS are observed while very unevenly distributed. We find some regions inside Rome with the relative size of 500 m $\times$ 500 m. Fig. 3 shows the total GPS points (as taxi trajectories), one of selected subregion and its traffic flow condition.
2) In order to simplify the computation in our experiments, the entire region was divided into 5 $\times$ 5 slots of 100 m $\times$ 100 m, i.e., $L = 25$. Due to the number of selected maps still too less for model training, we use random generated maps for model training and the algorithm evaluations are proceed with 10 000 exist generated maps and some real maps. We assume the GPS points in each subregion as its traffic flow condition.
3) We set the origin (where UAV and mobile unmanned charging station starts) in the lower-left corner of the map, and the charging point (where mobile unmanned charging station should be scheduled to) in almost the higher right corner of the map. The initial remaining energy is set only enough for the UAV to cross only eight subregions (i.e., recharging battery is compulsory).
4) The model training phase lasted at least 300 000 iterations, and the experiments for other algorithms are repeated at least 1 million times, and we take the average values as the final results shown in the figure.

B. Results and Discussions

The obtained movement trajectories for both UAV and mobile charging station are shown in Fig. 4. First, we examine the training loss for our proposed DRL-based unmanned vehicle control model, as shown in Fig. 5(a). It is obvious that there are two rapid drops for the training loss during model training phase before 300 000 iterations. After the second decrease, the value of training loss becomes quite stable, always lower than 1, although it continues to oscillate slightly until the end

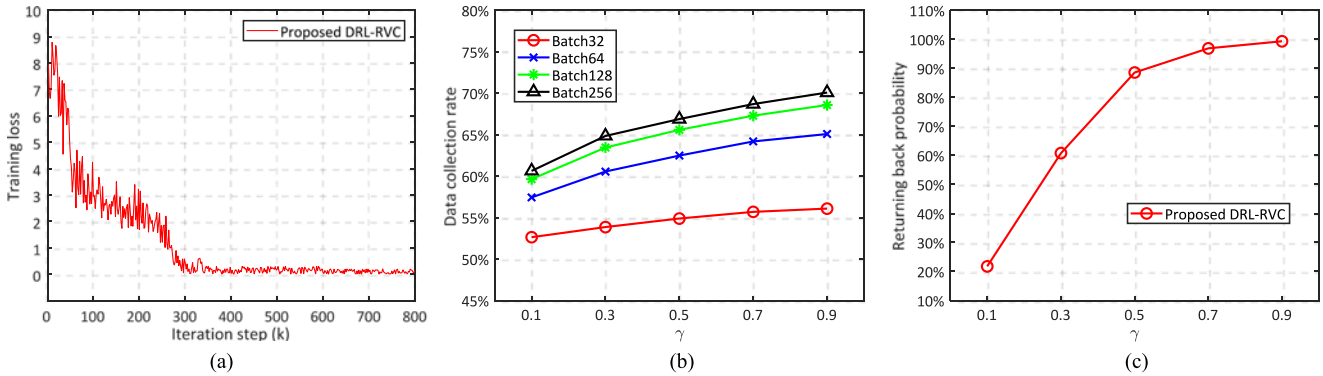

Fig. 5. Simulation results of (a) the training loss over time, (b) the impact of discount factor γ and replay buffer batch size on data collection ratio, and (c) the impact of discount factor γ on the returning probability of unmanned vehicles, all for our proposed DRL-RVC model.

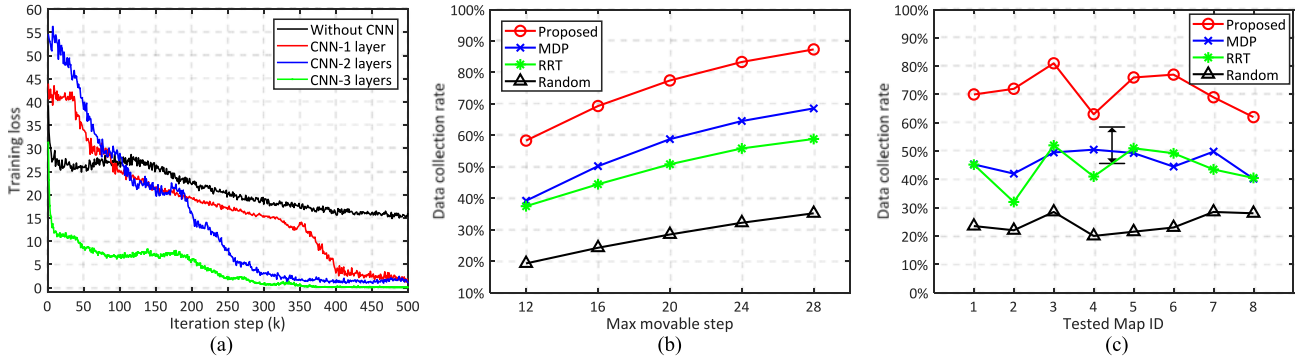

Fig. 6. Simulation results of (a) the impact of the number of CNN hidden layers on the training loss convergence over time, (b) data collection ratio over maximum moveable steps for four approaches, and (c) data collection ratio on different maps for four approaches.

of the training. This shows that the proposed deep model can successfully converge after training and make the optimal decision given any system state. It also means that with the trained model, UAV can collect most higher priority data samples, reach to the charging point during data collection, and finally return to the origin.

Then, we examine the impact of γ on data collection ratio, the decay rate of future observations for our proposed DRL-based unmanned vehicle control model after 300 000 iterations, as shown in Fig. 5(b). It is obvious that with the growth of discount factor γ, the system is able to collect more samples, such as when $\gamma = 0.9$, data collection ratio is 15.57% higher than that of $\gamma = 0.1$. Meanwhile, different batch sizes of the replay buffer used for model training has positive impact on the data collection rate as well. This is because insufficient amount of historical samples will not help the model learn the best action. However, this improvement becomes marginal when the batch size is over 256 since enough samples are used for training, and higher batch size will not generate distinct samples.

Given that discount factor γ is useful to improve the data collection ratio, we next examine its impact on the probability of the unmanned vehicles returning back to the origin. As shown in Fig. 5(c), we can observe that when γ is low, our proposed model cannot be trained successfully, and it is almost impossible for the unmanned vehicles to return back, but when γ is higher than 0.7, we can obtain a reliable deep model after 300 000 iterations that guarantees satisfactory returning back probability.

Then, we examine the impact of the number of CNN hidden layers, as shown in Fig. 6(a). It is obvious that CNN can speed up the model training process, i.e., without CNN, it takes more than 500 000 iterations to obtain a satisfactory training loss. With the increase of the number of CNN hidden layers, e.g., with three hidden layers, the model training is much faster, and will be completed right after 300 000 iterations.

Next, we compare our proposed algorithm, referred as "DRL-RVC," with the Markov decision process (MDP) based selection method, referred as "MDP," and the rapidly exploring random tree (RRT) based selection method, referred as "RRT." They keep selecting actions considering the future rewards in a greedy way. Besides, we also compare with the random selection method, referred as "Random," that randomly selects actions in each decision epoch.

As shown in Fig. 6(b), it is clear that our proposed algorithm can obtain the highest data collection rate for almost all the tested maps, and its obtain value is higher than other three methods. Especially, this value for map 3 is 55.87% higher than MDP-1, 63.42% higher than MDP-2, and 192% higher than the random selection method. This indicates that the selected actions in each decision epoch are almost optimal compared

TABLE II
PROBABILITIES OF UAV TO REACH THE CHARGING POINT AND RETURN BACK TO THE ORIGIN DURING 10 000 TESTS

Solution	Reaching charging point	Returning back
DRL-RVC	99.83%	99.61%
MDP	15.57%	0.18%
RRT	19.28%	1.55%
Random	2.92%	0.08%

with other methods. It is because that "reaching the charging station" and "returning back to the origin" are not the necessary parameters for the MDP-based approach nor for the random selection method, so that UAV may exhaust its energy during data collection while never returning back to the origin, or continuing to collect data afterward. Then we test our proposed algorithm with the real dataset, as shown in Fig. 6(c), it is clear that our proposed algorithm can obtain the highest data collection rate as the simulations.

After, we compare the probabilities for the UAV to reach the charging point and return back to the origin as two important considerations. This is because that UAVs cannot release freely without collecting back for future usages. As shown in Table II, it is clear that with our proposed algorithm, the data collector (i.e., UAV) can almost reach the charging point to recharge the battery before the next series of flies during data collection, and successfully return to the origin at the end after about 300 000 iterations. Compared with "MDP," the collector only reaches the charging point 1, 557 times before the battery is exhausted and return to the origin only 18 times in 10 000 tests. In "RRT," the collector reaches the charging station 1,928 times and only return back 155 times. In the random selection method, the collector reaches the charging point 292 times and finally return to the origin only eight times. This is because that neither of them explicitly consider returning back as the constraint, and they always try to maximize the collected data in a greedy way, i.e., we cannot control their final location for MDP, RRT, or random selection methods.

VII. CONCLUSION

In this paper, the problem of energy-efficient unmanned vehicle control is investigated. Specifically, we present a highly effective and practical DRL-based experience-driven unmanned vehicle control framework, DRL-RVC. In order to collect most higher priority data samples in the sensing region, we propose a DRL-based algorithm for UAV cruise route design; and in order to ensure that UAVs can have enough energy to go back to the origin, we also propose a DRL-based algorithm for mobile unmanned charging station movement route design considering traffic flow, which can ensure the mobile unmanned charging station can reach the destination as soon as possible. Finally, extensive simulation results, based on a real trace dataset of taxis in Rome, showed the effectiveness and robustness of our approach.

REFERENCES

[1] R. Sabatini, M. Richardson, C. Bartel, T. Shaid, and S. Ramasamy, "A low-cost vision based navigation system for small size unmanned aerial vehicle applications," *J. Aeronautics Aerosp. Eng.*, vol. 2, no. 2, pp. 1–16, 2013.

[2] M. Simic, C. Bil, and V. Vojisavljevic, "Investigation in wireless power transmission for UAV charging," *Procedia Comput. Sci.*, vol. 60, pp. 1846–1855, 2015.

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

[4] Y. Zhang, M. Chen, S. Mao, L. Hu, and V. C. Leung, "Cap: Community activity prediction based on big data analysis," *IEEE Netw.*, vol. 28, no. 4, pp. 52–57, Jul./Aug. 2014.

[5] Y. Zhang, M. Chen, N. Guizani, D. Wu, and V. C. M. Leung, "Sovcan: Safety-oriented vehicular controller area network," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 94–99, Aug. 2017.

[6] O. Yurur, C. H. Liu, and W. Moreno, "Unsupervised posture detection by smartphone accelerometer," *Electron. Lett.*, vol. 49, no. 8, pp. 562–564, 2013.

[7] C. H. Liu, J. Fan, P. Hui, J. Wu, and K. K. Leung, "Towards QoI and energy-efficiency in participatory crowdsourcing," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4684–4700, Oct. 2015.

[8] C. H. Liu, B. Zhang, X. Su, J. Ma, W. Wang, and K. K. Leung, "Energy-aware participant selection for smartphone-enabled mobile crowd sensing," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1435–1446, Sep. 2017.

[9] C. H. Liu, T. He, K. W. Lee, K. K. Leung, and A. Swami, "Dynamic control of data ferries under partial observations," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2010, pp. 1–6.

[10] G. Cardone, A. Cirri, A. Corradi, and L. Foschini, "The participact mobile crowd sensing living lab: The testbed for smart cities," *IEEE Commun. Mag.*, vol. 52, no. 10, pp. 78–85, Oct. 2014.

[11] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter, "Mobile crowd sensing in space weather monitoring: the Mahali project," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 22–28, Aug. 2014.

[12] S. Maharjan, Y. Zhang, and S. Gjessing, "Optimal incentive design for cloud-enabled multimedia crowdsourcing," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2470–2481, Dec. 2016.

[13] S. Nath, "Ace: Exploiting correlation for energy-efficient and continuous context sensing," in *Proc. ACM MobiSys'12*, Jun. 2012, pp. 29–42.

[14] Z. Zhou, J. Gong, Y. He, and Y. Zhang, "Software defined machine-to-machine communication for smart energy management," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 52–60, Oct. 2017.

[15] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.

[16] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A Stackelberg game approach," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 120–132, Mar. 2013.

[17] Y. Zhang, R. Yu, W. Yao, S. Xie, Y. Xiao, and M. Guizani, "Home M2M networks: Architectures, standards, and QoS improvement," *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 44–52, Apr. 2011.

[18] C. Watkins and J. C. Hellaby, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016.

[21] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. CVPR*, 2017, pp. 2261–2269.

[22] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[23] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," arXiv:1312.5602, 2013.

[24] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in Minecraft," in *Proc. ICML*, 2016.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[27] K. Andrej and F.-F. Li, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3128–3137.
[28] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, 1996.
[29] M. P. Deisenroth *et al.*, "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, no. 1–2, pp. 1–142, 2013.
[30] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[31] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. ICLR*, 2016.
[32] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016.
[33] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv:1603.04467, 2016.
[34] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," Jul. 2014. [Online]. Available: http://crawdad.org/roma/taxi/20140717

Bo Zhang received the M.S. and Ph.D. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 2010 and 2016, respectively.

He is currently a Postdoctoral Researcher. His research interests include participatory sensing and the Internet of things.

Chi Harold Liu (SM'15) received the B.Eng. degree from Tsinghua University, Beijing, China, and the Ph.D. degree from Imperial College, London, U.K., in 2006 and 2010, respectively.

He is currently a Full Professor and the Vice Dean with the School of Software, Beijing Institute of Technology, Beijing, China.

Jian Tang received the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2006.

He is currently a Full Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA. His research interests include cloud computing, big data, and wireless networking.

Zhiyuan Xu received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2015. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA.

His current research interests include deep learning and communication networks.

Jian Ma received the Ph.D. degree from Helsinki University of Technology, Espoo, Finland, in 1994.

He is a Guest Professor with the Department of Computer Science, Beijing University of Posts and Telecommunications.

He authored more than 160 conference and journal papers, and 4 books and book chapters.

Wendong Wang received the M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1990. He is currently a Professor with the Beijing University of Posts and Telecommunications. His current research interests include the next generation network architecture, IoT, wireless ad hoc, sensor and mesh networks, and mobile internet.